

LCP5

LCP-5 CPU CLUSTER DIA
CJKL5B0

COPYRIGHT (c) 1983-85
AH-T545B-M0
FICHE 01 OF 03

APR 1985

digital

Made In USA

Grid of data tables (fiches) with columns and rows of text.



LCP5

LCP-5 CPU CLUSTER DIA
CJKL5B0

COPYRIGHT (c) 1983-85
AH-T545B-MC
FICHE 02 OF 03

APR 1985

Digital

Made In USA

A microfiche card containing a grid of 14 columns and 14 rows of frames. Each frame contains a small, high-contrast image of a circuit board or component, likely related to the LCP-5 CPU cluster. The images are arranged in a regular grid pattern across the card.

5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763

.REM

IDENTIFICATION

PRODUCT CODE: AC-T544B-MC
PRODUCT NAME: CJKL580 LCP-5 CPU CLSTR DIAG
PRODUCT DATE: JANUARY 85
MAINTAINER: DIAGNOSTIC ENGINEERING

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

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C): 1983,1985 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DEC/X11

01

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58 P11 07 JAN-85 09:05

MACY11 30(1046) 07 JAN 85 09:28 PAGE 2 1

SEQ 0002

HISTORY

5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774

REVISION A

FIRST RELEASE OF DIAGNOSTIC

REVISION B

THIS REVISION WAS MADE TO CORRECT PROBLEMS
ENCOUNTERED WHILE RUNNING UNDER APT MODE
FOR CHANGES, SEE ;DD001

5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831

TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION
 - 1.1 ABSTRACT
 - 1.2 SYSTEM REQUIREMENTS
 - 1.3 RELATED DOCUMENTS AND STANDARDS
 - 1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
 - 1.5 ASSUMPTIONS
- 2.0 OPERATING INSTRUCTIONS
 - 2.1 LOADING AND STARTING PROCEDURE
 - 2.2 PROGRAM OPTIONS
 - 2.3 EXECUTION TIMES
- 3.0 ERROR INFORMATION
 - 3.1 ERROR REPORTING PROCEDURES
 - 3.2 ERROR HALTS
- 4.0 PERFORMANCE AND PROGRESS REPORTS
 - 4.1 PERFORMANCE REPORTS
 - 4.2 PROGRESS REPORTS
- 5.0 DEVICE INFORMATION TABLES
- 6.0 PROGRAM DESCRIPTION
 - 6.1 PROGRAM EXECUTION CHARACTERISTICS
 - 6.2 SUBTEST SUMMARIES
 - 6.3 SPECIAL SUBROUTINE DESCRIPTION
- 7.0 LISTING

1.0 ABSTRACT

THIS PROGRAM IS A GO-NOGO TEST FOR THE MICRO PDP-11 CPU BOARD. IT TESTS THE CPU INCLUDING EIS, THE MMU, THE FPP, THE LTC AND BOTH SLU'S. IT DOES NOT CONTAIN THE CAPABILITIES OF SCOPE LOOPING, ERROR RECOVERY OR PRINTING OF ERROR INFORMATION. ERROR HALTS DO INDICATE WHICH DEVICE FAILED TO ALLOW THE TECHNICIAN TO DETERMINE WHICH DIAGNOSTIC TO USE TO FIX THE BOARD OR WHAT FIELD REPLACEABLE UNIT (FRU) MAY FIX THE BOARD. THE PROGRAM WILL RUN UNDER THE ACT AND APT MANUFACTURING SYSTEMS AND IS CHAINABLE UNDER XXDP. THIS DIAGNOSTIC WAS CREATED FROM CJKDJB THE KDF11-B CLUSTER DIAGNOSTIC. THE DIFFERENCES ARE THE BDV TESTS ARE FOR 8K ROMS AND UPDATE WERE MADE SO THE PROGRAM WILL RUN UNDER UFD MODE.

1.1 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887

MICRO PDP 11 CPU MODULE

32K OF MEMORY

- THE SECOND SIU MUST HAVE TURN AROUND CONNECTOR.

B. SOFTWARE ENVIRONMENTS

- APT (MULTI-CPU TESTER)

- ACT

- XXDP (SLIDE)

- STAND-ALONE

1.2 RELATED DOCUMENTS AND STANDARDS

- ASSEMBLED WITH SYSMAC; SEE FIRST PAGE OF LISTING FOR REVISION NUMBER.

- MXXXX MODULE SPECIFICATION

- DIAGNOSTIC ENGINEERING FUNCTIONAL SPECIFICATION FOR SPECIAL MANUFACTURING TEST BGI-79-003-00-U.

- DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS 175-003-009-02.

1.3 PREREQUISITE DIAGNOSTICS

NONE

1.4 ASSUMPTIONS

THIS PROGRAM ASSUMES THE MACHINE IS UP SUFFICIENTLY TO ALLOW PROPER OPERATION OF THE MICRO-ODT OF THE DCF11-AA CHIP SET.

THE SYSTEM MUST HAVE PARITY MEMORY LOCATED IN THE FIRST 32K BLOCK.

THE SOFTWARE ASSUMES THAT THERE IS NO MEMORY OR DEVICES LOCATED AT OR BEYOND ADDRESS BIT 17 (64 KW). IF MEMORY IS THERE THE PROGRAM WILL FAIL WHEN IN THE EXTENDED ADDRESS TESTS. IF BIT 7 IS SET IN THE SWITCH REGISTER (176) THIS FAILURE CAN BE PREVENTED SINCE THAT PARTICULAR TEST WILL BE BYPASSED.

SEE PARAGRAPH 2.2.

5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

TO LOAD AND START THIS PROGRAM USE THE STANDARD PROCEDURES FOR THE DIAGNOSTIC SOFTWARE ENVIRONMENT THAT IS BEING USED.

2.2 PROGRAM OPTIONS

THIS PROGRAM USES THE SOFTWARE SWITCH LOCATION 176 IF PROGRAM IS NOT BEING RUN UNDER APT MODE (BIT 0 SET OF LOCATION \$ENV). IF PROGRAM IS BEING RUN IN APT MODE THE LOCATION \$SMREG IN THE APT ETABLE IS USED TO STORE OPERATING SWITCHES.

WARNING****THIS PROGRAM IS SET TO DO MINIMUM TESTING UNLESS CORRECTIVE ACTION IS TAKEN VIA THE SOFTWARE SWITCH REGISTER (176). BITS 1, 6,7-10 HAVE BEEN SET UP SUCH THAT THE PROGRAM WILL BYPASS CERTAIN TESTS UNLESS THE SWITCH REGISTER BIT IS SET. THIS CONDITION ALSO APPLIES WHEN UNDER CONTROL OF APT. THE APT SWITCH REGISTER, LOCATION 1022, MUST BE CORRECTLY SET AT APT LOAD TIME.

BIT #	DEFINITION
15-11	NOT USED
10	1 - TEST E102 SWITCHES 0 - INHIBIT TESTING E102 SWITCHES
9	1 - TEST PARITY ERROR DETECTION 0 - INHIBIT TESTING PARITY ERROR DETECTION
8	1 - USE THE Q22BE 0 - USE THE QBE IN PLACE OF THE Q22BE
7	1 - TEST THE UPPER 5 ADDRESS BITS FOR TIME OUT 0 - INHIBIT TESTING THE UPPER 5 ADRS BITS
6	1 - TEST USING A Q BUS EXERCISER (QBE OR Q22BE) 0 - INHIBIT TESTS THAT USE A Q BUS EXERCISER

5944 5 0 - PROGRAM RESERVED -- PROGRAM WILL CLEAR IF CIS CHIP SET NOT ON BOARD
5945 1 - PROGRAM RESERVED -- PROGRAM WILL SET IF CIS CHIP SET IS ON BOARD
5946 4 0 TEST SLU2 OF MICRO PDP-11
5947 1 INHIBIT TESTING OF SLU2
5948
5949 3 0 - TEST LTC OF MICRO PDP 11
5950 1 INHIBIT TESTING OF LTC
5951
5952 2 0 - TEST SLU1 OF MICRO PDP-11
5953 1 - INHIBIT TESTING OF SLU1
5954
5955 1 1 - TEST FPP INSTRUCTION SET
5956 0 - INHIBIT TESTING OF FPP
5957
5958 0 0 - TEST MEMORY MANAGEMENT UNIT
5959 1 - INHIBIT TESTING OF MEMORY MANAGEMENT UNIT
5960
5961
5962
5963

2.3 EXECUTION TIMES

5964
5965 FIRST PASS RUNTIME (WORST CASE).....45 SEC
5966 LONGEST TEST TIME.....30 SEC
5967 ADDITIONAL RUNTIME (EXTRA UNITS).....NONE
5968 LONGEST PASS TIME.....45 SEC
5969
5970

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999

THE PROGRAM DOES NOT TYPE OUT ANY ERROR REPORTS OF ITS OWN BUT TAKES ADVANTAGE OF THE HARDWARE FEATURE THAT TYPES THE PC WHEN A HALT OCCURS. WHEN AN ERROR IS DETECTED THE PROGRAM JUMPS TO ONE OF SEVEN HALT ROUTINES. THE ROUTINES SIMPLY MOVE A FATAL ERROR NUMBER INTO LOCATION \$FATAL, SET THE FATAL ERROR FLAG IN LOCATION \$MSGTY AND EITHER HALT OR IF ON APT DO A BRANCH DOT. THE OPERATOR HAS THREE WAYS TO DETERMINE THE FAILING DEVICE; 1) BY EXAMINING LOCATION \$FATAL, 2) BY DETERMINING THE HALT ADDRESS AND LOOKING UP THE ADDRESS IN THE LISTING AND 3) BY EXAMINING LOCATION \$TESTN WHICH WILL CONTAIN THE TEST NUMBER BEING EXECUTED.

3.2 ERROR HALTS

FOR DISCUSSION SEE SECTION 3.1. THE LABELS FOR THE HALTS AND THE DEVICE THEY INDICATE HAVING FAILED ARE:

CPUHLT: CPU
MMUHLT: MMU
FPPHLT: FPP

H

6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055

LTCMLT: LTC
 SL1MLT: SLU1
 SL2MLT: SLU2
 EXADHT: EXT ADRS TEST
 COMMLT: SYSTEM INTERACTION
 Q22MLT: Q22BE INTERRUPT TEST
 BDVMLT: BDV TEST

UPON RECEIVING THE ERROR HALT ADDRESS (PC) FROM THE MICRO-ODT THE OPERATOR CAN LOOK UP THESE TAGS IN THE SYMBOL TABLE AT THE END OF THE LISTING TO DETERMINE WHICH HALT WAS EXECUTED NOTE: THE PC SUPPLIED BY THE MICRO-ODT WILL BE THE HALT ADDRESS PLUS 2.

THE OPERATOR CAN DETERMINE WHICH TEST THE ERROR OCCURRED IN BY EXAMINING LOCATION "\$TESTN". THE TEST NUMBERS EQUATE TO FAILING DEVICES AS FOLLOWS:

DEVICE	CONTENTS OF \$TESTN
CPU	000001
MPU	000002
FPP	000003
SLU1	000004
LTC	000005
SLU2	000006
EXTND ADRS	000007
TESTS	
SYSTEM	
INTERACTION	000010
Q22BE	000011
BDV TESTS	000012

4.0 PERFORMANCE AND PROGRESS REPORTS

THE ONLY REPORT TYPED BY THIS PROGRAM IS THE END PASS MESSAGE WHICH IS:

CJKL580 END OF PASS #XXX

WHERE XXX IS THE DECIMAL NUMBER OF PASSES COMPLETED.

5.0 DEVICE INFORMATION TABLES

SLU1 RCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
I									I	I	I					I
									I	I						
RECEIVER DONE									I	I						
RECEIVER INTERRUPT ENABLE										I						

6056
 6057
 6058
 6059
 6060
 6061
 6062
 6063
 6064
 6065
 6066
 6067
 6068
 6069
 6070
 6071
 6072
 6073
 6074
 6075
 6076
 6077
 6078
 6079
 6080
 6081
 6082
 6083
 6084
 6085
 6086
 6087
 6088
 6089
 6090
 6091
 6092
 6093
 6094
 6095
 6096
 6097
 6098
 6099
 6100
 6101
 6102
 6103
 6104
 6105
 6106
 6107
 6108
 6109
 6110
 6111

SLU1 RBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I	I	I	I	I				I	I	I	I	I	I	I	I

		I	I	I	I											I
		I	I	I	I											I
ERROR-		I	I	I	I											I
OVERRUN			I	I	I											I
FRAME ERROR				I	I											I
RECEIVE PARITY					I											I
ERROR																I
RECEIVED DATA BITS (8)	-----															

SLU1 XCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I							I	I	I						I

								I	I							I
TRANSMITTER READY									I							I
TRANSMITTER INTERRUPT ENABLE																I

SLU1 XBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I								I	I	I	I	I	I	I	I

																I
																I
TRANSMITTER DATA BITS (8)	-----															

LTC CSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

	I									I	I					

											I					
											I					
LINE CLOCK																
INTERRUPT ENABLE	-----															

6112
 6113
 6114
 6115
 6116
 6117
 6118
 6119
 6120
 6121
 6122
 6123
 6124
 6125
 6126
 6127
 6128
 6129
 6130
 6131
 6132
 6133
 6134
 6135
 6136
 6137
 6138
 6139
 6140
 6141
 6142
 6143
 6144
 6145
 6146
 6147
 6148
 6149
 6150
 6151
 6152
 6153
 6154
 6155
 6156
 6157
 6158
 6159
 6160
 6161
 6162
 6163
 6164
 6165
 6166
 6167

SLU2 RCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								I	I	I						I
								I	I							
RECEIVER DONE									I							
INTERRUPT ENABLE																I

SLU2 RBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I	I	I	I	I				I	I	I	I	I	I	I	I
	I	I	I	I									I			
ERROR		I	I	I									I			
OVERRUN			I	I									I			
FRAME ERROR				I									I			
RECEIVER PARITY ERROR													I			
RECEIVER DATA BITS (8)													I			

SLU2 XCSR

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								I	I	I	I				I	I
									I	I						I
TRANSMITTER DONE										I						I
INTERRUPT ENABLE																I
BREAK																I

SLU2 XBUF

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	I								I	I	I	I	I	I	I	I
																I
TRANSMITTER DATA BITS (8)																I

6.0 PROGRAM DESCRIPTION

6163
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180
6181
6182
6183
6184
6185
6186
6187
6188
6189

6.1 PROGRAM EXECUTION CHARACTERISTICS

THIS PROGRAM RUNS THE SAME UNDER ALL DIAGNOSTIC MONITORS. WHEN THE TEST IS STARTED AT ADDRESS 200 OCTAL THE TESTING IS DONE AND ON COMPLETION THE TITLE IS TYPED AS PART OF THE END OF PASS MESSAGE.

6.2 SUB-TEST SUMMARIES

6.2.1 CENTRAL PROCESSING UNIT SUBTEST -

THESE TESTS CHECK THE BASIC INSTRUCTION SET AND ADDRESSING MODES, THE EXTENDED ELEVEN INSTRUCTION SET (EIS) AND TRAPS TESTING. IT IS EQUIVALENT TO CJKDB.

6191
6192
6193
6194
6195
6196
6197
6198
6199
6200
6201
6202
6203
6204
6205
6206
6207
6208
6209
6210
6211
6212
6213
6214
6215
6216
6217
6218
6219
6220
6221
6222
6223
6224
6225
6226
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246

6.2.2 MEMORY MANAGEMENT UNIT SUBTEST

THESE TESTS ARE THE SAME AS IN CJKDA, THE KEF11-AA TEST. THE PROGRAM BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC. NEXT THE MEMORY MANAGEMENT REGISTERS ARE CHECKED FOR DATA RELIABILITY, THEN RELOCATION CAPABILITIES (FORMATION OF A PHYSICAL ADDRESS FROM A VIRTUAL ADDRESS AND ASSOCIATED PAGE DESCRIPTOR (PDR) INFORMATION). FINALLY THE ABORT AND STATUS SEGMENTS OF THE LOGIC ARE CHECKED.

6.2.3 EXTENDED ADDRESS BIT TESTING AND PARITY ERROR LOGIC TEST

6.2.4 Q228E BR LEVEL TESTING

6.2.5 BDV TESTS

THE BDV TESTS CHECK OUT THE CHECKSUM IN THE DIAGNOSTIC BOOT ROMS OF THE MICRO PDP-11

6.2.6 FLOATING POINT PROCESSOR SUBTEST -

THE FLOATING POINT PROCESSOR SUBTEST CHECKS FLOATING POINT REGISTERS FIRST USING A LIMITED NUMBER OF FLOATING POINT INSTRUCTIONS. IT THEN VERIFIES THE REST OF THE FLOATING POINT INSTRUCTION SET USING A NUMBER OF DATA PATTERNS FOR EACH INSTRUCTION.

6.2.7 SERIAL LINE UNIT (SLU1) SUBTEST -

THESE TESTS CHECK THE SLU'S REGISTERS FOR ADDRESSING AND DATA HANDLING.

6.2.8 LINE TIME CLOCK (LTC) SUBTEST

FIRST THE REGISTER IS CHECKED FOR ADDRESSING AND BIT SETTING CAPABILITIES THEN THE INTERRUPT LOGIC IS CHECKED. THERE IS ALSO A

6.2.9 SERIAL LINE UNIT 2 SUBTEST -

THE TESTING DONE HERE IS SIMILAR AS FOR THE SLU1. AN EXTERNAL

JUMPER, TURN AROUND CONNECTOR, MUST BE PRESENT.

6.2.10 BLAST SUBTEST

THIS TEST CHECKS THE ABILITY OF THE MICRO PDP-11 TO HANDLE SYSTEM INTER ACTION. THE CPU HAS TO HANDLE DEVICES AT DIFFERENT PRIORITY LEVELS AND ARBITRATE BETWEEN THEM AND ITS OWN PRIORITY. THE TEST SETS UP ALL DEVICES TO INTERRUPT THEN ENABLES THEM ALL AT ONCE. THE SLU'S TRANSFER DATA UNTIL THEY TRANSFER 400(8) BYTES OR UNTIL ONE SECOND (60 TICKS) OF THE LINE CLOCK HAS BEEN RECEIVED. THE PROGRAM THEN VERIFIES THE NUMBER OF TRANSMITTER INTERRUPTS IS EQUAL TO THE NUMBER OF RECEIVER INTERRUPTS. FINALLY THE DATA TRANSFERRED BY EACH DEVICE IS CHECKED.

6.3 SPECIAL SUBROUTINE DESCRIPTIONS

THE ONLY SPECIAL SUBROUTINES ARE THE ERROR ROUTINES EACH SUBTEST HAS IS OWN. THE ROUTINES SIMPLY SET THE FATAL ERROR FLAG IN THE APT MAILBOX AND EITHER "BRANCH SELF" OR "HALT". THIS CHOICE IS DETERMINED IN THE INITIALIZE PORTION OF THE PROGRAM AND IS A "BRANCH SELF" IF RUNNING UNDER APT OR A "HALT" IF RUNNING UNDER ANY OTHER MONITOR.

6247
6248
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6270
6271
6272
6273
6354

6356
6357
6358
6359
6360 000240
6361 000007
6362 000006
6363 177776
6364 177564
6365 177566
6366 140000
6367 030000
6368 000001
6369 000006
6370 000006
6371 000003
6372 000001
6373 000005
6374 000002
6375 000000
6376 000003
6377 000004
6378 000004
6379 000014
6380 000030
6381 000020
6382 000034
6383 177564
6384 177560
6385 177562
6386 177566
6387 000240
6388 000240
6389 177776
6390 000077
6391 000010
6392 004700
6393 000100
6394
6395 177520
6396 177524
6397 177522
6398
6399 177776
6400 001000
6401 000600
6402 104377
6403 104777
6404 000001
6422

.TITLE CJKL580 LCP-5 CPU CLSTR DIAG
.ENABLE ABS
.NLIST CMD,MC,MD
.LIST ME
SCOPE=NOP
R7=#7
R6=#6
PS=177776
TPS=177564
TPB=177566
USRM=140000
PUSRM=30000
UFDSET=1
SP=#6
R6=#6
TAB=#3
LAST=#1
FIRST=#5
R2=#2
HLT=HALT
TRT=3
ITRAPS=4
RTRAPS=4
RTRAP4=14
RTRAP3=30
RTRAP2=20
RTRAP1=34
TTCSR=177564
TRCSR=177560
TKB=177562
TPB=177566
BELL=240
NOP=240
STATUS=177776
TRAPA=77
RTRAP=10
ILLA=004700
ILLB=100
PCR=177520
LSREG=177524
HWREG=177522
CC=177776
KERSTK=STBOT
USESTK=STBOT-200
EMTA=104377
TRAPC=104777
APTENV=1
.SBTTL ACT11 HOOKS

;RESERVED INST AND ILLEGAL ADDRESSES
;FOR TRACE TRAP
;FOR EMULATOR TRAP
;FOR IOT TRAP
;FOR TRAP INST

;HOOKS REQUIRED BY ACT11
\$SVPC= ;SAVE PC
.=46
\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP

(1)
(2)
(1)
(1) 000400
(1) 000046
(1) 000046 133226

(1) 000052 000052
(1) 000052 000000
(1) 000400
6423 001000
6424
(1)
(2)
(1)
(1) 001000
(1) 001000 000000
(1) 001002 000000
(1) 001004 000000
(1) 001006 000000
(1) 001010 000000
(1) 001012 000000
(1) 001014 000000
(1) 001016 000000
(1) 001020
(1) 001020 000
(1) 001021 000
(1) 001022 000000
(1) 001024 000000
(1) 001026 000000
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1) 001030
(1)
6425
(1)
(2)
(1)
(2)
(1) 001030
(1) 000024 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 001030
(1) 001030
(2)
(1)
(1)
(1)
(1) 001030
(1) 001030 000000
(1) 001032 001000
(1) 001034 000010
(1) 001036 000025
(1) 001040 000000
(1) 001042 000014
6426
6427
6428

. =52
.WORD 0 ;(2)SET LOC.52 TO ZERO
. =SVPC ; RESTORE PC
. =1000
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
\$MAIL: ;APT MAILBOX
\$MSGTY: .WORD AMSGTY ;MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;TEST NUMBER
\$PASS: .WORD APASS ;PASS COUNT
\$DEVCT: .WORD ADEVCT ;DEVICE COUNT
\$UNIT: .WORD AUNIT ;I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;MESSAGE ADDRESS
\$MSGLG: .WORD AMSGLG ;MESSAGE LENGTH
\$ETABLE: ;APT ENVIRONMENT TABLE
\$ENV: .BYTE AENV ;ENVIRONMENT BYTE
\$ENVM: .BYTE AENVM ;ENVIRONMENT MODE BITS
\$SWREG: .WORD ASWREG ;APT SWITCH REGISTER
\$USMR: .WORD AUSMR ;USER SWITCHES
\$CPUOP: .WORD ACPUOP ;CPU TYPE,OPTIONS
;*
;* 11/04-01,11/05-02,11/20-03,11/40-04,11/45-05
;* 11/70-06,PDQ-07,Q-10
;* BIT 10-REAL TIME CLOCK
;* BIT 9-FLOATING POINT PROCESSOR
;* BIT 8-MEMORY MANAGEMENT
\$ETEND:
.MEXIT
.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
. \$X= ;SAVE CURRENT LOCATION
. =24 ;SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;FOR APT START UP
. =44 ;POINT TO APT INDIRECT ADDRESS PNTR.
\$APTHDR ;POINT TO APT HEADER BLOCK
. = \$X ;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.
\$APTHD:
\$HIBTS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
\$TSTM: .WORD 10 ;RUN TIM OF LONGEST TEST
\$PASTM: .WORD 25 ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 0 ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
.WORD \$ETEND-\$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)
;*****
;SOME POINTERS TO CPU TRAP HANDLERS
;*****

```

6429          000004          .-4
6430 000004 021526          T04
6431 000006 000000          0
6432 000010 021530          T010
6433 000012 000000          0
6434 000014 021532          T014
6435 000016 000000          0
6436 000020 021534          T020
6437 000022 000000          0
6438          000034          .-34
6439 000034 021540          T034
6440 000036 000000          0
6441          000040          .-40
6442 000040 021542          T040
6443 000042 000000          0
6444          000114          .-114
6445 000114 021544          T0114
6446 000116 000000          0
6447          000244          .-244
6448 000244 021546          T0244
6449 000246 000000          0
6450 000250 021550          T0250
6451 000252 000000          0
6452
6453          000172          .-172
6454 000172 000000          MTFLAG: 0 ;MULTI-TESTER ACTIVE BIT
6455 000174 000000          DISPREG: 0 ;SOFTWARE DISPLAY REGISTER
6456 000176 000024          SWREG: 24 ;SOFTWARE SWITCH REGISTER
6457
6458          ;*****
6459          ;DATA TABLE FOR USE IN ADDRESSING MODE TESTS
6460          ;*****
6461          000370          .-370
6462 000370 000000 000000 000000          0,0,0,0,0,0
6463 000376 000000 000000 000000          1,1,-1
6464          ;*****
6465          ;SET UP STARTING ADDRESS
6466          001000          .-1000
6467 001000 000000          STBOT: .WORD 0 ;STACK POINTER
6468          ;*****
6469
6470
6471          000510          .-510          ;Q22BE DEVICE VECTOR
6472          ;AREA
6473
6474          000200          .-200
6475 000200 000167 000776          JMP START
6476 000204 012706 001000          MOV #STBOT,R6 ;SET STACK POINTER
6477 000210 012702 001004          MOV #TESTN,R2 ;SET MAILBOX POINTER
6478 000214 000137          JMP B(PC) ;JUMP TO SUBTEST
6479 000216 000000          0 ;ADDR. OF SUBTEST GOES HERE
6480
6481          001200          .-1200
6482 001200 000000          SAV42: .WORD 0 ;LOCATION TO SAVE MONITORS RETURN ADDRESS
6483          .SBTTL **STARTING OF CPU TEST **

```

D.

```

6484 001202          START:
(2)                ;; LCP/ORION ROUTINE TO SAVE EMULATOR AND PRIORITY
(2)                EMTSAV: TST      SAV30          ;; FIRST TIME THROUGH ?
(2) 001202 005767 000062      BNE      VMKOR          ;; BRANCH IF BEEN HERE ALREADY
(2) 001206 001034          BIT      @BITS,@#52      ;; ARE WE IN UFD MODE ?
(2) 001210 032737 000040 000052  BEQ      VMKOR          ;; LEAVE IF NOT
(2) 001216 001430          MOV      @-1,UFDPLG      ;; SET UFD FLAG
(2) 001220 012767 177777 000046  BIT      @BIT6,@#52      ;; ARE WE IN QUIET MODE ?
(2) 001226 032737 000100 000052  BEQ      1#            ;; BR IF NOT
(2) 001234 001403          MOV      @-1,UQUIET      ;; SET QUIET MODE
(2) 001236 012767 177777 000032 1#:    EMT      42          ;; GET ADDRESS OF XXDP DCA TABLE
(2) 001244 104042          CLR      42(R0)         ;; CLR XXDP "DRSERR"
(2) 001246 005060 000042          MOV      30,SAV30       ;; SAVE EMULATOR ADDRESS
(2) 001252 016767 176552 000010  MOV      32,SAV32       ;; SAVE EMULATOR PRIORITY LEVEL
(2) 001260 016767 176546 000004  BR       VMKOR          ;; GET AROUND TAG AREA
(2) 001266 000404          SAV30: .WORD 0          ;; PUT EMULATOR INFO HERE
(2) 001270 000000          SAV32: .WORD 0          ;; PUT PRIORITY LOCATION HERE
(2) 001272 000000          UFDPLG: .WORD 0        ;; USER FRIENDLY MODE FLAG
(2) 001274 000000          UQUIET: .WORD 0       ;; UFD QUIET MODE FLAG
(2) 001276 000000          VMKOR:
(2)                ;*****
(2)
(1)
6485 001300 032737 000040 000052      BIT      @40,@#52      ;DD001
6486 001306 001403          BEQ      CLM           ;DD001
6487 001310 104042          EMT      42           ;DD001
6488 001312 005060 000042          CLR      42(R0)       ;DD001
6489 001316 012737 000776 000202  CLM:    MOV      @776,@#202 ;DD001
6490 001324 013767 000042 177646  MOV      @42,SAV42     ;SAVE MONITOR RETURN ADDRESS
6491          ;MOV      @24,@#176 ;DD001
6492 001332 012737 021536 000030 5#:    MOV      @T030,@#30   ;SET UP THE EMT VECTOR
6493 001340 012737 000000 000032  MOV      @0,@#32
6494 001346 032737 000001 001020  BIT      @1,@#ENV     ;UNDER APT ?
6495 001354 001010          BNE      CONTIN       ;THEN SKIP MESSAGE PRINTOUT ?
6496 001356 032737 000100 000052  BIT      @100,@#52    ;UNDER UFD ?
6497 001364 001004          BNE      CONTIN       ;THEN SKIP MESSAGE PRINTOUT ?
6498 001366 012700 133635          MOV      @STRMSG,R0   ;START MESSAGE
6499 001372 004767 131766          JSR      PC,TYPE
6500 001376 012737 000000 001006  CONTIN: MOV      @0,@#PASS ;CLEAR PASS COUNT
6501 001404 012737 133314 000024  RESTR:  MOV      @PURDN,@#24 ;SET UP FOR POWER FAIL
6502 001412 012706 001000          MOV      @STBOT,R6   ;SET UP STACK
6503 001416 012737 001446 000004  MOV      @1,@#4       ;SET UP FOR TIMEOUT IF NO MULTI TESTER
6504 001424 012737 000340 000006  MOV      @340,@#6
6505 001432 012737 000002 164000  MOV      @2,@#164000  ;SET BIT1 FOR MULTI TESTER
6506 001440 012737 000001 000172  MOV      @1,@#MTFLAG  ;SET FLAG TO INDICATE MULTI-TESTER
6507 001446 012737 021526 000004 1#:    MOV      @T04,@#4     ;SET TRAP CATCHER
6508 001454 012737 000000 000006  MOV      @0,@#6       ;SET HALT BACK IN LOCATION 6
6509 001462 012706 001000          MOV      @STBOT,R6   ;INITIALIZE STACK POINTER
6510 001466 012737 000001 001004  MOV      @1,@#TESTN   ;SET TEST NUMBER TO 1
6511 001474 012737 000000 001002  MOV      @0,@#FATAL   ;CLEAR ERROR INDICATOR
6512 001502 012737 000000 001000  MOV      @0,@#MSGTY   ;CLEAR MESSAGE TYPE(FOR APT)
6513
6514 001510 005067 176010          CLR      LSREG        ;THIS WILL TURN ON THE 4 LEDS
6515
6516 001514 105737 001020          TSTB     @#ENV        ;RUNNING ON APT

```

6517 001520 001036
6518 001522 012737 000000 002346
6519 001530 012737 000000 050706
6520 001536 012737 000000 051626
6521 001544 012737 000000 053442
6522 001552 012737 000000 124512
6523 001560 012737 000000 126664
6524 001566 012737 000000 125740
6525 001574 012737 000000 131534
6526 001602 012737 000000 132520
6527 001610 012737 000000 054442

BNE TS1 ;IF YES DO BRANCH SELF ON ERROR
MOV #0,B#CPUHLT ;IF NOT THEN PUT A HALT IN ON ERROR
MOV #0,B#YUHLT
MOV #0,B#EXADHT
MOV #0,B#Q22HLT
MOV #0,B#FPHLT
MOV #0,B#LTCHLT
MOV #0,B#SL1HLT
MOV #0,B#SL2HLT
MOV #0,B#COMHLT
MOV #0,B#BDVHLT

6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550

; THIS TEST EXECUTES EVERY POSSIBLE BRANCH WITH EVERY POSSIBLE
; CONDITION CODE COMBINATION.
; THE ROUTINE USES TWO TABLES. THE BRANCH TABLE HOLDS ALL THE
; POSSIBLE BRANCH INSTRUCTIONS. THE OTHER TABLE (YNTAB) HOLDS BIT MAPS FOR
; EACH BRANCH. A ONE IN THE BIT MAP INDICATES THAT THE CORRESPONDING
; BRANCH INSTRUCTION SHOULD BRANCH FOR THE CONDITION CODE SETTING WHICH
; CORRESPONDS TO THE BIT POSITION WITHIN THE MAP. FOR EXAMPLE IF THE LEFT
; MOST BIT IS A ONE THEN THE CORRESPONDING BRANCH INSTRUCTION SHOULD BRANCH
; WHEN THE CONDITION CODES ARE 0.
; THE ROUTINE CONSISTS OF NESTED LOOPS; THE OUTER LOOP SETS UP
; ALL THE POSSIBLE BRANCH INSTRUCTIONS. THE INNER LOOP SETS UP EVERY POSSIBLE
; CONDITION CODE FOR EACH BRANCH.
; THE BIT MAP IS USED TO SET THE ADDRESS LOCATION IN TWO
; JUMP MODE 3 INSTRUCTIONS. THE ADDRESSES ARE CHANGED TO ALLOW THE
; PROGRAM TO CONTINUE OR JUMP TO AN ERROR ROUTINE DEPENDING UPON
; WHETHER IT HANDLED THE BRANCH INSTRUCTION CORRECTLY.
; AT ANY ERROR HALT, LOCATION, BRH, HOLDS THE BRANCH INSTRUCTION
; UNDER TEST AND LOCATION, CC, HOLDS THE VALUE OF THE CONDITION CODES
; AT THE TIME THE BRANCH WAS EXECUTED.

6551 (2)
6552 (2) 001616
6553 001616 012700 021432
6554 001622 012704 021470
6555 001626 012767 000017 000130
6556 001634 012067 000110
6557 001640 012401
6558 001642 012767 177777 000074
6559 001650 012703 000020
6560 001654 005267 000064
6561 001660 032701 100000
6562 (1) 001664 013705 177776
6563 (1) 001670 042705 177773
6564 (1) 001674 000165 001700
6565 (1) 001700 000167 000020
6566 001704 012767 001766 000042
6567 001712 012767 001762 000040
6568 001720 000167 000014
6569 001724 012767 001762 000022
6570 001732 012767 001766 000020

TEST 1 TEST THE BRANCH ROM

TS1:
SETUP: MOV #BRTAB,R0 ;INITIALIZE BRANCH TABLE POINTER
MOV #YNTAB,R4 ;INITIALIZE YES/NO BRANCH MAP POINTER
MOV #15,BRCT ;INITIALIZE BRANCH TABLE COUNT
SETBR: MOV (R0)+,BRH ;GET NEXT BRANCH INST.
MOV (R4)+,R1 ;GET NEXT BRANCH MAP
MOV #-1,CC1 ;INITIALIZE CONDITION CODE VALUE
MOV #16,R3 ;INITIALIZE CONDITION CODE COUNT
SETCC: INC CC1 ;SET FOR NEXT CC VALUE
BIT #100000,R1 ;SEE IF SHOULD BR W/ THESE CC'S
MOV #0177776,R5 ;SIMULATE A JNE
BIC #177773,R5 ; (JUMP NOT EQUAL)
JMP .+4(R5) ; TO SET2BR
MOV #CONT,NBR ;SET TO CONTINUE IF NO BRANCH
MOV #ER,YBR ;SET TO REPORT ERROR IF BRANCH
JMP AROUND ;GO AROUND OPPOSITE CONDITION
SET2BR: MOV #ER,NBR ;SET TO REPORT ERROR IF NO BRANCH
MOV #CONT,YBR ;SET TO CONTINUE IF BRANCH

6567	001740	006101		AROUN:	ROL	R1		;UPDATE BIT MAP
6568								
6569	001742	012737		CC1:	MOV	(PC),B(PC)		;SET CONDITION CODE
6570	001744	000000			0			;NEW CC VALUE GOES HERE
6571	001746	177776				177776		
6572	001750	000000		BRH:	0			;BRANCH INST. GOES HERE
6573	001752	000137			JMP	B(PC)		;THIS JUMP IF NO BRANCH
6574	001754	000000		NBR:	0			;WHERE TO GO IF NO BRANCH OCCURS
6575	001756	000137			JMP	B(PC)		;THIS JUMP IF BRANCH OCCURS
6576	001760	000000		YBR:	0			;WHERE TO GO IF BRANCH OCCURS
6577	001762			ER:				
(2)	001762	000551			BR	ERROR1		
6578	001764	000000		BRCT:	0			
6579	001766	005303		CONT:	DEC	R3		;CC'S DONE?
6580	001770	013705	177776		MOV	#0177776,R5		;SIMULATE A JNE
(1)	001774	042705	177773		BIC	#177773,R5		; (JUMP NOT EQUAL)
(1)	002000	000165	002004		JMP	.+4(R5)		; TO SETCC
(1)	002004	000167	177644		JMP	SETCC		
6581	002010	005367	177750		DEC	BRCT		;BR'S DONE?
6582	002014	013705	177776		MOV	#0177776,R5		;SIMULATE A JNE
(1)	002020	042705	177773		BIC	#177773,R5		; (JUMP NOT EQUAL)
(1)	002024	000165	002030		JMP	.+4(R5)		; TO SETBR
(1)	002030	000167	177600		JMP	SETBR		
6583	002034	012700	000357		MOV	#357,R0		;IF THIS TEST IS DONE SET UP R0 FOR THE NEXT
6584								;SEVEN TESTS. THIS IS SAVING 4 LOCATIONS PER
6585								;TEST WHICH I NEED BECAUSE BRANCHES WERE OUT
6586								;OF BOUNDS.

 ;TEST 2 TEST TRAP OF RESERVED INSTRUCTION

6624				TS2:				
6625					MOV	#STBOT,SP		;INITIALIZE THE STACK POINTER
6626	(3)				MOV	#1,RTRAP		;SET UP NEW PC IN VECTOR
(4)					MOV	R0,RTRAP+2		;SET UP NEW PSW IN VECTOR
(3)	002040				CLR	STATUS		;CLEAR PRESENT (OLD) STATUS
(1)	002040	012706	001000		TRAPA			;DO TRAP
(1)	002044	012767	002066	175736				
(1)	002052	010067	175734					
(1)	002056	005067	175714					
(1)	002062	000077						
(1)	002064			8:				
(3)	002064	000510			BR	ERROR1		;INSTRUCTION FAILED TO TRAP
(1)	002066	020067	175704	1:	CMP	R0,STATUS		;IS NEW STATUS CORRECT
(1)	002072	001105			BNE	ERROR1		;NEW STATUS WRONG
(1)	002074	020627	000774	2:	CMP	SP,#STBOT-4		;DID STACK DECREMENT CORRECTLY
(1)	002100	001102			BNE	ERROR1		;STACK DID NOT DECREMENT CORRECTLY
(1)	002102	022767	002064	3:	CMP	#0,STBOT-4		;WAS PROPER PC SAVED
(1)	002110	001076	176664		BNE	ERROR1		;PROPER PC WAS NOT SAVED
(1)	002112	005767	176660	4:	TST	STBOT-2		;WAS OLD PSW SAVED
(1)	002116	001073			BNE	ERROR1		;WRONG PSW SAVED
(1)	002120	062706	000004	5:	ADD	#4,SP		;RESET STACK POINTER
(1)	002124	012767	002146		MOV	#6,RTRAP		;SET UP NEW PC IN VECTOR
(1)	002132	005067	175654		CLR	RTRAP+2		;SET UP NEW PSW IN VECTOR
(1)	002136	010037	177776		MOV	R0,#STATUS		;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1)	002142	000077			TRAPA			;DO TRAP
(3)	002144	000460			BR	ERROR1		;INSTRUCTION FAILED TO TRAP
(1)	002146	005767	175624	6:	TST	STATUS		;IS NEW PSW CORRECT
(1)	002152	001055			BNE	ERROR1		;NEW PSW WRONG

```

(1) 002154 020067 176616 74: CMP R0,STBOT-2 ;WAS OLD STATUS STORED
(1) 002160 001052 BNE ERROR1 ;OLD STATUS WRONG
6627 ;*****
(3) ;TEST 3 TEST TRAP OF TRAP INSTRUCTION
(4) ;*****
(3) TS3:
(1) 002162 012706 001000 MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
(1) 002166 012767 002210 MOV #1,RTRAP1 ;SET UP NEW PC IN VECTOR
(1) 002174 010067 175636 MOV R0,RTRAP1+2 ;SET UP NEW PSW IN VECTOR
(1) 002200 005067 175572 CLR STATUS ;CLEAR PRESENT (OLD) STATUS
(1) 002204 104400 TRAP ;DO TRAP
(1) 002206 84:
(3) 002206 000437 BR ERROR1 ;INSTRUCTION FAILED TO TRAP
(1) 002210 020067 175562 14: CMP R0,STATUS ;IS NEW STATUS CORRECT
(1) 002214 001034 BNE ERROR1 ;NEW STATUS WRONG
(1) 002216 020627 000774 24: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 002222 001031 BNE ERROR1 ;STACK DID NOT DECREMENT CORRECTLY
(1) 002224 022767 002206 176542 34: CMP #8,STBOT-4 ;WAS PROPER PC SAVED
(1) 002232 001025 BNE ERROR1 ;PROPER PC WAS NOT SAVED
(1) 002234 005767 176536 44: TST STBOT-2 ;WAS OLD PSW SAVED
(1) 002240 001022 BNE ERROR1 ;WRONG PSW SAVED
(1) 002242 062706 000004 54: ADD #4,SP ;RESET STACK POINTER
(1) 002246 012767 002270 175560 MOV #6,RTRAP1 ;SET UP NEW PC IN VECTOR
(1) 002254 005067 175556 CLR RTRAP1+2 ;SET UP NEW PSW IN VECTOR
(1) 002260 010037 177776 MOV R0,#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 002264 104777 TRAPC ;DO TRAP WITH LOWER BYTE ALL ONES
(3) 002266 000407 BR ERROR1 ;INSTRUCTION FAILED TO TRAP
(1) 002270 005767 175502 64: TST STATUS ;IS NEW PSW CORRECT
(1) 002274 001004 BNE ERROR1 ;NEW PSW WRONG
(1) 002276 020067 176474 74: CMP R0,STBOT-2 ;WAS OLD STATUS STORED
(1) 002302 001001 BNE ERROR1 ;OLD STATUS WRONG
6628 002304 000436 BR CPUHLT+34 ;GET OVER ERROR CALL TO NEXT TEST IF NO ERROR
6629 ;*****
6630 ;THIS ERROR IS USED FOR THE ENTIRE CPU,TRAPS AND EIS PORTION OF
6631 ;THIS TEST
6632 ;*****
6633 002306 004767 131116 ERROR1: JSR PC,ABORT ;CHECK IF WE ARE UNDER UFD ?
6634 002312 012737 000001 001002 MOV #1,#FATAL ;SET UP FATAL ERROR NUMBER
6635 002320 012767 000001 176452 MOV #1,#MSGTY ;SET FATAL ERROR FLAG
6636 002326 032737 000001 001020 BIT #1,#ENV ;UNDER APT
6637 002334 001004 BNE CPUHLT ;YES, THEN DO NOT PRINT
6638 002336 012700 002350 MOV #CPUMSG,R0
6639 002342 004767 131016 JSR PC,TYPE ;TYPE MSG
6640 002346 000777 CPUHLT: BR .
6641
6642 002350 040506 046111 042105 CPUMSG: .ASCIZ /FAILED DURING CPU TESTS/<12><15>
002356 042040 051125 047111
002364 020107 050103 020125
002372 042524 052123 005123
002400 000015
6643 .EVEN
6644
6645 ;*****
(3) ;TEST 4 TEST TRAP OF IOT INSTRUCTION
(4) ;*****
(3) 002402 TS4:

```

```
(1) 002402 012706 001000          MOV    #STBOT,SP      ;INITIALIZE THE STACK POINTER
(1) 002406 012767 002430 175404  MOV    #1#,RTRAP2    ;SET UP NEW PC IN VECTOR
(1) 002414 010067 175402          MOV    R0,RTRAP2+2  ;SET UP NEW PSW IN VECTOR
(1) 002420 005067 175352          CLR    STATUS        ;CLEAR PRESENT (OLD) STATUS
(1) 002424 000004          IOT                ;DO TRAP
(1) 002426          8#:          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(3) 002426 000727          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(1) 002430 020067 175342          1#:    CMP    R0,STATUS ;IS NEW STATUS CORRECT
(1) 002434 001324          BNE   ERROR1       ;NEW STATUS WRONG
(1) 002436 020627 000774          2#:    CMP    SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 002442 001321          BNE   ERROR1       ;STACK DID NOT DECREMENT CORRECTLY
(1) 002444 022767 002426 176322  3#:    CMP    #8#,STBOT-4 ;WAS PROPER PC SAVED
(1) 002452 001315          BNE   ERROR1       ;PROPER PC WAS NOT SAVED
(1) 002454 005767 176316          4#:    TST   STBOT-2   ;WAS OLD PSW SAVED
(1) 002460 001312          BNE   ERROR1       ;WRONG PSW SAVED
(1) 002462 062706 000004          5#:    ADD    #4,SP    ;RESET STACK POINTER
(1) 002466 012767 002510 175324  MOV    #6#,RTRAP2    ;SET UP NEW PC IN VECTOR
(1) 002474 005067 175322          CLR    RTRAP2+2     ;SET UP NEW PSW IN VECTOR
(1) 002500 010037 177776          MOV    R0,#STATUS   ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 002504 000004          IOT                ;DO TRAP
(3) 002506 000677          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(1) 002510 005767 175262          6#:    TST   STATUS    ;IS NEW PSW CORRECT
(1) 002514 001274          BNE   ERROR1       ;NEW PSW WRONG
(1) 002516 020067 176254          7#:    CMP    R0,STBOT-2 ;WAS OLD STATUS
(1) 002522 001271          BNE   ERROR1       ;OLD STATUS WRONG
6646          ;*****
(3)          ;TEST 5 TEST TRAP OF EMT INSTRUCTION
(4)          ;*****
(3) 002524          TSS:
(1) 002524 012706 001000          MOV    #STBOT,SP    ;INITIALIZE THE STACK POINTER
(1) 002530 012767 002552 175272  MOV    #1#,RTRAP3    ;SET UP NEW PC IN VECTOR
(1) 002536 010067 175270          MOV    R0,RTRAP3+2  ;SET UP NEW PSW IN VECTOR
(1) 002542 005067 175230          CLR    STATUS        ;CLEAR PRESENT (OLD) STATUS
(1) 002546 104000          EMT                ;DO TRAP
(1) 002550          8#:          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(3) 002550 000656          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(1) 002552 020067 175220          1#:    CMP    R0,STATUS ;IS NEW STATUS CORRECT
(1) 002556 001253          BNE   ERROR1       ;NEW STATUS WRONG
(1) 002560 020627 000774          2#:    CMP    SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 002564 001250          BNE   ERROR1       ;STACK DID NOT DECREMENT CORRECTLY
(1) 002566 022767 002550 176200  3#:    CMP    #8#,STBOT-4 ;WAS PROPER PC SAVED
(1) 002574 001244          BNE   ERROR1       ;PROPER PC WAS NOT SAVED
(1) 002576 005767 176174          4#:    TST   STBOT-2   ;WAS OLD PSW SAVED
(1) 002602 001241          BNE   ERROR1       ;WRONG PSW SAVED
(1) 002604 062706 000004          5#:    ADD    #4,SP    ;RESET STACK POINTER
(1) 002610 012767 002632 175212  MOV    #6#,RTRAP3    ;SET UP NEW PC IN VECTOR
(1) 002616 005067 175210          CLR    RTRAP3+2     ;SET UP NEW PSW IN VECTOR
(1) 002622 010037 177776          MOV    R0,#STATUS   ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 002626 104377          EMTA              ;DO TRAP WITH LOWER BYTE ALL ONES
(3) 002630 000626          BR     ERROR1       ;INSTRUCTION FAILED TO TRAP
(1) 002632 005767 175140          6#:    TST   STATUS    ;IS NEW PSW CORRECT
(1) 002636 001223          BNE   ERROR1       ;NEW PSW WRONG
(1) 002640 020067 176132          7#:    CMP    R0,STBOT-2 ;WAS OLD STATUS STORED
(1) 002644 001220          BNE   ERROR1       ;OLD STATUS WRONG
6647 002646 000401          BR     ERROR2+2     ;WE MUST GET OVER ERROR CALL AT END OF THIS TEST
6648          ;*****
```

```
6649 ;THIS ERROR IS NEEDED BECAUSE BRANCHES IN TRAP TESTS BEYOND HERE CAN NOT
6650 ;REACH ERROR1.
6651 ;*****
6652 002650 000616 ERROR2: BR ERROR1
6653
6654
6655 ;*****
(3) ;TEST 6 TEST TRAP OF TRACE-TRAP INSTRUCTION
(4) ;*****
(3) T56:
(1) 002652 012706 001000 MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
(1) 002656 012767 002700 175130 MOV #1$,RTRAP4 ;SET UP NEW PC IN VECTOR
(1) 002664 010067 175126 MOV R0,RTRAP4+2 ;SET UP NEW PSW IN VECTOR
(1) 002670 005067 175102 CLR STATUS ;CLEAR PRESENT (OLD) STATUS
(1) 002674 000003 TRT ;DO TRAP
(1) 002676 8$:
(3) 002676 000764 BR ERROR2 ;INSTRUCTION FAILED TO TRAP
(1) 002700 020067 175072 1$: CMP R0,STATUS ;IS NEW STATUS CORRECT
(1) 002704 001361 BNE ERROR2 ;NEW STATUS WRONG
(1) 002706 020627 000774 2$: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 002712 001356 BNE ERROR2 ;STACK DID NOT DECREMENT CORRECTLY
(1) 002714 022767 002676 176052 3$: CMP #8$,STBOT-4 ;WAS PROPER PC SAVED
(1) 002722 001352 BNE ERROR2 ;PROPER PC WAS NOT SAVED
(1) 002724 005767 176046 4$: TST STBOT-2 ;WAS OLD PSW SAVED
(1) 002730 001347 BNE ERROR2 ;WRONG PSW SAVED
(1) 002732 062706 000004 5$: ADD #4,SP ;RESET STACK POINTER
(1) 002736 012767 002760 175050 MOV #6$,RTRAP4 ;SET UP NEW PC IN VECTOR
(1) 002744 005067 175046 CLR RTRAP4+2 ;SET UP NEW PSW IN VECTOR
(1) 002750 010037 177776 MOV R0,#STATUS ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 002754 000003 TRT ;DO TRAP
(3) 002756 000734 BR ERROR2 ;INSTRUCTION FAILED TO TRAP
(1) 002760 005767 175012 6$: TST STATUS ;IS NEW PSW CORRECT
(1) 002764 001331 BNE ERROR2 ;NEW PSW WRONG
(1) 002766 020067 176004 7$: CMP R0,STBOT-2 ;WAS OLD STATUS STORED
(1) 002772 001326 BNE ERROR2 ;OLD STATUS WRONG
6656 ;PDP-11 ILLEGAL AND ADDRESS INSTRUCTION TEST
6657 ;ALL INSTRUCTIONS THAT ARE RESERVED
6658 ;SHOULD TRAP TO LOCATION 4, AND THE
6659 ;PC THAT POINTS TO THE TRAPPING INSTRUCTION
6660 ;SHOULD BE PLACED ON THE STACK
6661
6662 ;*****
(3) ;TEST 7 TEST TRAP OF ILLEGAL INSTRUCTION
(4) ;*****
(3) T57:
(1) 002774 012706 001000 MOV #STBOT,SP ;INITIALIZE THE STACK POINTER
(1) 003000 012767 003022 174776 MOV #1$,RTRAP5 ;SET UP NEW PC IN VECTOR
(1) 003006 010067 174774 MOV R0,RTRAP5+2 ;SET UP NEW PSW IN VECTOR
(1) 003012 005067 174760 CLR STATUS ;CLEAR PRESENT (OLD) STATUS
(1) 003016 000100 JMP #0 ;DO TRAP
(1) 003020 8$:
(3) 003020 000713 BR ERROR2 ;INSTRUCTION FAILED TO TRAP
(1) 003022 020067 174750 1$: CMP R0,STATUS ;IS NEW STATUS CORRECT
(1) 003026 001310 BNE ERROR2 ;NEW STATUS WRONG
(1) 003030 020627 000774 2$: CMP SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 003034 001305 BNE ERROR2 ;STACK DID NOT DECREMENT CORRECTLY
```



```
(1) 003036 022767 003020 175730 3:  CMP    #0,STBOT-4    ;WAS PROPER PC SAVED
(1) 003044 001301                BNE    ERROR2        ;PROPER PC WAS NOT SAVED
(1) 003046 005767 175724        4:  TST    STBOT-2        ;WAS OLD PSW SAVED
(1) 003052 001276                BNE    ERROR2        ;WRONG PSW SAVED
(1) 003054 062706 000004        5:  ADD    #4,SP         ;RESET STACK POINTER
(1) 003060 012767 003102 174716  MOV    #6,RTRAPS5    ;SET UP NEW PC IN VECTOR
(1) 003066 005067 174714        CLR    RTRAPS+2      ;SET UP NEW PSW IN VECTOR
(1) 003072 010037 177776        MOV    R0,#STATUS    ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 003076 000100                JMP    #0            ;DO TRAP
(3) 003100 000663                BR     ERROR2        ;INSTRUCTION FAILED TO TRAP
(1) 003102 005767 174670        6:  TST    STATUS        ;IS NEW PSW CORRECT
(1) 003106 001260                BNE    ERROR2        ;NEW PSW WRONG
(1) 003110 020067 175662        7:  CMP    R0,STBOT-2    ;WAS OLD STATUS STORED
(1) 003114 001255                BNE    ERROR2        ;OLD STATUS WRONG
```

```
6663 ;*****
(3) ;TEST 10 TEST TRAP OF ALL ILLEGAL INSTRUCTION
(4) ;*****
```

```
(3) 003116 TS10:
(1) 003116 012706 001000 174654  MOV    #STBOT,SP    ;INITIALIZE THE STACK POINTER
(1) 003122 012767 003144        MOV    #1,RTRAPS5   ;SET UP NEW PC IN VECTOR
(1) 003130 010067 174652        MOV    R0,RTRAPS+2 ;SET UP NEW PSW IN VECTOR
(1) 003134 005067 174636        CLR    STATUS       ;CLEAR PRESENT (OLD) STATUS
(1) 003140 004000                JSR    #0,#0        ;DO TRAP
(1) 003142 8:
(3) 003142 000642                BR     ERROR2        ;INSTRUCTION FAILED TO TRAP
(1) 003144 020067 174626        1:  CMP    R0,STATUS    ;IS NEW STATUS CORRECT
(1) 003150 001237                BNE    ERROR2        ;NEW STATUS WRONG
(1) 003152 020627 000774        2:  CMP    SP,#STBOT-4 ;DID STACK DECREMENT CORRECTLY
(1) 003156 001234                BNE    ERROR2        ;STACK DID NOT DECREMENT CORRECTLY
(1) 003160 022767 003142 175606  3:  CMP    #0,STBOT-4    ;WAS PROPER PC SAVED
(1) 003166 001230                BNE    ERROR2        ;PROPER PC WAS NOT SAVED
(1) 003170 005767 175602        4:  TST    STBOT-2        ;WAS OLD PSW SAVED
(1) 003174 001225                BNE    ERROR2        ;WRONG PSW SAVED
(1) 003176 062706 000004        5:  ADD    #4,SP         ;RESET STACK POINTER
(1) 003202 012767 003224 174574  MOV    #6,RTRAPS5    ;SET UP NEW PC IN VECTOR
(1) 003210 005067 174572        CLR    RTRAPS+2      ;SET UP NEW PSW IN VECTOR
(1) 003214 010037 177776        MOV    R0,#STATUS    ;SET UP OLD STATUS FOR COMPARISON AFTER TRAP
(1) 003220 004000                JSR    #0,#0        ;DO TRAP
(3) 003222 000612                BR     ERROR2        ;INSTRUCTION FAILED TO TRAP
(1) 003224 005767 174546        6:  TST    STATUS        ;IS NEW PSW CORRECT
(1) 003230 001207                BNE    ERROR2        ;NEW PSW WRONG
(1) 003232 020067 175540        7:  CMP    R0,STBOT-2    ;WAS OLD STATUS STORED
(1) 003236 001204                BNE    ERROR2        ;OLD STATUS WRONG
```

```
6664 ;*****
6665
```

```
6666 .SBTTL DATA PATH TESTS
6667 ;
6668 ; THE DATA PATH TESTS ARE USED TO VERIFY THAT VARIOUS
6669 ; DATA PATTERNS CAN BE SUCCESSFULLY MOVED THROUGH THE DATA PATHS
6670 ; MOVE AND COMPARE MODE 2,3 INSTRUCTIONS ARE USED TO PASS AND
6671 ; TEST VARIOUS DATA PATTERNS IN THE DATA PATHS.
6672 ; THE TEST EXERCISES THE INTERNAL DATA PATHS, AND THE UNIBUS
6673 ; DATA TRANSCIEVERS.
6674 ; IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0)
6675 ; TO SEE WHICH BITS OF THE DATA PATH ARE FAILING.
6676 003240 012737 002306 000030  MOV    #ERROR1,#30 ;SET UP VECTOR FOR ERROR CALLS
```

6677	003246	012737	000340	000032	MOV	#340,0#32	;SET UP NEW PSW
6678	003254	012737	021526	000004	MOV	#T04,0#4	;SET UP FOR UNEXPECTED TRAP TO 4
6679	003262	012737	021530	000010	MOV	#T010,0#10	;SET UP FOR UNEXPECTED TRAP TO 10
6680	003270	012737	021532	000014	MOV	#T014,0#14	;SET UP FOR UNEXPECTED TRAP TO 14
6681	003276	012737	021540	000034	MOV	#T034,0#34	;SET UP FOR UNEXPECTED TRAP TO 34
6682	003304	012737	021542	000040	MOV	#T040,0#40	;SET UP FOR UNEXPECTED TRAP TO 40 DD001

6683
6684
6685
(2)
(3)
(2)

```
*****  
;TEST 11 TEST OF ZEROES IN THE DATA PATH  
*****
```

6686	003312	012737	000000	000000	TS11: MOV	#0,0#0	;MOVE ZEROES THRU ADDRESS LINES, DATA ;LINES AND INTERNAL PATHS
6687							;SUCCESSFUL?
6688	003320	005737	000000		TST	0#0	
6689	003324	001401			BEQ	TS12	
(3)	003326	104000			EMT		;DATA INCORRECT

6690
6691
(2)
(3)
(2)

```
*****  
;TEST 12 TEST OF PATTERN 125252 IN DATA PATH  
*****
```

6692	003330	012737	125252	000000	TS12: MOV	#125252,0#0	;MOVE ALTERNATING ONES AND ZEROES ;THRU DATA PATHS
6693							;SUCCESSFUL
6694	003336	022737	125252	000000	CMP	#125252,0#0	
6695	003344	001401			BEQ	TS13	
(3)	003346	104000			EMT		;DATA INCORRECT

6696
6697
(2)
(3)
(2)

```
*****  
;TEST 13 TEST OF PATTERN 052525 IN DATA PATH  
*****
```

6698	003350	012737	052525	000000	TS13: MOV	#052525,0#0	;MOVE ALTERNATING ZEROES AND ONES ;THRU DATA PATH
6699							;SUCCESSFUL?
6700	003356	022737	052525	000000	CMP	#052525,0#0	
6701	003364	001401			BEQ	TS14	
(3)	003366	104000			EMT		;DATA INCORRECT

6702
6703
(2)
(3)
(2)

```
*****  
;TEST 14 TEST OF ALL ONES IN DATA PATH  
*****
```

6704	003370	012737	177777	000000	TS14: MOV	#177777,0#0	;MOVE ONES THRU DATA PATH
6705	003376	022737	177777	000000	CMP	#177777,0#0	;SUCCESSFUL
6706	003404	001401			BEQ	TS15	
(3)	003406	104000			EMT		;DATA INCORRECT

6707
6708
6709
6710

```
*****  
;SBTTL B-REGISTER TEST  
;
```

6711
6712
6713
6714
6715
6716

```
THE B-REGISTER (LOCATION 0) SHIFTING LOGIC TESTS ARE USED  
;TO TEST THAT THE B-REGISTER CAN HOLD VARIOUS DATA PATTERNS AND THAT  
;THE ASSOCIATED LOGIC SUPPORTS THE SHIFTING FUNCTIONS WITHIN THE  
;B-REGISTER AND C-BIT.  
;A ONE IS SHIFTED THROUGH EVERY BIT IN THE B-REGISTER AND C-BIT IN  
;BOTH DIRECTIONS.
```

L2

THE B-REGISTER ITSELF IS TESTED IN ITS ABILITY AS A BUFFER AND AS
A SHIFT REGISTER. DATA IS ALSO PASSED THROUGH THE DATA PATH AND ALU,
IF THESE TESTS FAIL, EXAMINE THE TARGET LOCATION (LOC. 0) TO SEE
WHICH BITS OF THE B-REGISTER MAY BE FAILING.

;TEST 15 SHIFT BIT 0 TO BIT 1

TS15:
CLC ;CLEAR CARRY BIT
MOV #1,B#0 ;LOAD A 1
ROL B#0 ;SHIFT LEFT
CMP #2,B#0 ;SUCCESSFUL
BEQ TS16
EMT ;BIT 1 NOT SET

;TEST 16 SHIFT CARRY INTO BIT 0

TS16:
MOV #0,B#0 ;CLEAR LOCATION
SEC ;SET CARRY
ROL B#0 ;ROTATE CARRY BIT TO BIT 0
BCC CARRY1
EMT ;CARRY CLEAR
CARRY1: CMP #1,B#0 ;BIT 0 SET
BEQ TS17
EMT ;BIT 0 NOT SET

;TEST 17 LEFT SHIFT FROM BIT 0 TO C-BIT

TS17:
MOV #1,B#0 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
SHL: INC R0 ;INCREMENT BIT COUNTER
BEQ SHLE ;BR TO ERROR HALT IF BIT IS LOST
ROL B#0 ;SHIFT LEFT ONE POSITION
BCC SHL ;BRANCH IF C-BIT NOT SET
BEQ TS20
SHLE: EMT ;LEFT SHIFTING LOGIC FAILED

;TEST 20 SHIFT BIT 15 TO BIT 14

TS20:
MOV #100000,B#0 ;SET BIT 15
CLC ;CLEAR CARRY
ROR B#0 ;SHIFT BIT 15 TO BIT 14
CMP #40000,B#0 ;SUCCESSFUL
BEQ TS21
EMT ;BIT 14 NOT SET

6717
6718
6719
6720
6721
6722
(2)
(3)
(2) 003410
6723 003410 000241
6724 003412 012737 000001 000000
6725 003420 006137 000000
6726 003424 022737 000002 000000
6727 003432 001401
(3) 003434 104000
6728
6729
(2)
(3)
(2) 003436
6730 003436 012737 000000 000000
6731 003444 000261
6732 003446 006137 000000
6733 003452 103001
(2) 003454 104000
6734 003456 022737 000001 000000
6735 003464 001401
(3) 003466 104000
6736
6737
(2)
(3)
(2) 003470
6738 003470 012737 000001 000000
6739 003476 012700 177757
6740 003502 000241
6741 003504 005200
6742 003506 001404
6743 003510 006137 000000
6744 003514 103373
6745 003516 001401
(2) 003520
(3) 003520 104000
6746
6747
(2)
(3)
(2) 003522
6748 003522 012737 100000 000000
6749 003530 000241
6750 003532 006037 000000
6751 003536 022737 040000 000000
6752 003544 001401
(3) 003546 104000
6753
6754

(2)
(3)
(2) 003550
6755 003550 012737 100000 000000
6756 003556 012700 177757
6757 003562 000241
6758 003564 005200
6759 003566 001404
6760 003570 006037 000000
6761 003574 103373
6762 003576 001401
(2) 003600
(3) 003600 104000

6763
6764
6765
6766
6767
6768
6769
6770
6771
6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784

(2)
(3)
(2) 003602
6785
6786 003602 012700 000000
6787 003606 005700
6788 003610 001401
(3) 003612 104000
6789

6790
(2)
(3)
(2) 003614
6791 003614 012700 125252
6792 003620 020027 125252
6793 003624 001401
(3) 003626 104000
6794
6795
(2)
(3)

```
;TEST 21 RIGHT SHIFT FROM BIT 15 TO C-BIT  
;*****  
TS21:  
MOV #100000,B#0 ;SET BIT 15  
MOV #-21,R0 ;SET BIT COUNTER  
CLC ;CLEAR C-BIT  
SHR: INC R0 ;INCREMENT BIT COUNTER  
BEQ SHRE ;BR TO ERROR HALT IF BIT IS LOST  
ROR B#0 ;ROTATE RIGHT ONE POSITION  
BCC SHR ;BRANCH IF C-BIT CLEAR  
BEQ TS22  
SHRE: EMT ;RIGHT SHIFT LOGIC FAILED
```

```
;*****  
;SBTTL SCRATCH PAD TESTS  
;  
; THE SCRATCH PAD TESTS ARE USED TO VERIFY THAT VARIOUS  
; DATA PATTERNS CAN BE SUCCESSFULLY HELD IN THE SCRATCH PAD  
; CIRCUITRY. MOVE AND COMPARE INSTRUCTIONS ARE USED TO TEST THAT  
; R0 CAN HOLD VARIOUS DATA PATTERNS. EACH DATA PATTERN IS  
; MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR SCOPING. THE  
; SUCCESSFUL COMPLETION OF THESE TESTS SHOULD VERIFY THE CIRCUITRY EXTERNAL  
; TO THE SCRATCH PAD ITSELF.  
; THE REMAINDER OF THE GENERAL REGISTERS ARE TESTED BY MOVING  
; A BIT INTO BIT 0 OF THE REGISTER AND SHIFTING IT LEFT ONE  
; BIT AT A TIME INTO THE CARRY BIT. THE RESULT IS THEN CHECKED TO INSURE THAT  
; NO BITS WERE PICKED. THE PROCEDURE IS THEN REPEATED UNDER OPPOSITE  
; CONDITIONS. THE GENERAL REGISTER AND THE CARRY BIT ARE SET TO  
; ALL ONES, AND A ZERO IS SHIFTED LEFT FROM BIT 0 INTO THE CARRY BIT.  
; THE RESULT IS THEN CHECKED TO INSURE THAT NO ZEROES WERE PICKED.  
; AT THIS POINT ALL OF THE GENERAL REGISTERS HAVE BEEN EXERCISED  
; AS WELL AS REGISTER 11.
```

```
;*****  
;TEST 22 TEST IF R0 CAN HOLD ALL ZEROES  
;*****  
TS22:  
MOV #0,R0 ;MOVE ZEROES TO R0  
TST R0 ;SUCCESSFUL?  
BEQ TS23  
EMT ;R0 NOT 0
```

```
;*****  
;TEST 23 TEST IF R0 CAN HOLD ONES AND ZEROES  
;*****  
TS23:  
MOV #125252,R0 ;MOVE ALTERNATING ONES AND ZEROES TO R0  
CMP R0,#125252 ;SUCCESSFUL?  
BEQ TS24  
EMT ;R0 NOT 125252
```

```
;*****  
;TEST 24 TEST IF R0 CAN HOLD ZEROES AND ONES  
;*****
```

(2) 003630
6796 003630 012700 052525
6797 003634 020027 052525
6798 003640 001401
(3) 003642 104000
6799
6800
(2)
(3)
(2) 003644
6801 003644 012700 177777
6802 003650 020027 177777
6803 003654 001401
(3) 003656 104000
6804
6805
(2)
(3)
(2) 003660
6806 003660 012701 000001
6807 003664 012700 177757
6808 003670 000241
6809 003672 005200
6810 003674 001
6811 003676 006101
6812 003700 103374
6813 003702 001401
(2) 003704
(3) 003704 104000
6814
6815
(2)
(3)
(2) 003706
6816 003706 012701 177776
6817 003712 012700 177757
6818 003716 000261
6819 003720 005200
6820 003722 001405
6821 003724 006101
6822 003726 103774
6823 003730 022701 177777
6824 003734 001401
(2) 003736
(3) 003736 104000
6825
(2)
(3)
(2) 003740
6826 003740 012702 000001
6827 003744 012700 177757
6828 003750 000241
6829 003752 005200
6830 003754 001403
6831 003756 006102
6832 003760 103374

```
TS24:
MOV #052525,R0 ;MOVE ALTERNATING ZEROES AND ONES TO R0
CMP R0,#052525 ;SUCCESSFUL?
BEQ TS25
EMT ;RO NOT 52525

;*****
;TEST 25 TEST IF R0 CAN HOLD ALL ONES
;*****
TS25:
MOV #177777,R0 ;MOVE ALL ONES TO R0
CMP R0,#177777 ;SUCCESSFUL?
BEQ TS26
EMT ;RO NOT 177777

;*****
;TEST 26 TEST IF R1 CAN HOLD A ONE IN ALL BITS
;*****
TS26:
MOV #1,R1 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG1: INC R0 ;INCREMENT BIT COUNTER
BEQ REG1E ;BR TO ERROR HALT IF BIT IS LOST
ROL R1 ;ROTATE 1 POSITION
BCC REG1 ;ALL DONE
BEQ TS27
REG1E: EMT ;FAILURE WITH R1

;*****
;TEST 27 TEST IF R1 CAN HOLD A ZERO IN ALL BITS
;*****
TS27:
MOV #-2,R1 ;SET ALL ONES IN R1 EXCEPT FOR BIT 0
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG1A: INC R0 ;INCREMENT COUNTER
BEQ R1ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R1 ;ROTATE 1 POSITION
BCS REG1A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R1 ;CHECK DATA IN R1
BEQ TS30
R1ERR: EMT ;FAILURE WITH R1

;*****
;TEST 30 TEST IF R2 CAN HOLD A ONE IN ALL BITS
;*****
TS30:
MOV #1,R2 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG2: INC R0 ;INCREMENT BIT COUNTER
BEQ REG2E ;BR TO ERROR HALT IF BIT IS LOST
ROL R2 ;ROTATE 1 POSITION
BCC REG2 ;ALL DONE
```

```

6833 003762 001401
(2) 003764
(3) 003764 104000
6834
6835
(2)
(3)
(2) 003766
6836 003766 012702 177776
6837 003772 012700 177757
6838 003776 000261
6839 004000 005200
6840 004002 001405
6841 004004 006102
6842 004006 103774
6843 004010 022702 177777
6844 004014 001401
(2) 004016
(3) 004016 104000
6845
6846
(2)
(3)
(2) 004020
6847 004020 012703 000001
6848 004024 012700 177757
6849 004030 000241
6850 004032 005200
6851 004034 001403
6852 004036 006103
6853 004040 103374
6854 004042 001401
(2) 004044
(3) 004044 104000
6855
6856
(2)
(3)
(2) 004046
6857 004046 012703 177776
6858 004052 012700 177757
6859 004056 000261
6860 004060 005200
6861 004062 001405
6862 004064 006103
6863 004066 103774
6864 004070 022703 177777
6865 004074 001401
(2) 004076
(3) 004076 104000
6866
6867
(2)
(3)
(2) 004100
6868 004100 012704 000001

```

```

REG2E: BEQ TS31
          EMT ;FAILURE WITH R2
;.....
;TEST 31 TEST IF R2 CAN HOLD A ZERO IN ALL BITS
;.....
TS31:
          MOV #2,R2 ;SET ALL ONES IN R2 EXCEPT FOR BIT 0
          MOV #21,R0 ;SET BIT COUNTER
          SEC ;SET C-BIT
REG2B: INC R0 ;INCREMENT BIT COUNTER
          BEQ R2ERR ;BR TO ERROR HALT IF COUNTER=0
          ROL R2 ;ROTATE 1 POSITION
          BCS REG2B ;CONTINUE UNTIL C-BIT IS CLEAR
          CMP #1,R2 ;CHECK DATA IN R2
          BEQ TS32
R2ERR:
          EMT ;FAILURE WITH R2
;.....
;TEST 32 TEST IF R3 CAN HOLD A ONE IN ALL BITS
;.....
TS32:
          MOV #1,R3 ;SET BIT 0
          MOV #21,R0 ;SET BIT COUNTER
          CLC ;CLEAR C-BIT
REG3: INC R0 ;INCREMENT BIT COUNTER
          BEQ REG3E ;BR TO ERROR HALT IF BIT IS LOST
          ROL R3 ;ROTATE 1 POSITION
          BCC REG3 ;ALL DONE
          BEQ TS33
REG3E:
          EMT ;FAILURE WITH R3
;.....
;TEST 33 TEST IF R3 CAN HOLD A ZERO IN ALL BITS
;.....
TS33:
          MOV #2,R3 ;SET ALL ONES IN R3 EXCEPT FOR BIT 0
          MOV #21,R0 ;SET BIT COUNTER
          SEC ;SET C-BIT
REG3A: INC R0 ;INCREMENT BIT COUNTER
          BEQ R3ERR ;BR TO ERROR HALT IF COUNTER=0
          ROL R3 ;ROTATE 1 POSITION
          BCS REG3A ;CONTINUE UNTIL C-BIT IS CLEAR
          CMP #1,R3 ;CHECK DATA
          BEQ TS34
R3ERR:
          EMT ;FAILURE WITH R3
;.....
;TEST 34 TEST IF R4 CAN HOLD A ONE IN ALL BITS
;.....
TS34:
          MOV #1,R4 ;SET BIT 0

```

C3

6869 004104 012700 177757
6870 004110 000241
6871 004112 005200
6872 004114 001403
6873 004116 006104
6874 004120 103374
6875 004122 001401
(2) 004124
(3) 004124 104000
6876
6877

MOV # -21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG4: INC R0 ;INCREMENT BIT COUNTER
BEQ REG4E ;BR TO ERROR HALT IF BIT IS LOST
ROL R4 ;ROTATE 1 POSITION
BCC REG4 ;ALL DONE
BEQ TS35
REG4E: EMT ;FAILURE WITH R4

;*****
;TEST 35 TEST IF R4 CAN HOLD A ZERO IN ALL BITS
;*****
TS35:

(2) 004126
6878 004126 012704 177776
6879 004132 012700 177757
6880 004136 000261
6881 004140 005200
6882 004142 001405
6883 004144 006104
6884 004146 103774
6885 004150 022704 177777
6886 004154 001401
(2) 004156
(3) 004156 104000
6887
6888
6889

MOV # -2,R4 ;SET ALL ONES IN R4 EXCEPT FOR BIT 0
MOV # -21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG4A: INC R0 ;INCREMENT BIT COUNTER
BEQ R4ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R4 ;ROTATE 1 POSITION
BCS REG4A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP # -1,R4 ;CHECK DATA
BEQ TS36
R4ERR: EMT ;FAILURE WITH R4

;*****
;TEST 36 TEST IF R5 CAN HOLD A ONE IN ALL BITS
;*****
TS36:

(2) 004160
6890 004160 012705 000001
6891 004164 012700 177757
6892 004170 000241
6893 004172 005200
6894 004174 001403
6895 004176 006105
6896 004200 103374
6897 004202 001401
(2) 004204
(3) 004204 104000
6898
6899

MOV #1,R5 ;SET BIT 0
MOV # -21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG5: INC R0 ;INCREMENT BIT COUNTER
BEQ REG5E ;BR TO ERROR HALT IF BIT IS LOST
ROL R5 ;ROTATE 1 POSITION
BCC REG5 ;ALL DONE
BEQ TS37
REG5E: EMT ;FAILURE WITH R5

;*****
;TEST 37 TEST IF R5 CAN HOLD A ZERO IN ALL BITS
;*****
TS37:

(2) 004206
6900 004206 012705 177776
6901 004212 012700 177757
6902 004216 000261
6903 004220 005200
6904 004222 001405
6905 004224 006105
6906 004226 103774
6907 004230 022705 177777
6908 004234 001401
(2) 004236

MOV # -2,R5 ;SET ALL ONES IN R5 EXCEPT FOR BIT 0
MOV # -21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG5A: INC R0 ;INCREMENT BIT COUNTER
BEQ R5ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R5 ;ROTATE 1 POSITION
BCS REG5A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP # -1,R5 ;CHECK DATA
BEQ TS40
R5ERR: EMT ;FAILURE WITH R5

```

(3) 004236 104000
6909
6910
(2)
(3)
(2) 004240
6911 004240 012706 000001
6912 004244 012700 177757
6913 004250 000241
6914 004252 005200
6915 004254 001403
6916 004256 006106
6917 004260 103374
6918 004262 001401
(2) 004264
(3) 004264 104000
6919
6920
(2)
(3)
(2) 004266
6921 004266 012706 177776
6922 004272 012700 177757
6923 004276 000261
6924 004300 005200
6925 004302 001405
6926 004304 006106
6927 004306 103774
6928 004310 022706 177777
6929 004314 001401
(2) 004316
(3) 004316 104000
6930
6931
6932
6933
6934
6935
6936
6937
6938
6939
6940
6941
6942
6943
6944
6945
(2)
(3)
(2) 004320
6946 004320 012706 001000
6947 004324 012737 000000 177776
6948 004332 005737 177776
6949 004336 001401
(3) 004340 104000

```

```

EMT ;FAILURE WITH R5
;*****
;TEST 40 TEST IF R6 CAN HOLD A ONE IN ALL BITS
;*****
TS40:
MOV #1,R6 ;SET BIT 0
MOV #-21,R0 ;SET BIT COUNTER
CLC ;CLEAR C-BIT
REG6: INC RC ;INCREMENT BIT COUNTER
BEQ REG6E ;BR TO ERROR HALT IF BIT IS LOST
ROL R6 ;ROTATE 1 POSITION
BCC REG6 ;ALL DONE
BEQ TS41
REG6E:
EMT ;FAILURE WITH R6
;*****
;TEST 41 TEST IF R6 CAN HOLD A ZERO IN ALL BITS
;*****
TS41:
MOV #-2,R6 ;SET ALL ONES IN R6 EXCEPT FOR BIT C
MOV #-21,R0 ;SET BIT COUNTER
SEC ;SET C-BIT
REG6A: INC R0 ;INCREMENT BIT COUNT
BEQ R6ERR ;BR TO ERROR HALT IF COUNTER=0
ROL R6 ;ROTATE 1 POSITION
BCS REG6A ;CONTINUE UNTIL C-BIT IS CLEAR
CMP #-1,R6 ;CHECK DATA
BEQ TS42
R6ERR:
EMT ;FAILURE WITH R6
;*****
.SBTTL PSW TESTS
;
; THE PSW TESTS ARE USED TO VERIFY THAT VARIOUS DATA
; PATTERNS CAN BE SUCCESSFULLY HELD IN THE PSW AND THAT THE
; PSW ADDRESSING LOGIC IS FUNCTIONING. MOVE AND COMPARE INSTRUCTIONS
; ARE USED TO TEST THAT THE PSW CAN HOLD VARIOUS DATA PATTERNS.
; EACH DATA PATTERN IS MOVED AND TESTED IN A SMALL LOOP CONVENIENT FOR
; SCOPING.
;
; THE PSW REGISTER IS TESTED, THE CC INPUTS ARE TESTED
; LATER IN THE MICROCODE TESTS. SETTING OF THE T-BIT BY THE
; TEST PATTERNS IS PURPOSELY AVOIDED. TESTING OF THE
; T-BIT TRAP CIRCUITRY IS LEFT FOR THE TRAP TEST.
;*****
;TEST 42 TEST IF PSW WILL HOLD ZEROES
;*****
TS42:
MOV #STBOT,R6
MOV #0,B#PS ;SET PSW TO ZERO
TST B#PS ;SUCCESSFUL
BEQ TS43
EMT ;PSW NOT 0

```


6950
6951
(2)
(3)
(2) 004342
6952 004342 012737 000252 177776
6953 004350 023727 177776 000252
6954 004356 001401
(3) 004360 104000
6955
6956
(2)
(3)
(2) 004362
6957 004362 012737 000105 177776
6958 004370 023727 177776 000105
6959 004376 001401
(3) 004400 104000
6960
6961
(2)
(3)
(2) 004402
6962 004402 012737 000357 177776
6963 004410 023727 177776 000357
6964 004416 001401
(3) 004420 104000
6965
6966
6967
6968
6969
6970
6971
6972
6973
6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993

```

;*****
;TEST 43      TEST IF PSW WILL HOLD ONES AND ZEROES
;*****
TS43:
MOV      #252,00PS      ;MOVE ALT. ONES AND ZEROES TO PSW
CMP      00PS,0252      ;SUCCESSFUL?
BEQ      TS44
EMT                      ;PSW NOT 252

;*****
;TEST 44      TEST IF PSW (EXCEPT T-BIT) WILL HOLD ZEROES AND ONES
;*****
TS44:
MOV      #105,00PS      ;MOVE ALT. ONES AND ZEROES TO PSW
CMP      00PS,0105      ;SUCCESSFUL?
BEQ      TS45
EMT                      ;PSW NOT 105

;*****
;TEST 45      TEST IF PSW (EXCEPT T-BIT) WILL HOLD ALL ONES
;*****
TS45:
MOV      #357,00PS      ;MOVE ONES TO PSW
CMP      00PS,0357      ;SUCCESSFUL
BEQ      TS46
EMT                      ;PSW NOT 357

;*****
.SBTTL  MICROCODE TESTS
;
;   THE TEST EXERCISES BRANCHES IN THE MICROCODE BY
; TESTING AT LEAST ONE INSTRUCTION FROM EVERY CLASS OF INSTRUCTION IN
; ALL POSSIBLE MODES.  FOR EXAMPLE, TO TEST THE SINGLE OPERAND INSTRUCTIONS,
; AT LEAST ONE SINGLE OPERAND INSTRUCTION IS VERIFIED IN ALL UNIQUE
; ADDRESSING MODES.  BYTE MODES ARE ALSO TESTED.  AS EACH NEW
; MODE IS INTRODUCED THE SAME INSTRUCTION IS TRIED AND TESTED IN
; A SMALL LOOP CONVENIENT FOR SCOPING.  THE TEST IS SET UP USING
; ONLY INSTRUCTIONS AND ADDRESSING MODES WHICH HAVE BEEN PREVIOUSLY
; VERIFIED.
;   IF THESE TESTS FAIL, CHECK THE RESULTS FOR A CLUE TO THE
; FAULT.
;
;*****

;*****
;
;   THE CLR INSTRUCTION IS USED TO INTRODUCE EACH ADDRESSING
; MODE WITH THE SINGLE OPERAND INSTRUCTION.  FOLLOWING THE SEQUENCE CHECK,
; THE CLR INSTRUCTION IS EXECUTED AND A BRANCH TEST IS EXECUTED WHICH
; CHECKS THAT THE Z-BIT WAS PROPERLY SET.  THIS TEST CAN CHECK IR DECODE
; AND MICROCODE FOR SOP INSTRUCTIONS WITH MODE 0.  FOLLOWING THIS TEST
; SEVERAL OTHER SOP INSTRUCTIONS ARE INTRODUCED WITH MODE 0.  THESE
; INSTRUCTIONS MAINPULATE DATA AND SERVE TO CHECK THE DATA RESULTS
; OF THE SOP INSTRUCTIONS IN THIS TEST.  THE DATA IN THIS TEST IS

```

```

6994
6995
6996
(2)
(3)
(2) 004422
6997 004422 005000
6998 004424 001401
(2) 004426 104000
6999 004430 005200
7000 004432 005100
7001 004434 005200
7002 004436 100401
(2) 004440 104000
7003 004442 005100
7004 004444 001401
(3) 004446 104000
7005
7006
7007
7008
7009
7010
7011
7012
7013
7014
7015
7016
(2)
(3)
(2) 004450
7017 004450 005000
7018 004452 005300
7019 004454 100401
(2) 004456 104000
7020 004460 000261
7021 004462 005500
7022 004464 001007
7023 004466 000261
7024 004470 005600
7025 004472 100004
7026 004474 005100
7027 004476 005200
7028 004500 005300
7029 004502 001401
(2) 004504
(3) 004504 104000
7030
7031
7032
7033
7034
7035
7036
7037

```

```

; OPERATED ON BY EACH INSTRUCTION WITHOUT REINITIALIZING.
;
; *****
; TEST 46 TEST MODE 0 USING SOP INST.
; *****
TS46:
      CLR      RO      ; TRY THE CLEAR INST.
      BEQ     SOPOA
      EMT
SOP0A: INC      RO      ; CLR DID NOT SET Z-BIT
      COM     RO      ; TRY THE INCREMENT INST.
      INC     RO      ; TRY COMPLEMENT
      BMI     SOPOB
      EMT
SOP0B: COM     RO      ; NEGATE DID NOT SET N-BIT
      BEQ     TS47    ; TRY COMPLEMENT INST.
      EMT
; CUMULATIVE RESULT OF CLR, INC, NEG AND COM INSTS. FAILED

```

```

; *****
;
; THIS TEST INTRODUCES THE REMAINING SOP INSTRUCTIONS AND TESTS
; THEM IN MODE 0. THE PURPOSE IS TO PROVIDE A BASELINE OF
; INSTRUCTIONS FOR USE IN THE SUBSEQUENT TESTS. SINCE THE MICROCODE FOR
; THESE INSTRUCTIONS IS IDENTICAL TO THAT ALREADY TESTED, ANY TROUBLE
; SHOOTING EFFORTS SHOULD BE AIMED AT THE ACTUAL IR DECODE AND ALU
; FUNCTIONING.
;
; *****
; TEST 47 TEST REMAINDER OF SOP INSTS IN MODE 0
; *****
TS47:
      CLR      RO      ; INITIALIZE
      DEC     RO      ; TRY DECREMENT INST.
      BMI     SOPOC
      EMT
SOP0C: SEC
      ADC     RO      ; N-BIT NOT SET ON DEC
      BNE     SOPOD  ; INITIALIZE CARRY
      SBC     RO      ; TRY ADD CARRY INST
      BPL     SOPOD
      COM     RO      ; INITIALIZE CARRY
      INC     RO      ; TRY SUBTRACT-CARRY INST
      DEC     RO
      BEQ     TS50
SOP0D: EMT
; CUMULATIVE RESULT OF ADC, SBC, COM, INC AND DEC INSTS. F

```

```

; *****
;
; THIS TEST INTRODUCES THE BYTE CONTROL LOGIC OF THE PROCESSOR.
; THE MODE 0 BYTE MICROCODE IS TESTED. THE METHOD AND SEQUENCE
; OF TESTING IS THE SAME AS THAT USED IN THE SOP MODE 0 TESTS.
;
; *****

```

```

(2)
(3)
(2) 004506
7038 004506 105000
7039 004510 001401
(2) 004512 104000
7040 004514 105100
7041 004516 100002
7042 004520 105200
7043 004522 001401
(2) 004524
(3) 004524 104000
7044
7045
7046
7047
7048
7049
7050
7051
7052
7053
(2)
(3)
(2) 004526
7054 004526 005000
7055 004530 005010
7056 004532 001401
(2) 004534 104000
7057 004536 005310
7058 004540 100003
7059 004542 000261
7060 004544 005510
7061 004546 001401
(2) 004550
(3) 004550 104000
7062
7063
7064
7065
7066
7067
7068
7069
7070
(2)
(3)
(2) 004552
7071 004552 005000
7072 004554 005010
7073 004556 005110
7074 004560 105010
7075 004562 001401
(2) 004564 104000
7076 004566 005210
7077 004570 100005

```

```

;TEST 50 TEST MODE 0 EVEN BYTE USING SOP INST
;*****
TS50:
      CLR      RO          ;TRY CLEARING EVEN BYTE OF REGISTER
      BEQ      SOP80A
      EMT
SOP80A: COMB      RO
      BPL      SOP80B
      INCB     RO          ;TRY INCREMENTING EVEN BYTE OF REGISTER>>
      BEQ      TS51
SOP80B:
      EMT                ;TEST CUMULATIVE RESULT OF ABOVE BYTE INST.
;*****
;
; THIS TEST USES THE CLR INSTRUCTION TO INTRODUCE AND TEST
; SINGLE OPERAND MODE 1 INSTRUCTIONS. AGAIN, THE CLR INSTRUCTION
; IS USED TO INTRODUCE THE MICROCODE AND TO TEST THAT THE PROPER
; CONDITION CODES ARE SET. OTHER SOP INSTRUCTIONS ARE USED TO MANIPULATE
; COMMON DATA TO VERIFY THAT THE CORRECT DATA IS PRODUCED.
;*****
;TEST 51 TEST MODE 1 USING SOP INST.
;*****
TS51:
      CLR      RO          ;INITIALIZE RO
      CLR      (RO)        ;TRY CLEAR INST W/MODE 1
      BEQ      SOP1A
      EMT
SOP1A: DEC      (RO)
      BPL      SOP1B
      SEC
      ADC      (RO)        ;INITIALIZE CARRY
      BEQ      TS52        ;TRY ADD-CARRY W/MODE 1
SOP1B:
      EMT                ;TEST CUMULATIVE RESULT OF ABOVE INST
;*****
;
; THIS TEST VERIFIES THE BYTE INSTRUCTION MICROCODE FOR MODE 1
; SINGLE OPERAND INSTRUCTIONS.
; THIS IS THE FIRST PLACE THE SIGN EXTEND LOGIC IS EXERCISED
; AND VERIFIED.
;*****
;TEST 52 TEST MODE 1 EVEN BYTE USING SOP INST
;*****
TS52:
      CLR      RO          ;INITIALIZE RO
      CLR      (RO)        ;INITIALIZE LOC. 0
      COM      (RO)
      CLRB     (RO)        ;TRY TO CLEAR BYTE 0
      BEQ      SOPB1A
      EMT
SOPB1A: INC      (RO)
      BPL      SOPB1B
      BPL      SOPB1B

```

7078 004572 105110
7079 004574 105210
7080 004576 100002
7081 004600 105210
7082 004602 001401
(2) 004604
(3) 004604 104000

COMB (RO) ;COMPLEMENT: ODD BYTE = 376
INCB (RO) ;INC: ODD BYTE = 377
BPL SOPB1B
INCB (RO) ;INCREMENT ODD BYTE=0
BEQ TS53
SOPB1B: EMT ;CHECK CUMULATIVE RESULT OF ABOVE INST

7083
7084
7085
7086
7087
7088
7089
7090
7091
7092
7093
7094

; THIS TEST VERIFIES THAT SINGLE OPERAND BYTE INSTRUCTIONS WILL
; FUNCTION CORRECTLY FOR ODD BYTES.
; THIS IS THE FIRST TIME THAT ADDRESS LINE 0 HAS BEEN
; EXERCISED. CHECKS ARE MADE THAT THE PROPER BYTE IS MODIFIED AND
; THE CONDITION CODES ARE CHECKED. IT IS ALSO VERIFIED THAT THE UNADDRESSED
; BYTE IS NOT ALTERED BY THE INSTRUCTION.

(2) 004606
(3) 004606 005000
7095 004606 005000
7096 004610 005010
7097 004612 005110
7098 004614 005200
7099 004616 105010
7100 004620 001401
(2) 004622 104000
7101 004624 005300
7102 004626 005210
7103 004630 005200
7104 004632 105110
7105 004634 105210
7106 004636 100002
7107 004640 105210
7108 004642 001401
(2) 004644
(3) 004644 104000

; TEST 53 TEST MODE 1 ODD BYTE USING SOP INST

TS53:
CLR RO ;INITIALIZE RO
CLR (RO) ;INITIALIZE LOC. 0
COM (RO)
INC RO ;RO=ODD BYTE
CLRB (RO) ;TRY TO CLEAR BYTE 1
BEQ SOPB1C
EMT ;CLRB DID NOT SET Z-BIT
SOPB1C: DEC RO ;RO=WORD ADDR.
INC (RO) ;INCREMENT TO TEST WORD
INC RO ;RO=ODD BYTE
COMB (RO) ;TRY TO COMPLEMENT BYTE 1
INCB (RO)
BPL SOPB1D
INCB (RO) ;TRY TO INCREMENT BYTE 1
BEQ TS54
SOPB1D: EMT ;TEST CUMULATIVE RESULT OF ABOVE INST.

7109
7110
7111
7112
7113
7114
7115
7116
7117
7118
7119
7120

; THIS TEST VERIFIES MODE 2 SINGLE-OPERAND INSTRUCTIONS. PREVIOUSLY
; TESTED INSTRUCTIONS ARE USED TO SET A POINTER IN RO TO LOC. 400.
; LOC. 400 IS INITIALIZED TO -1 BEFORE A CLR MODE 2 IS EXECUTED.
; THEN RO IS DECREMENTED BY TWO TO AGAIN POINT TO 400 BEFORE EACH
; OF SEVERAL MODE 2 INSTRUCTIONS ARE USED TO VERIFY THE DATA RESULTS OF
; THE TEST. THIS PROCEDURE ALSO VERIFIES THE PROPER INCREMENTING OF THE
; REGISTER.

(2) 004646
(3) 004646 005000
7121 004646 005000
7122 004650 105100

; TEST 54 TEST MODE 2 USING SOP INST.

TS54:
CLR RO ;SET RO=400
COMB RO

I 3

7123 004652 005200
7124 004654 005010
7125 004656 005110
7126 004660 005020
7127 004662 001401
(2) 004664 104000
7128 004666 005300
7129 004670 005300
7130 004672 005120
7131 004674 100004
7132 004676 005300
7133 004700 005300
7134 004702 005220
7135 004704 001401
(2) 004706
(3) 004706 104000

```
INC      RO
CLR      (RO)      ;CLEAR 400
COM      (RO)      ;INITIALIZE: 400= 1
CLR      (RO)+     ;TRY CLEARING WITH MODE 2
BEQ      SOPZA
EMT
SOPZA: DEC      RO
DEC      RO
COM      (RO)+     ;TRY COMPLEMENTING WITH MODE 2
BPL      SOP28
DEC      RO
DEC      RO
INC      (RO)+     ;TRY INCREMENTING WITH MODE 2
BEQ      T555
SOP28:  EMT
EMT      ;CHECK CUMULATIVE RESULT OF ABOVE INST
```

7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147

```
*****
;
; THIS TEST VERIFIES MODE 2 SINGLE OPERAND INSTRUCTIONS WHICH
; ADDRESS EVEN BYTES. RO IS SET TO 400 AND USED TO INITIALIZE LOCATION
; 400 TO -1. CLRB INSTRUCTION IS THEN EXECUTED ON BYTE 400 WITH
; MODE 2.
; RO IS THEN DECREMENTED BEFORE EACH OF SEVERAL MODE 2 INSTRUCTIONS
; WHICH ARE USED TO VERIFY THE DATA RESULTS OF THE TEST. THIS PROCEDURE ALSO
; VERIFIES THE PROPER INCREMENTING OF THE REGISTER.
;
```

(2) 004710
(3) 004710 005000
7148 004710 005000
7149 004712 105100
7150 004714 005200
7151 004716 005010
7152 004720 005110
7153 004722 105020
7154 004724 001401
(2) 004726 104000
7155 004730 005300
7156 004732 005210
7157 004734 105110
7158 004736 105220
7159 004740 100003
7160 004742 005300
7161 004744 105220
7162 004746 001401
(2) 004750
(3) 004750 104000

```
*****
;TEST 55 TEST MODE 2 EVEN BYTE USING SOP INST.
*****
T555: CLR      RO      ;SET RO=400
COMB    RO
INC     RO
CLR     (RO)      ;CLEAR 400
COM     (RO)      ;INITIALIZE: 400=-1
CLRB   (RO)+     ;TRY TO CLEAT 400 W/MODE 2
BEQ    SOPB2A
EMT
SOPB2A: DEC     RO
INC     (RO)
COMB   (RO)
INCB  (RO)+     ;TRY TO INC EVEN BYTE
BPL   SOPB28
DEC   RO
INCB (RO)+     ;RESET RO=400
BEQ  S56
SOPB28: EMT
EMT    ;TEST CUMULATIVE RESULT OF ABOVE INST.
```

7163
7164
7165
7166
7167
7168
7169

```
*****
;
; THIS TEST FOLLOWS THE SAME PROCEDURE DESCRIBED IN THE PREVIOUS
; TEST. HERE, THE BYTE INSTRUCTION IS USED TO ADDRESS AN ODD BYTE.
;
*****
```

```

(2)
(3)
(2) 004752
7170 004752 005000
7171 004754 105100
7172 004756 005200
7173 004760 005010
7174 004762 005110
7175 004764 005200
7176 004766 105020
7177 004770 001401
(2) 004772 104000
7178 004774 005300
7179 004776 005300
7180 005000 005220
7181 005002 005300
7182 005004 105110
7183 005006 105220
7184 005010 100003
7185 005012 005300
7186 005014 105220
7187 005016 001401
(2) 005020
(3) 005020 104000
7188
7189
7190
7191
7192
7193
7194
(2)
(3)
(2) 005022
7195 005022 005000
7196 005024 005200
7197 005026 005400
7198 005030 100003
7199 005032 001402
7200 005034 102401
7201 005036 103401
(1) 005040
(2) 005040 104000
7202
7203 005042 005200
7204 005044 001401
(2) 005046 104000
7205
7206 005050 105100
7207 005052 105400
7208 005054 100403
7209 005056 001402
7210 005060 102401
7211 005062 103401
(1) 005064
(2) 005064 104000

```

```

;TEST 56 TEST MODE 2 ODD BYTE USING SOP INST.
;*****
TS56:
      CLR      R0          ;SET R0=400
      COMB    R0
      INC     R0
      CLR     (R0)        ;CLEAR LOC 400
      COM     (R0)        ;INITIALIZE: 400= 1
      INC     R0          ;R0=ODD BYTE
      CLRB    (R0)+       ;TRY TO CLEAR ODD BYTE
      BEQ     SOPB2C
      EMT
SOPB2C: DEC     R0          ;CLRB DID NOT SET Z BIT
      DEC     R0          ;R0=WORD ADDR.
      INC     (R0)+       ;INCREMENT WORD
      DEC     R0          ;POINT TO ODD BYTE
      COMB    (R0)        ;COMPLEMENT ODD BYTE
      INCB    (R0)+       ;TRY TO INCREMENT ODD BYTE
      BPL     SOPB2D
      DEC     R0          ;RESET R0 TO ODD BYTE
      INCB    (R0)+       ;TRY TO INCREMENT ODD BYTE
      BEQ     TS57
SOPB2D: EMT              ;TEST CUMULATIVE RESULT OF ABOVE INST.
;*****
;
; THESE TESTS CHECK THE NEGATE INSTRUCTION IN ALL MODES. PREVIOUSLY
; TESTED SINGLE-OPERAND INSTRUCTIONS ARE USED TO TEST THE NEGATE INSTRUCTION.
;*****
;TEST 57 TEST MODE 0 USING NEGATE INSTRUCTION
;*****
TS57:
      CLR      R0          ;SET R0=0
      INC     R0          ; R0=1
      NEG     R0          ;TRY NEGATE MODE 0: R0= 1
      BPL     NEG00       ;CC=1001?
      BEQ     NEG00
      BVS     NEG00
      BCS     NEG01
NEG00: EMT              ;NEGATE DID NOT SET CC'S CORRECTLY
NEG01: INC     R0          ;TEST DATA RESULT
      BEQ     NEG02
      EMT              ;DATA RESULT OF NEGATE INCORRECT
NEG02: COMB    R0          ;R0=377
      NEGB   R0          ;R0=1
      BMI     NEG03       ;CC=0001?
      BEQ     NEG03
      BVS     NEG03
      BCS     NEG04
NEG03: EMT              ;NEGB DID NOT SET CC'S CORRECTLY

```

103

7212 005066 005300
7213 005070 001401
(3) 005072 104000
7214
(2)
(3)
(2) 005074
7215 005074 005000
7216 005076 005010
7217 005100 005210
7218 005102 005410
7219 005104 100003
7220 005106 001402
7221 005110 102401
7222 005112 103401
(1) 005114
(2) 005114 104000
7223
7224 005116 005237 000000
7225 005122 001401
(2) 005124 104000
7226 005126 105110
7227 005130 105410
7228 005132 100403
7229 005134 001402
7230 005136 102401
7231 005140 103401
(1) 005142
(2) 005142 104000
7232 005144 005337 000000
7233 005150 001401
(3) 005152 104000
7234
(2)
(3)
(2) 005154
7235 005154 005000
7236 005156 005010
7237 005160 005210
7238 005162 005420
7239 005164 100003
7240 005166 001402
7241 005170 102401
7242 005172 103401
(1) 005174
(2) 005174 104000
7243 005176 105300
7244 005200 105300
7245 005202 105420
7246 005204 105420
7247 005206 105340
7248 005210 005300
7249 005212 001401
(2) 005214 104000
7250 005216 005337 000000
7251 005222 001401

```
NEG04: DEC R0 ;TEST DATA RESULT
        BEQ TS60 ;DATA RESULT OF NEGB INCORRECT
        EMT
;*****
;TEST 60 TEST MODE 1 USING NEGATE INST.
;*****
TS60:
        CLR R0 ;POINT TO LOC. 0
        CLR (R0) ;CLEAR LOC. 0
        INC (R0) ;LOC. 0=1
        NEG (R0) ;TRY NEG. LOC. 0= 1
        BPL NEG10 ;CC=1001
        BEQ NEG10
        BVS NEG10
        BCS NEG11
NEG10:
        EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG11: INC @#0 ;TEST DATA RESULT
        BEQ NEG12 ;DATA RESULT OF NEGATE INCORRECT
        EMT
NEG12: COMB (R0) ;LOC. 0=377
        NEGB (R0) ;TRY NEGB LOC. 0=1
        BMT NEG13 ;CC=0001?
        BEQ NEG13
        BVS NEG13
        BCS NEG14
NEG13:
        EMT ;NEGB DID NOT SET CC'S CORRECTLY
NEG14: DEC @#0 ;TEST DATA RESULT
        BEQ TS61 ;DATA RESULT OF NEGB INCORRECT
        EMT
;*****
;TEST 61 TEST MODE 2 USING NEGATE INSTRUCTION
;*****
TS61:
        CLR R0 ;POINT TO LOC. 0
        CLR (R0) ;CLEAR LOC. 0
        INC (R0) ;LOC. 0=1
        NEG (R0)+ ;TRY NEG.: LOC. 0= 1
        BPL NEG20 ;CC=1001?
        BEQ NEG20
        BVS NEG20
        BCS NEG21
NEG20:
        EMT ;NEGATE DID NOT SET CC'S CORRECTLY
NEG21: DECB R0 ;R0=LOC. 0
        DECB R0
        NEGB (R0)+ ;BYTE 0=1 R0=1
        NEGB (R0)+ ;BYTE 1=1 R0=2
        DECB -(R0) ;R0=1 LOC. 0=01
        DEC R0 ;R0=0
        BEQ NEG22
        EMT ;REGISTER NOT INCREMENTED CORRECTLY
NEG22: DEC @#0 ;LOC. 0=0
        BEQ TS62
```

13

EMT ;NEG BYTE INSTRUCTIONS FAILED

(3) 005224 104000
7252
7253
7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265
7266
7267

; THIS TEST VERIFIES MODE 3 SINGLE OPERAND INSTRUCTIONS. IT
; USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 400
; THRU 402 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
; INSTRUCTIONS UNDER TEST.
; R0 IS SET TO 400, THE START OF THE ADDRESS TABLE, AND A CLR
; INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR LOC. 0. THEN R0
; IS DECREMENTED BY TWO AND TWO OTHER MODE 3 INSTRUCTIONS OPERATE ON
; LOC. 0 TO VERIFY THE DATA RESULTS OF THE TEST. THE PROPER INCREMENTING
; OF THE REGISTER IS ALSO VERIFIED IN THIS MANNER.
; IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE
; (LOC. 400-402) HAS THE PROPER VALUES (0).

TEST 62 TEST MODE 3 USING SOP INST.

TS62:

CLR R0 ;SET R0=400
COMB R0
INC R0
CLR (R0) ;CLEAR LOC 400
CLR B(R0) ;TRY TO CLEAR LOC 0 USING MODE 3 ;R0=402
BEQ SOP3A
EMT ;CLR DID NOT SET Z-BIT
SOP3A: DEC R0 ;RESET R0=400
DEC R0
COM B(R0) ;TRY TO COMPLEMENT LOC 0 OF MODE 3 ;R0=402
BPL SOP38
INC B(R0) ;TRY TO INCREMENT LOC 0 W/MODE 3 ;R0=404
BEQ TS63

SOP38: EMT ;CUMULATIVE RESULT OF ABOVE INST FAILED

; THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
; WHICH ADDRESS EVEN BYTES. AGAIN, THE TARGET LOCATION 0 IS USED
; AND THE SAME TABLE AT 400 IS EMPLOYED.
; AFTER POINTING R4 TO THE TABLE (400) AND SETTING LOCATION
; 0 TO -1, A CLRB INSTRUCTION IS USED TO CLEAR BYTE 0.
; SEVERAL OTHER MODE 3 INSTRUCTIONS ARE THEN USED WITH THE TABLE
; TO VERIFY THE DATA RESULTS AND THE PROPER INCREMENTING OF THE REGISTER.
; IF A FAILURE IS DETECTED, BE SURE THAT THE TABLE (LOCATION 400-402) HAS
; THE PROPER VALUES (0).

TEST 63 TEST MODE 3 EVEN BYTE USING SOP INST.

TS63:

CLR R4 ;SET R4=400
COMB R4
INC R4
CLR R0 ;INITIALIZE LOC. 0= 1

(2) 005226
(3)
(2) 005226 005000
7268 005226 105100
7269 005230 005200
7270 005232 005010
7271 005234 005030
7272 005240 001401
(2) 005242 104000
7274 005244 005300
7275 005246 005130
7276 005250 100002
7277 005252 005230
7278 005254 001401
(2) 005260
(3) 005260 104000
7280
7281
7282
7283
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
(2) 005262
(3)
(2) 005262 005004
7294 005264 105104
7295 005266 005204
7296 005270 005000

7298 005272 005010
 7299 005274 005110
 7300 005276 105034
 7301 005300 001401
 (2) 005302 104000
 7302 005304 005304
 7303 005306 005304
 7304 005310 005234
 7305 005312 100006
 7306 005314 105434
 7307 005316 100004
 7308 005320 005304
 7309 005322 005304
 7310 005324 105234
 7311 005326 001401
 (2) 005330
 (3) 005330 104000
 7312
 7313
 7314
 7315
 7316
 7317
 7318
 7319
 7320
 7321
 7322
 7323
 7324
 7325

```

          CLR      (R0)
          COM      (R0)          ;LOC. 0= 1
          CLRB    @ (R4)+       ;TRY TO CLEAR EVEN BYTE ;LOC. 0=177400 R4=402
          BEQ     SOPB3A
          EMT
SOPB3A:  DEC      R4            ;CLRB DID NOT SET Z-BIT
          DEC      R4            ;RESET POINTER R4=400
          INC     @ (R4)+       ;TRY INCREMENTING WORD LOC.0=177401 R4=402
          BPL     SOPB38
          NEGB    @ (R4)+       ;TRY TO NEGATE EVEN BYTE ;LOC.0=-1 R4=404
          BPL     SOPB38
          DEC     R4            ;R4=402
          DEC     R4
          INCB    @ (R4)+       ;TRY TO INCREMENT EVEN BYTE ;LOC. 0=17400
          BEQ     TS64
SOPB38:  EMT                    ;CUMMULATIVE RESULT OF ABOVE INST FAILED
  
```

```

;*****
;
;   THIS TEST VERIFIES MODE 3 SINGLE OPERAND BYTE INSTRUCTIONS
; WHICH ADDRESS ODD BYTES. THE TARGET IS BYTE 1. A TABLE AT
; LOC. 400-406 IS USED. R0 SERVES AS THE TABLE POINTER.
; R0 IS INITIALIZED TO 400. LOC. 0 IS SET TO -1 USING THE
; FIRST TWO TABLE ENTRIES. A CLRB MODE 3 IS EXECUTED ON BYTE 1 USING
; TABLE ADDRESS AT 404. R0 IS DECREMENTED TO 402 AND SEVERAL SOP
; MODE 3 INSTRUCTIONS ARE USED TO VERIFY DATA RESULTS AND PROPER
; REGISTER INCREMENTING.
; THE TABLE (400-406) SHOULD CONTAIN 0,0,1,1 BEFORE AND
; AFTER THE TEST IS RUN.
;
  
```


B4

7341 005364 105430
7342 005366 100002
7343 005370 105230
7344 005372 001401
(2) 005374
(3) 005374 104000
7345
(2)
(3)
(2) 005370
7346 005376 005000
7347 005400 105100
7348 005402 005200
7349 005404 005010
7350 005406 005004
7351 005410 005014
7352 005412 005214
7353 005414 005430
7354 005416 100003
7355 005420 001402
7356 005422 102401
7357 005424 103401
(1) 005426
(2) 005426 104000
7358 005430 005214
7359 005432 001401
(2) 005434 104000
7360 005436 105137 000001
7361 005442 005237 000000
7362 005446 105430
7363 005450 100401
(2) 005452 104000
7364 005454 105430
7365 005456 100001
(2) 005460 104000
7366 005462 105137 000001
7367 005466 105237 000001
7368 005472 005214
7369 005474 001401
(3) 005476 104000
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
(2)
(3)
(2) 005500
7380 005500 005000
7381 005502 105100
7382 005504 005200

NEGB B(R0). ;TRY TO NEGATE ODD BYTE LOC. 0=177400 R0=406
BPL SOPB30
INCB B(R0). ;TRY TO INCREMENT ODD BYTE LOC.0=0 R0=410
BEQ TS65

SOPB30: EMT ;CUMULATIVE RESULT OF ABOVE INSTS FAILED

;.....
;TEST 65 TEST MODE 3 USING NEGATE INSTRUCTION
;.....

TS65:
CLR R0 ;R0=400
COMB R0
INC R0
CLR (R0) ;LOC. 400=0
CLR R4 ;R4=0
CLR (R4) ;LOC. 0=0
INC (R4) ;LOC. 0=1
NEG B(R0). ;TRY NEGATE LOC. 0=-1 R0=402
BPL NEG30 ;CC=1001?
BEQ NEG30
BVS NEG30
BCS NEG31

NEG30: EMT ;NEG DID NOT SET CC'S CORRECTLY
NEG31: INC (R4) ;LOC. 0=0
BEQ NEG32

NEG32: COMB B#1 ;DATA RESULT OF NEG INCORRECT
INC B#0 ;LOC 0=177400
NEGB B(R0). ;TRY NEGB LOC. 0=177777 R0=404
BMI NEG33

NEG33: EMT ;NEGB FAILED WITH EVEN BYTE
NEGB B(R0). ;TRY NEGB LOC.0=777 R0=406
BPL NEG34

NEG34: EMT ;NEGB FAILED WITH ODD BYTE
COMB B#1 ;LOC. 0=177377
INCB B#1 ;LOC. 0=177777
INC (R4) ;LOC. 0=0
BEQ TS66

EMT ;DATA RESULT OF NEGB'S INCORRECT

;.....
;

THIS TEST VERIFIES MODE 4 SINGLE OPERAND INSTRUCTIONS.
;R0 IS SET TO 400. A CLR INSTRUCTION IS EXECUTED IN MODE 4 TO CLEAR
;LOC. 376. R0 IS RESET TO 400 AND A COM INSTRUCTION USING MODE 4
;COMPLEMENTS LOC.376.

;TWO INC INSTRUCTIONS AND A MODE 4 INSTRUCTION ARE EXECUTED
;TO COMPLETE THE TEST.

;.....
;TEST 66 TEST MODE 4 USING SOP INSTS
;.....

TS66:
CLR R0 ;SET R0=400
COMB R0
INC R0

```

7383 005506 005040
7384 005510 001401
(2) 005512 104000
7385 005514 005200
7386 005516 005200
7387 005520 005140
7388 005522 100004
7389 005524 005200
7390 005526 005200
7391 005530 005240
7392 005532 001401
(2) 005534
(3) 005534 104000
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
(2)
(3)
(2) 005536
7410 005536 012700 000370
7411 005542 005020
7412 005544 005020
7413 005546 005020
7414 005550 005010
7415 005552 005000
7416 005554 005020
7417 005556 105400
7418 005560 005050
7419 005562 001401
(2) 005564 104000
7420 005566 005200
7421 005570 005200
7422 005572 005150
7423 005574 100002
7424 005576 005250
7425 005600 001401
(2) 005602
(3) 005602 104000
7426
7427
7428
7429

```

```

CLR -(RO) ;TRY TO CLEAR USING MODE 4
BEQ SOP4A
EMT ;CLR DID NOT SET Z BIT
SOP4A: INC RO ;RESET RO
INC RO
COM -(RO) ;TRY TO COMPLEMENT USING MODE 4
BPL SOP4B
INC RO ;MOVE POINTER
INC RO
INC -(RO)
BEQ TS67
SOP4B: EMT ;CHECK CUMULATIVE RESULT OF ABOVE INST.

```

```

;*****
;
; THIS TEST VERIFIES MODE 5 SINGLE OPERAND INSTRUCTIONS. IT
; USES LOCATION 0 AS ITS TARGET DATA. A TABLE LOCATED AT LOC. 372
; THRU 374 IS USED TO SUPPLY THE ADDRESS OF LOCATION 0 TO THE
; INSTRUCTIONS UNDER TEST.
; RO IS SET TO 376, (THE START OF THE ADDRESS TABLE) +2,
; AND A CLR INSTRUCTION IS EXECUTED WITH MODE 3 TO CLEAR
; LOC. 0. THEN RO IS INCREMENTED BY TWO AND TWO OTHER MODE 3
; INSTRUCTIONS OPERATE ON LOC. 0 TO VERIFY THE DATA RESULTS OF
; THE TEST. THE PROPER DECREMENTING OF THE REGISTER IS ALSO
; VERIFIED IN THIS MANNER.
; IF A FAILURE IS DETECTED BE SURE TO VERIFY THAT THE TABLE
; (LOC. 372 THRU 374) HAS THE PROPER VALUES (0).
;*****
;TEST 67 TEST MODE 5 USING SOP INSTS
;*****
TS67:

```

```

MOV #370,RO ;CLEAR LOCATION 370-376
CLR (RO). ;370
CLR (RO). ;372
CLR (RO). ;374
CLR (RO) ;376
CLR RO ;SET RO=376 (LOW BYTE)
CLR (RO).
NEGB RO
CLR B-(RO) ;TRY TO CLEAR LOC 0 W/MODE 5
BEQ SOP5A
EMT ;CLR DID NOT SET Z-BIT
SOP5A: INC RO ;RESET RO
INC RO
COM B-(RO) ;TRY TO COMPLEMENT LOC. 0 W/MODE 5
BPL SOP5B
INC B-(RO) ;TRY TO INCREMENT LOC. 0 W/MODE 5
BEQ TS70
SOP5B: EMT ;TEST CUMULATIVE RESULT OF ABOVE INSTS

```

```

;*****
;
; THIS TEST VERIFIES MODE 6 SINGLE OPERAND INSTRUCTIONS. IT

```

7430
7431
7432
7433
7434
7435
(2)
(3)
(2) 005604
7436 005604 005000
7437 005606 105100
7438 005610 005200
7439 005612 005060 177400
7440 005616 001401
(2) 005620 104000
7441 005622 005160 177400
7442 005626 100003
7443 005630 005260 177400
7444 005634 001401
(2) 005636
(3) 005636 104000
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
(2)
(3)
(2) 005640
7456 005640 005000
7457 005642 105100
7458 005644 005200
7459 005646 005210
7460 005650 005070 000002
7461 005654 001401
(2) 005656 104000
7462 005660 005170 000002
7463 005664 100003
7464 005666 005270 000002
7465 005672 001401
(2) 005674
(3) 005674 104000
7466
7467
(2)
(3)
(2) 005676
7468 005676 005000
7469 005700 005010
7470 005702 005120

;USES LOCATION 0 AS ITS TARGET DATA. R0 IS SET TO 400 USING
;PREVIOUSLY TESTED INSTRUCTIONS AND A MODE 6 CLF INSTRUCTION IS
;EXECUTED ON LOC. 0 USING R0 AND A -400 OFFSET. COM AND INC
;INSTRUCTIONS ARE THEN USED TO VERIFY THE DATA.
;.....
;TEST 70 TEST MODE 6 USING SOP INSTS
;.....
TS70:
CLR R0 ;SET R0=400
COMB R0
INC R0
CLR -400(R0) ;TRY TO CLEAR LOCATION 0 W/MODE 6
BEQ SOP6A
EMT ;CLR DID NOT SET Z-BIT
SOP6A: COM -400(R0) ;TRY TO COMPLEMENT LOCATION 0 W/MODE 6
BPL SOP6B
INC -400(R0) ;TRY TO INCREMENT LOCATION 0 W/MODE 6
BEQ TS71
SOP6B: EMT ;TEST CUMULATIVE RESULT OF ABOVE INSTS
;.....
; THIS TEST VERIFIES MODE 7 SINGLE OPERAND INSTRUCTIONS. IT USES
;THE POINTER TO LOC. 0 WHICH IS STORED AT LOC. 402.
; R0 IS SET TO 400 AND A MODE 7 CLR INSTRUCTION IS
;EXECUTED WITH A +2 OFFSET TO CLEAR LOC. 0.
; SEVERAL OTHER MODE 7 INSTRUCTIONS ARE THEN USED ON THE COMMON
;LOCATION TO VERIFY THE DATA RESULTS.
;.....
;TEST 71 TEST MODE 7 USING SOP INST.
;.....
TS71:
CLR R0 ;SET R0=400
COMB R0
INC R0
INC (R0) ;R0=1
CLR B2(R0) ;TRY TO CLEAR LOC. 0 W/MODE 7
BEQ SOP7A
EMT ;CLR DID NOT SET Z-BIT
SOP7A: COM B2(R0) ;TRY TO COMPLEMENT LOC. 0 W/MODE 7
BPL SOP7B
INC B2(R0) ;TRY TO INCREMENT LOC. 0 W/MODE 7
BEQ TS72
SOP7B: EMT ;TEST CUMULATIVE RESULT OF ABOVE INSTS.
;.....
;TEST 72 TEST MODE 4 WITH NEGATE INSTRUCTION
;.....
TS72:
CLR R0
CLR (R0)
COM (R0). ;LOC. 0=177777, R0=2

7471 005704 005440
 7472 005706 100403
 7473 005710 001402
 7474 005712 102401
 7475 005714 103401
 (1) 005716
 (2) 005716 104000
 7476 005720 005400
 7477 005722 001401
 (2) 005724 104000
 7478 005726 005310
 7479 005730 001401
 (3) 005732 104000
 7480
 (1)
 (3)
 (2) 005734
 7481 005734 005000
 7482 005736 005010
 7483 005740 105100
 7484 005742 005200
 7485 005744 005010
 7486 005746 005004
 7487 005750 005314
 7488 005752 005450
 7489 005754 100403
 7490 005756 001402
 7491 005760 102401
 7492 005762 103401
 (1) 005764
 (2) 005764 104000
 7493 005766 005314
 7494 005770 001401
 (2) 005772 104000
 7495 005774 105100
 7496 005776 005300
 7497 006000 001401
 (3) 006002 104000
 7498
 (2)
 (3)
 (2) 006004
 7499 006004 005000
 7500 006006 005004
 7501 006010 105100
 7502 006012 005014
 7503 006014 105024
 7504 006016 105114
 7505 006020 005460 177401
 7506 006024 100403
 7507 006026 001402
 7508 006030 102401
 7509 006032 103401
 (1) 006034
 (2) 006034 104000
 7510 006036 105314

```

NEG      -(R0)      ;TRY NEGATE, LOC. 0=1
BMI      NEG40      ;CC=0001?
BEQ      NEG40
BVS      NEG40
BCS      NEG41

NEG40:   EMT
NEG41:   NEG      R0      ;NEG DID NOT SET CC'S CORRECTLY
        BEQ      NEG42   ;TST R0 WITH A NEG.
        EMT
NEG42:   DEC      (R0)   ;R0 NOT DECREMENTED PROPERLY
        BEQ      TS73   ;TEST DTA RESULT OF NEG
        EMT
                          ;DATA RESULT OF NEG INCORRECT
;*****
;TEST 73      TEST MODE 5 WITH NEGATE INSTRUCTION
;*****
TS73:
CLR      R0          ;R0=0
CLR      (R0)       ;LOC. 0=0
COMB    R0          ;R0=377
INC     R0          ;R0=400
CLR      (R0)       ;SET 400 = 0
CLR     R4          ;R4=0
DEC     (R4)        ;LOC. 0=177777
NEG     @-(R0)      ;TRY NEGATE: LOC. 0=1
BMI     NEG50       ;CC=0001?
BEQ     NEG50
BVS     NEG50
BCS     NEG51

NEG50:   EMT
NEG51:   DEC      (R4)   ;NEG DID NOT SET CC'S CORRECTLY
        BEQ      NEG52
        EMT
NEG52:   COMB    R0
        DEC     R0
        BEQ     TS74
        EMT
                          ;REGISTER NOT DECREMENTED PROPERLY
;*****
;TEST 74      TEST MODE 6 WITH NEGATE
;*****
TS74:
CLR      R0          ;R0=0
CLR     R4          ;R4=0
COMB    R0          ;R0=377
CLR     (R4)        ;LOC. 0=0
CLRB   (R4)+       ;LOC. 0=177777, R4=1
COMB   (R4)        ;LOC. 0=177400
NEG     -377(R0)   ;LOC. 0=400
BMI     NEG60       ;CC=0001
BEQ     NEG60
BVS     NEG60
BCS     NEG61

NEG60:   EMT
NEG61:   DECB   (R4)   ;NEG DID NOT SET CC'S CORRECTLY

```

```

7511 006040 001401
(3) 006042 104000
7512
(2)
(3)
(2) 006044
7513 006044 005000
7514 006046 005010
7515 006050 005110
7516 006052 105100
7517 006054 105470 000005
7518 006060 100403
7519 006062 001402
7520 006064 102401
7521 006066 103401
(1) 006070
(2) 006070 104000
7522 006072 105100
7523 006074 105120
7524 006076 105310
7525 006100 005467 171674
7526 006104 001401
(3) 006106 104000
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
(2)
(3)
(2) 006110
7537 006110 005027
7538 006112 177777
7539 006114 001401
(2) 006116 104000
7540 006120 005237 006112
7541 006124 005467 177762
7542 006130 100003
7543 006132 005277 000004
7544 006136 001402
(2) 006140
(3) 006140 104000
7545 006142 006112
7546
7547
7548
7549
7550
7551
7552
7553

```

```

      BEQ      TS75
      EMT
;DATA RESULT OF NEG INCORRECT
;*****
;TEST 75      TEST MODE 7 W/ NEGATE
;*****
TS75:
      CLR      R0          ;R0=0
      CLR      (R0)       ;LOC. 0=0
      COM      (R0)       ;LOC. 0=177777
      COMB     R0          ;R0=377
      NEGB     @S(R0)     ;R0=5=404, 404=1, LOC. 0=777
      BMI      NEG70      ;CC=0001?
      BEQ      NEG70
      BVS      NEG70
      BCS      NEG71

NEG70:
      EMT
NEG71:
      COMB     R0          ;NEG DID NOT SET CC'S CORRECTLY
      COMB     (R0)+      ;R0=0
      DECB    (R0)       ;LOC. 0=400, R0=1
      NEG      0          ;LOC. 0=0
      BEQ      TS76      ;USE NEG MODE 67 TO TST FOR ZERO
      EMT
;DATA RESULT OF NEG WAS INCORRECT
;*****
;
;      THIS TEST VERIFIES PROGRAM COUNTER ADDRESSING WITH SOP
;INSTRUCTIONS. CLR MODE 77 IS USED TO CLEAR THE LOCATION FOLLOWING THE
;INSTRUCTION (SOPX). THEN SINGLE OPERAND INSTRUCTIONS WITH MODES 37, 67, AND
;77, USING INDIRECT POINTER SOPXAD ARE USED TO VERIFY THE DATA RESULTS
;OF THESE INSTRUCTIONS.
;
;*****
;TEST 76      TEST SOP INSTRUCTIONS MODES 2,3,6,7 WITH REGISTER 7
;*****
TS76:
      CLR      (R7)+      ;CLEAR NEXT LOCATION: (SOPX)
SOPX:
      -1
      USE MODE 27
      BEQ      SOPA
      EMT
SOPA:
      INC      @SOPX      ;CLR DID NOT SET Z-BIT
      NEG     SOPX        ;INC SOPX W/MODE 37
      BPL     SOPB        ;NEGATE SOPX W/MODE 67
      INC     @SOPXAD     ;INC SOPX W/MODE 77
      BEQ     TS77
SOPB:
      EMT
SOPXAD:
      SOPX
      ;INC DID NOT SET Z-BIT
      ;INDIRECT ADDRESS OF SOPX
;*****
;
;      THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING INSTRUCTIONS
;USING MODE 0. R0 IS SET TO ZERO AND THE CONDITION CODES ARE SET
;TO THE COMPLEMENT OF THAT EXPECTED BY THE INSTRUCTION. A TST INSTRUCTION
;IS EXECUTED AND CONDITIONAL BRANCHES ARE USED TO TEST THE CONDITION
;CODES.

```

7554
7555
(2)
(3)
(2) 006144
7556 006144 005000
7557 006146 000277
7558 006150 000244
7559 006152 005700
7560 006154 102403
7561 006156 100402
7562 006160 103401
7563 006162 001401
(2) 006164
(3) 006164 104000
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
(2)
(3)
(2) 006166
7574 006166 005000
7575 006170 105100
7576 006172 000277
7577 006174 000250
7578 006176 105700
7579 006200 102402
7580 006202 101401
7581 006204 100401
(2) 006206
(3) 006206 104000
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
(2)
(3)
(2) 006210
7592 006210 005000
7593 006212 005010
7594 006214 000277
7595 006216 000244
7596 006220 005710

```
;  
;.....  
;TEST 77 TEST MODE 0 SOP NON-MODIFYING  
;.....  
TS77:  
    CLR      RO          ;INITIALIZE RO=0  
    SCC          ;SET CC=1011  
    CLZ  
    TST      RO          ;TRY TST W/ MODE 0  
    BVS      SNMOA       ;CHECK THAT CC=0100  
    BMI      SNMOA  
    BCS      SNMOA  
    BEQ      TS100  
  
SNMOA:  
    EMT          ;CONDITION CODES NOT SET PROPERLY  
  
;.....  
;  
; THIS TEST VERIFIES SINGLE OPERAND NON-MODIFYING BYTE INSTRUCTIONS WITH MODE 0.  
;RO IS SET TO 377 AND COMPLEMENT OF THE EXPECTED CONDITION CODES  
;IS LOADED IN PSW. A TSTB INSTRUCTION IS EXECUTED AND THE RESULTS  
;ARE CHECKED WITH SEVERAL CONDITIONAL BRANCH INSTRUCTIONS.  
; THIS VERIFIES THAT THE PROPER BYTE WAS TESTED.  
;  
;.....  
;TEST 100 TEST MODE 0 EVEN BYTE W/ SOP NON-MODIFYING  
;.....  
TS100:  
    CLR      RO          ;INITIALIZE  
    COMB     RO          ;RO=377  
    SCC          ;SET CC=0111  
    CLN  
    TSTB     RO          ;TRY TST EVEN BYTE  
    BVS      SNMBOA     ;CHECK CC=1000  
    BLOS     SNMBOA  
    BMI      TS101  
  
SNMBOA:  
    EMT          ;CONDITION CODES NOT SET PROPERLY  
  
;.....  
;  
; THIS TEST VERIFIES SINGLE OPERAND INSTRUCTIONS WITH MODE 1.  
;RO IS USED TO POINT TO AND CLEAR LOC. 0. THE COMPLEMENT OF THE  
;EXPECTED CONDITION CODES ARE LOADED IN THE PSW. A TST INSTRUCTION  
;IS THEN EXECUTED ON LOC. 0 USING RO AND CONDITIONAL BRANCHES TEST  
;THE RESULTS.  
;  
;.....  
;TEST 101 TEST MODE 1 SOP NON-MODIFYING  
;.....  
TS101:  
    CLR      RO          ;POINT TO LOC 0  
    CLR      (RO)       ;CLEAR LOC 0  
    SCC          ;INITIALIZE  
    CLZ          ;CC=1011  
    TST      (RO)       ;TRY TST W/ MODE 1
```


7597 006222 102403
7598 006224 103402
7599 006226 100401
7600 006230 001401
(2) 006232
(3) 006232 104000
7601
7602
7603
7604
7605
7606
7607
7608
7609
(2)
(3)
(2) 006234
7610 006234 005000
7611 006236 005010
7612 006240 105110
7613 006242 000277
7614 006244 000250
7615 006246 105710
7616 006250 102402
7617 006252 101401
7618 006254 100401
(1) 006256
(2) 006256 104000
7619 006260 005000
7620 006262 005200
7621 006264 000277
7622 006266 000244
7623 006270 105710
7624 006272 102403
7625 006274 103402
7626 006276 100401
7627 006300 001401
(2) 006302
(3) 006302 104000
7628
7629
7630
7631
7632
7633
7634
7635
7636
(2)
(3)
(2) 006304
7637 006304 005000
7638 006306 005010
7639 006310 000277
7640 006312 000244

BVS SNM1A ;CHECK CC=0100
BCS SNM1A
BMI SNM1A
BEQ TS102
SNM1A:
EMT ;CC'S NOT SET PROPERLY

; THIS TEST SETS LOCATION 0 TO 377 AND THEN USES R0 TO TEST
; THE EVEN BYTE AND THE ODD BYTE USING SOP BYTE INSTRUCTIONS WITH MODE 1.
; AGAIN, CONDITIONAL BRANCHES ARE USED TO VERIFY THE SETTING OF THE
; PROPER CONDITION CODE BITS.

; TEST 102 TEST MODE 1 BYTE INST. NON-MODIFYING

TS102:
CLR R0 ;POINT TO LOC 0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;COMPLEMENT BYTE 0
SCC ;SET CC=0111
CLN
TSTB (R0) ;TRY TST ON EVEN BYTE
BVS SNMB1A
BLOS SNMB1A
BMI SNMB1B

SNMB1A:
EMT ;CC'S NOT CORRECT

SNMB1B: CLR R0
INC R0
SCC ;SET CC=1011
CLZ
TSTB (R0) ;TRY TO TST AN ODD BYTE
BVS SNMB1C ;CHECK CC=0100
BCS SNMB1C
BMI SNMB1C
BEQ TS103

SNMB1C:
EMT ;CC'S NOT CORRECT

; THIS TEST VERIFIES THE SINGLE-OPERAND NON-MODIFYING INSTRUCTIONS
; USING MODE 2. IT USES THE IDENTICAL PROCEDURE EMPLOYED IN THE
; MODE 1 TESTS. ADDITIONALLY, THE REGISTER IS CHECKED TO ASSURE THAT
; IT IS INCREMENTED PROPERLY.

; TEST 103 TEST MODE 2 WITH SOP NON-MODIFYING

TS103:
CLR R0 ;INITIALIZE R0=0
CLR (R0) ;CLEAR LOC 0
SCC ;SET CC=1011
CLZ

7641	006314	005720	TST	(R0)+	;TRY TST W/ MODE 2
7642	006316	102403	BVS	SNM2A	;CHECK CC=0100
7643	006320	103402	BCS	SNM2A	
7644	006322	100401	BMI	SNM2A	
7645	006324	001401	BEQ	SNM2B	
(1)	006326		SNM2A:		
(2)	006326	104000	EMT		;CC'S NOT CORRECT
7646	006330	005300	SNM2B:	DEC R0	;RESET R0
7647	006332	005300	DEC	R0	
7648	006334	001401	BEQ	TS104	
(3)	006336	104000	EMT		;MODE 2 DID NOT INC REG CORRECTLY

; THIS TEST VERIFIES MODE 2 SINGLE OPERAND NON-MODIFYING BYTE
; INSTRUCTIONS IT USES R0 TO POINT TO LOC. 0. WITH LOCATION 0
; SET TO 377, THE EVEN AND ODD BYTE IS TESTED WITH TSTB INSTRUCTIONS
; TO VERIFY THE CORRECT CC ARE SET. THE REGISTER IS CHECKED FOR
; PROPER INCREMENTING.

; TEST 104 TEST MODE 2 - BYTE W/ SOP NON-MODIFYING

TS104:

7659	006340	005000	CLR	R0	;CLEAR R0
7660	006342	005010	CLR	(R0)	;CLEAR LOC 0
7661	006344	105110	COMB	(R0)	;SET LOC 0=377
7662	006346	000277	SCC		;SET CC=0111
7663	006350	000250	CLN		
7664	006352	105720	TSTB	(R0)+	;TRY TST OF EVEN BYTE
7665	006354	102402	BVS	SNMB2A	
7666	006356	101401	BLOS	SNMB2A	
7667	006360	100401	BMI	SNMB2B	
(1)	006362		SNMB2A:		
(2)	006362	104000	EMT		;CC'S NOT SET CORRECTLY
7668	006364	005300	SNMB2B:	DEC R0	;DECREMENT R0
7669	006366	001401	BEQ	SNMB2C	
(2)	006370	104000	EMT		;MODE 2 DID NOT INC REG CORRECTLY
7670	006372	005200	SNMB2C:	INC R0	;POINT TO ODD BYTE
7671	006374	000277	SCC		;SET CC=1011
7672	006376	000244	CLZ		
7673	006400	105720	TSTB	(R0)+	;TRY TST OF ODD BYTE
7674	006402	102403	BVS	SNMB2D	;CHECK CC'S=0100
7675	006404	103402	BCS	SNMB2D	
7676	006406	100401	BMI	SNMB2D	
7677	006410	001401	BEQ	SNMB2E	
(1)	006412		SNMB2D:		
(2)	006412	104000	EMT		;CC'S NOT CORRECT
7678	006414	005300	SNMB2E:	DEC R0	
7679	006416	005300	DEC	R0	
7680	006420	001401	BEQ	TS105	
(3)	006422	104000	EMT		;R0 DID NOT INCREMENT PROPERLY

; THIS TEST VERIFIES MODE 3 SINGLE OPERAND NON-MODIFYING INSTRUCTIONS.

7681
7682
7683
7684

7685
7686
7687
7688
7689
(2)
(3)
(2) 006424
7690 006424 005000
7691 006426 005010
7692 006430 105100
7693 006432 005300
7694 006434 000277
7695 006436 000244
7696 006440 005730
7697 006442 102403
7698 006444 103402
7699 006446 100401
7700 006450 001401
(1) 006452
(2) 006452 104000
7701 006454 005300
7702 006456 105100
7703 006460 001401
(3) 006462 104000
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
(2)
(3)
(2) 006464
7715 006464 005000
7716 006466 005010
7717 006470 105110
7718 006472 105100
7719 006474 005200
7720 006476 005720
7721 006500 000277
7722 006502 000250
7723 006504 105730
7724 006506 102402
7725 006510 101401
7726 006512 100401
(1) 006514
(2) 006514 104000
7727 006516 000277
7728 006520 000244
7729 006522 105730

;A POINTER IN A TABLE AT LOC. 376 IS USED TO TEST LOCATION 0.
;THE CC'S AND THE REGISTER ARE CHECKED FOLLOWING THE
;TST MODE 3 INSTRUCTION.

;TEST 105 TEST MODE 3 W/ SOP NON-MODIFYING INSTS

TS105:
CLR R0 ;RO=0
CLR (R0) ;CLEAR LOC 0
COMB R0 ;RO=376
DEC R0
SCC ;SET CC=1011
CLZ
TST @ (R0) ;TRY TST W/ MODE 3
BVS SNM3A ;CHECK CC=0100
BCS SNM3A
BMI SNM3A
BEQ SNM3B
SNM3A: EMT ;CC'S NOT CORRECT
SNM3B: DEC R0 ;RO=377
COMB R0 ;RO=0
BEQ TS105
EMT ;MODE 3 DID NOT INC REG CORRECTLY

; THIS TEST VERIFIES SOP NON-MODIFYING BYTE INSTRUCTIONS MODE 3
; LOC. 0 IS SET TO 377. TABLE AT LOC. 402-404 IS USED TO TEST
; BYTE 0 AND BYTE 1. THE REGISTER IS CHECKED FOR PROPER INCREMENTING AND
; THE CC'S ARE VERIFIED.
; THE TABLE AT LOC. 402-404 SHOULD CONTAIN 0 AND 1 BEFORE AND
; AFTER THE TEST IS RUN.

;TEST 106 TEST MODE 3 - BYTES W/ SOP NON-MODIFYING INSTS.

TS106:
CLR R0 ;RO=0
CLR (R0) ;CLEAR LOC 0
COMB (R0) ;LOC. 0 =377
COMB R0
INC R0
TST (R0) ;RO=402
SCC ;CC=0111
CLN
TSTB @ (R0) ;TRY TST OF EVEN BYTE
BVS SNM3A ;CHECK CC=1000
BLOS SNM3A
BMI SNM3B
SNM3A: EMT ;CC'S NOT CORRECT
SNM3B: SCC ;SET CC=1011
CLZ
TSTB @ (R0) ;TRY TST OF ODD BYTE

7730 006524 102403
7731 006526 103402
7732 006530 100401
7733 006532 001401
 (1) 006534
 (2) 006534 104000
7734 006536 005720
7735 006540 005710
7736 006542 100401
 (3) 006544 104000
7737
7738
7739
7740
7741
7742
7743
7744
7745
 (2)
 (3)
 (2) 006546
7746 006546 005000
7747 006550 005010
7748 006552 005120
7749 006554 000277
7750 006556 000244
7751 006560 005740
7752 006562 102402
7753 006564 101401
7754 006566 100401
 (1) 006570
 (2) 006570 104000
7755 006572 005700
7756 006574 001401
 (3) 006576 104000
7757
7758
7759
7760
7761
7762
7763
7764
7765
 (2)
 (3)
 (2) 006600
7766 006600 005000
7767 006602 005010
7768 006604 005110
7769 006606 105100
7770 006610 005200
7771 006612 000277
7772 006614 000250
7773 006616 005750

```
      BVS      SNMB3C      ;CHECK CC=0100
      BCS      SNMB3C
      BMI      SNMB3C
      BEQ      SNMB3D
SNMB3C:  EMT                      ;CC'S NOT CORRECT
SNMB3D:  TST      (RO)+        ;RO=410
          TST      (RO)
          BMI      TS107
          EMT                      ;TSTB DID NOT INCREMENT RO CORRECTLY
;*****
;
;      THIS TEST VERIFIES MODE 4 SOP NON-MODIFYING INSTRUCTIONS.
;LOC. 0 IS SET TO -1 AND THE CC'S ARE SET TO THE COMPLEMENT OF THE
;EXPECTED RESULTS.  RO AND SET TO 2 AND A TST MODE 4 IS EXECUTED.
;THE CC'S ARE CHECKED WITH CONDITIONAL BRANCH INSTRUCTIONS AND THE REGISTER
;IS CHECKED FOR PROPER DECREMENTING.
;*****
;TEST 107      TEST MODE 4 W/ SOP NON-MODIFYING INSTS
;*****
TS107:
      CLR      RO      ;RO=0
      CLR      (RO)   ;LOC 0=0
      COM      (RO)+  ;LOC 0=-1
      SCC                      ;SET CC=1011
      CLZ
      TST      -(RO)  ;TRY TST W/ MODE 4
      BVS      SNM4A  ;CHECK CC=0100
      BLOS     SNM4A
      BMI      SNM4B
SNM4A:  EMT                      ;CC'S NOT CORRECT
SNM4B:  TST      RO
          BEQ      TS110
          EMT                      ;TST MODE 4 DID NOT DEC RO CORRECTLY
;*****
;
;      THIS TEST VERIFIES MODE 5 SOP NON-MODIFYING INSTRUCTIONS.
;IT USES A POINTER AT LOC. 376 TO TEST LOC. 0.  RO IS SET
;TO 400, A TST MODE 5 INSTRUCTION IS EXECUTED AND THE CC'S CHECKED.
;RO IS CHECKED TO INSURE PROPER DECREMENTING.
;*****
;TEST 110      TEST MODE 5 W/ SOP NON-MODIFYING INSTS
;*****
TS110:
      CLR      RO      ;RO=0
      CLR      (RO)   ;LOC 0=0
      COM      (RO)   ;LOC 0=-1
      RO      ;RO=377
      INC      RO     ;RO=400
      SCC                      ;SET CC=0111
      CLN
      TST      @-(RO) ;TRY TST W/ MODE 5
```

7774 006620 102402
 7775 006622 101401
 7776 006624 100401
 (1) 006626
 (2) 006626 104000
 7777 006630 005200
 7778 006632 105100
 7779 006634 001401
 (3) 006636 104000
 7780
 7781
 7782
 7783
 7784
 7785
 7786
 7787
 7788
 (2)
 (3)
 (2) 006640
 7789 006640 005000
 7790 006642 005010
 7791 006644 005110
 7792 006646 105100
 7793 006650 000277
 7794 006652 000250
 7795 006654 005760 177401
 7796 006660 102402
 7797 006662 101401
 7798 006664 100401
 (1) 006666
 (2) 006666 104000
 7799 006670 105100
 7800 006672 001401
 (3) 006674 104000

BVS SNM5A ;CHECK CC=1000
 BLOS SNM5A
 BMI SNM5B
 SNM5A: EMT ;CC'S NOT SET PROPERLY
 SNM5B: INC RO ;RO=377
 COMB RO ;RO=0
 BEQ TS111
 EMT ;MODE 5 DID NOT DEC RO CORRECTLY

 ; THIS TEST VERIFIES MODE 6 SOP NON-MODIFYING INSTRUCTIONS.
 ; RO IS SET TO 377 AND A MODE 6 TST INSTRUCTION IS EXECUTED
 ; USING RO AND AN OFFSET OF -377. THE CC'S ARE CHECKED AS WELL
 ; AS RO TO INSURE IT WAS NOT ALTERED.
 ;

 ; TEST 111 TEST MODE 6 W/ SOP NON-MODIFYING INSTS
 ; *****

TS111: CLR RO ;RO=0
 CLR (RO) ;LOC 0=0
 COM (RO) ;LOC 0=-1
 COMB RO ;RO=377
 SCC ;SET CC=0111
 CLN
 TST -377(RO) ;TRY TST W/ MODE 6
 BVS SNM6A ;CHECK CC=1000
 BLOS SNM6A
 BMI SNM6B
 SNM6A: EMT ;CC'S INCORRECT
 SNM6B: COMB RO ;RO=0
 BEQ TS112
 EMT ;TST MODE 6 INCORRECTLY CHANGED RO

7802
7803
7804
7805
7806
7807
7808
7809
7810
(2)
(3)
(2) 006676
7811 006676 005000
7812 006700 005010
7813 006702 005110
7814 006704 105100
7815 006706 000277
7816 006710 000250
7817 006712 005770 000001
7818 006716 102402
7819 006720 101401
7820 006722 100401
(1) 006724
(2) 006724 104000
7821 006726 105100
7822 006730 001401
(3) 006732 104000
7823
7824
7825
7826
7827
7828
7829
7830
(2)
(3)
(2) 006734
7831 006734 005000
7832 006736 005100
7833 006740 005004
7834 006742 060004
7835 006744 005204
7836 006746 001401
(3) 006750 104000
7837
7838
7839
7840
7841
7842
(2)
(3)
(2) 006752
7843 006752 005000
7844 006754 005004

; THIS TEST VERIFIES MODE 7 SOP NON-MODIFYING INSTRUCTIONS.
; IT USES A POINTER TO LOC. 0 STORED AT LOC. 400 TO TST LOC. 0.
; RO IS SET TO 377 AND LOC. 0 IS TESTED THRU THE POINTER AT 400 USING
; RO AND AN OFFSET OF 1.

; TEST 112 TEST MODE 7 W/ SOP NON-MODIFYING INSTS.

TS112:

CLR R0 ;RO=0
CLR (R0) ;LOC 0=0
COM (R0) ;LOC 0=-1
COMB R0 ;RO=377
SCC ;CC=0111
CLN
TST @1(R0) ;TRY TST W/ MODE 7
BVS SNM7A ;CHECK CC=1000
BLOS SNM7A
BMI SNM7B
SNM7A: EMT ;CC'S NOT CORRECT
SNM7B: COMB R0 ;RO=0
BEQ TS113
EMT ;TST MODE 7 INCORRECTLY CHANGED RO

; THIS TEST VERIFIES MODE 0 DOUBLE OPERAND INSTRUCTIONS. IT SETS
; DATA IN R0 AND R4 AND USES THE ADD INSTRUCTION TO TEST THE DOP
; MICROCODE.

; TEST 113 TEST MODE 0 DOUBLE-OPERAND (DOP) INSTS.

TS113:

CLR R0 ;RO=0
COM R0 ;RO=-1
CLR R4 ;R4=0
ADD R0,R4 ;TRY ADD: R4=-1
INC R4 ;R4=0
BEQ TS114
EMT ;ADD INST. FAILED W/ MODE 0

; THIS TEST VERIFIES THE MOVE INSTRUCTION WITH MODE 0 TO MODE 0.

; TEST 114 MOV MODE 0 TO MODE 0

TS114:

CLR R0 ;RO=0
CLR R4 ;R4=0

7845 006756 005100
7846 006760 010004
7847 006762 005204
7848 006764 001401
(3) 006766 104000

COM R0 ;R0=-1
MOV R0,R4 ;TRY MOVE -1 TO R4
INC R4 ;INC R4
BEQ TS115
EMT ;MOVE FAILED MODE 0 TO MODE C

7849
7850
7851
7852
7853
7854

: THIS TEST VERIFIES THE SUBTRACT INSTRUCTION WITH MODE 0,0.
: *****

(2)
(3)
(2) 006770

;TEST 115 TEST SUB MODE 0,0
; *****

7855 006770 005000
7856 006772 005004
7857 006774 005204
7858 006776 160400
7859 007000 100003
7860 007002 001402
7861 007004 102401
7862 007006 103401

(S115:
CLR R0 ;R0=0
CLR R4 ;R4=0
INC R4 ;R4=1
SUB R4,R0 ;TRY SUB 0,0 R0= 1
BPL SUB0 ;CC=1001
BEQ SUB0
BVS SUB0
BCS SUB0A

(1) 007010
(2) 007010 104000
7863 007012 005200
7864 007014 001401
(3) 007016 104000

SUB0: EMT ;CONDITION CODE FAILED ON SUB
SUB0A: INC R0
BEQ TS116
EMT ;DATA RESULT OF SUB FAILED

7865
7866
7867
7868
7869
7870
7871
7872
7873
7874

: THIS TEST QUICKLY VERIFIES THE REMAINING DOP MODIFYING INSTRUCTIONS
: WITH MODE 0,0 TO PROVIDE A BASELINE FOR SUBSEQUENT TESTS.
: SINGLE OPERAND INSTRUCTIONS ARE USED TO SET UP DATA IN R0 AND R4
: BEFORE EACH OF THE SEVERAL DOP MODIFYING INSTRUCTIONS ARE USED AND
: VERIFIED.
: *****

(2)
(3)
(2) 007020

;TEST 116 TEST ALL THE DOP INSTRUCTIONS W/ SOURCE MODE 0,0
; *****

7875 007020 005000
7876 007022 010004
7877 007024 001401
(2) 007026 104000

TS116:
CLR R0 ;R0=0
MOV R0,R4 ;TRY MOVE MODE 0,0
BEQ DOP0A
EMT ;Z-BIT NOT SET

7878 007030 005200
7879 007032 005100
7880 007034 005104
7881 007036 040004
7882 007040 005304
7883 007042 001401

DOP0A: INC R0 ;R0=1
COM R0 ;R0=177776
COM R4 ;R4=177777
BIC R0,R4 ;TRY BIC: R4=1
DEC R4 ;R4=0
BEQ DOP0B

(2) 007044 104000
7884 007046 050004
7885 007050 005204
7886 007052 005204
7887 007054 001401

DOP0B: EMT ;BIC CLEAR RESULT INCORRECT
BIS R0,R4 ;TRY BIS: R4=177777
INC R4
INC R4 ;R4=0

(2) 007056 104000

DOP0C: BEQ DOP0C
EMT ;RESULT OF BIS INCORRECT

```

7888 007060 005000
7889 007062 105100
7890 007064 005004
7891 007066 005104
7892 007070 040004
7893 007072 0600C4
7894 007074 005204
7895 007076 001401
(2) 007100 104000
7896 007102 160004
7897 007104 105404
7898 007106 005204
7899 007110 001401
(3) 007112 104000
7900
7901
7902
7903
7904
7905
7906
(2)
(3)
(2) 007114
7907 007114 005000
7908 007116 005010
7909 007120 105110
7910 007122 005220
7911 007124 005400
7912 007126 060037 000000
7913 007132 100403
7914 007134 001402
7915 007136 102401
7916 007140 103401
(1) 007142
(2) 007142 104000
7917 007144 105137 000000
7918 007150 005337 000000
7919 007154 001401
(3) 007156 104000
7920
7921
7922
7923
7924
7925
7926
(2)
(3)
(2) 007160
7927 007160 005000
7928 007162 005004
7929 007164 005204
7930 007166 020400
7931 007170 003001
(2) 007172 104000

```

```

DOPOC: CLR R0 ;R0=0
        COMB R0 ;R0=377
        CLR R4 ;R4=0
        COM R4 ;R4=177777
        BIC R0,R4 ;R4=177400
        ADD R0,R4 ;TRY ADD: R4=177777
        INC R4 ;R4=0
        BEQ DOP00
        EMT ;RESULT OF ADD INCORRECT
DOPO0: SUB R0,R4 ;177401=R4
        NEGB R4 ;R4=177777
        INC R4 ;R0=0
        BEQ TS117
        EMT ;RESULT OF SUB INCORRECT

```

```

;.....
;
; THIS TEST VERIFIES MODE 0,X DOUBLE OPERAND INSTRUCTIONS. IT SETS
; DATA IN R0 AND LOCATION 0 AND OPERATES UPON IT USING DOP INSTRUCTIONS.
;
;.....
;TEST 117 TEST MODE 0,X DOUBLE-OPERAND INSTRUCTIONS
;.....
TS117:

```

```

        CLR R0 ;R0=0
        CLR (R0) ;LOC. 0=0
        COMB (R0) ;LOC. 0=377
        INC (R0) ;LOC. 0=400 R0=2
        NEG R0 ;R0=-2
        ADD R0,B#0 ;TRY ADD 0,3; LOC. 0=376
        BMI DOP03A ;CC=0001?
        BEQ DOP03A
        BVS DOP03A
        BCS DOP03B
DOP03A: EMT ;CC'S NOT SET CORRECTLY
DOP03B: COMB B#0 ;LOC. 0=1
        DEC B#0 ;LOC. 0=0
        BEQ TS120
        EMT ;DATA RESULT INCORRECT

```

```

;.....
;
; THIS TEST VERIFIES MODE 0,0 DOP NON-MODIFYING INSTRUCTIONS.
; R0 AND R4 ARE PRESET TO 0 AND 1 RESPECTIVELY. COMPARE INSTRUCTIONS ARE
; THEN EXECUTED AND CHECKED. FIRST R4 IS COMPARED TO R0 THEN R0 TO R4.
;
;.....
;TEST 120 TEST DOP NON-MODIFYING INST. W/ SOURCE MODE 0,0
;.....
TS120:

```

```

        CLR R0 ;R0=0
        CLR R4 ;R4=0
        INC R4 ;R4=1
        CMP R4,R0 ;TRY COMPARE R4 TO R0
        BGT DNM1
        EMT ;CC'S NOT CORRECT FOR CMP

```


7932 007174 020004
7933 007176 002401
(2) 007200 104000
7934 007202 005200
7935 007204 020400
7936 007206 001401
(2) 007210 104000
7937 007212 005000
7938 007214 005100
7939 007216 005004
7940 007220 030004
7941 007222 001401
(2) 007224 104000
7942 007226 005304
7943 007230 030004
7944 007232 100401
(3) 007234 104000
7945
7946
7947
7948
7949
7950
(2)
(3)
(2) 007236
7951 007236 005000
7952 007240 005010
7953 007242 005110
7954 007244 005200
7955 007246 020037 000000
7956 007252 100403
7957 007254 001402
7958 007256 102401
7959 007260 103401
(1) 007262
(2) 007262 104000
7960 007264 005300
7961 007266 001002
7962 007270 005210
7963 007272 001401
(2) 007274
(3) 007274 104000
7964
7965
7966
7967
7968
7969
7970
7971
(2)
(3)
(2) 007276
7972 007276 005000
7973 007300 005100

DNM1: CMP R0,R4 ;TRY COMPARE R0 TO R4
BLT DNM2 ;CC'S NOT CORRECT FOR CMP
EMT ;RO=1
DNM2: INC R0 ;RO=1
CMP R4,R0 ;TRY COMPARE R4-1 TO R0-1
BEQ DNM3 ;CC'S NOT CORRECT (Z-1) FOR CMP
EMT ;RO=0
DNM3: CLR R0 ;RO=177777
COM R0 ;R4=0
CLR R4 ;TRY BIT R0 TO R4
BIT R0,R4
BEQ DNM4 ;CC'S NOT CORRECT FOR BIT
EMT ;R4=177777
DNM4: DEC R4 ;TRY BIT AGAIN
BIT R0,R4
BMI TS121 ;CC'S NOT CORRECT FOR BIT
EMT

.....
; THIS TEST VERIFIES MODE 0,X DOUBLE OPERAND NON-MODIFYING INSTRUCTIONS.
; IT SETS DATA IN R0 AND LOCATION 0 AND COMPARES THEM USING DOPNM INSTRUCTIONS.
;

;TEST 121 TEST MODE 0,X DOUBLE-OPERAND NON-MODIFYING INSTS.
;

TS121: CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
INC R0 ;RC=1
CMP R0,800 ;TRY CMP MODE 0,3
BMI DNM03A ;CC=0001
BEQ DNM03A
BVS DNM03A
BCS DNM03B

DNM03A: EMT ;CC'S NOT SET CORRECTLY
DNM03B: DEC R0
BNE DNM03C
INC (R0)
BEQ TS122

DNM03C: EMT ;DATA INCORRECTLY MODIFIED BY CMP
;

; THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS. R0 IS SET TO -1
; AND LOC 0 TO 1. R4 IS THEN CLEARED AND USED TO POINT TO LOC 0.
; IN THE ADD MODE 1 INSTRUCTION, LOC 0 IS ADDED TO R0 AND THE
; RESULTS VERIFIED.
;

;TEST 122 TEST MODE 1 W/ DOP INST.
;

TS122: CLR R0 ;RO=0
COM R0 ;RO=177777

7974 007302 005004
7975 007304 005014
7976 007306 005214
7977 007310 06140C
7978 007312 001401
(3) 007314 104000

CLR R4 ;R4=0
CLR (R4) ;LOC 0=0
INC (R4) ;LOC 0=1
ADD (R4),R0 ;TRY ADD SOURCE MODE 1
BEQ TS123
EMT ;RESULT OF ADD INCORRECT

7979
7980
7981
7982
7983
7984
7985
7986

.....
; THIS TEST VERIFIES MODE 1 DOP BYTE INSTRUCTIONS WHICH ADDRESS
; EVEN BYTES. LOC. 0 IS SET TO -1 AND R4 IS CLEARED. THEN R4 IS
; SET TO 1 USING A BISB THRU R0 WITH MODE 1.
;

(2) 007316
(3)
(2) 007316 005000
7987 007316 005000
7988 007320 005010
7989 007322 005110
7990 007324 005004
7991 007326 151004
7992 007330 105104
7993 007332 001401
(3) 007334 104000

.....
; TEST 123 TEST MODE 1 - EVEN BYTE W/ DOP INSTS.
;

TS123:
CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
CLR R4 ;R4=0
BISB (R0),R4 ;TRY MODE 1- EVEN BYTE W/ DOP
COMB R4 ;R4=0
BEQ TS124
EMT ;RESULT OF BISB IS INCORRECT

7994
7995
7996
7997
7998
7999

.....
; THIS TEST VERIFIES MODE 1 DOP NON-MODIFYING INSTRUCTIONS
; WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO -1 AND R0 IS CLEARED
; AND USED AS THE ADDRESSING REGISTER. R4 IS SET TO 377 AND A
; MODE 1,0 CMPB INSTRUCTION IS USED THE RESULTS VERIFIED.
;

8000
8001
8002
(2) 007336
(3)
(2) 007336 005000
8003 007336 005000
8004 007340 005010
8005 007342 005110
8006 007344 005004
8007 007346 105104
8008 007350 121004
8009 007352 001401
(3) 007354 104000

.....
; TEST 124 TEST MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING INST.
;

TS124:
CLR R0 ;R0=0
CLR (R0) ;LOC 0=0
COM (R0) ;LOC 0=177777
CLR R4 ;R4=0
COMB R4 ;R4=377
CMPB (R0),R4 ;TRY MODE 1 - EVEN BYTE W/ DOP NON-MODIFYING
BEQ TS125
EMT ;RESULT OF CMPB INCORRECT

8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8020

.....
; THIS TEST VERIFIES MODE 1,0 MOVB INSTRUCTIONS
; WHICH ADDRESS EVEN BYTES. LOC. 0 IS SET TO 177400, R0 IS CLEARED AND
; R4 IS SET TO -1. MOVB ARE USED TO MOVE BYTE 0 TO R4. THIS
; VERIFIES THAT THE PROPER BYTE WAS SELECTED AND THAT THE SIGN-X-TEND
; FUNCTION WITH MODE 0.
; THEN LOC. 0 IS COMPLEMENTED AND THE SAME PROCEDURE EXERCISES
; THE LOGIC FOR COMPLEMENTARY DATA.
; THIS TEST EXERCISES UNIQUE MICROCODE.

[5]

8021
8022 (2)
(3)
(2) 007356
8023 007356 005000
8024 007360 005010
8025 007362 105110
8026 007364 005110
8027 007366 005004
8028 007370 005104
8029 007372 111004
8030 007374 005704
8031 007376 001401
(2) 007400 104000
8032 007402 005110
8033 007404 111004
8034 007406 100401
(3) 007410 104000
8035
8036
8037
8038
8039
8040
8041
8042
8043
(2)
(3)
(2) 007412
8044 007412 005000
8045 007414 005010
8046 007416 005004
8047 007420 005204
8048 007422 105114
8049 007424 151410
8050 007426 005210
8051 007430 001401
(3) 007432 104000
8052
8053
8054
8055
8056
8057
8058
8059
8060
(2)
(3)
(2) 007434
8061 007434 005000
8062 007436 005010
8063 007440 005110
8064 007442 012004

```
;  
;*****  
;TEST 125 TEST MOV INSTRUCTION MODE 1,0 EVEN BYTE  
;*****  
TS125:  
      CLR      R0          ;R0=0  
      CLR      (R0)       ;LOC 0=0  
      COMB     (R0)       ;LOC 0=177400  
      COM      (R0)       ;  
      CLR      R4          ;R4=0  
      COM      R4         ;R4=177777  
      MOVB     (R0),R4    ;R4=0  
      TST      R4         ;CHECK SIGN OF WORD  
      BEQ      DOP1  
DOP1: EMT                ;MOVB SHOULD SIGN X TEND  
      COM      (R0)       ;LOC 0=177777  
      MOVB     (R0),R4    ;DO MOVB W/ EVEN BYTE  
      BMI      TS126  
      EMT                ;MOVB SHOULD SIGN X-TEND  
;  
;*****  
; THIS TEST VERIFIES MODE 1 DOP INSTRUCTIONS WHICH REFERENCE  
; ODD BYTES. LOC. 0 IS SET TO 177400. R0 IS SET TO 0 AND R4 IS  
; SET TO 1. THE BISB INSTRUCTION USES THE DATA IN BYTE 1 TO SET BYTE 0.  
; THE RESULT IS CHECKED BY INCREMENTING THE WORD (LOC. 0) TO ZERO.  
;  
;*****  
;TEST 126 TEST MODE 1-ODD BYTE W/ DOP INSTS.  
;*****  
TS126:  
      CLR      R0          ;R0=0  
      CLR      (R0)       ;LOC. 0=0  
      CLR      R4          ;R4=0  
      INC      R4         ;R4=1  
      COMB     (R4)       ;LOC. 0=177400  
      BISB     (R4),(R0)  ;TRY TO BIS LOW ORDER BITS W/ MODE 1  
      INC      (R0)       ;CHECK RESULT  
      BEQ      TS127  
      EMT                ;RESULT OF BISB INCORRECT  
;  
;*****  
; THIS TEST VERIFIES MODE 2 DOP INSTRUCTIONS. LOC. 0 IS SET TO -1.  
; R0 IS CLEARED AND USED AS THE MODE 2 ADDRESSING REGISTER TO MOVE LOC. 0  
; TO R7. THE DATA RESULTS ARE VERIFIED AND THE INCREMENTING OF THE REGISTER  
; IS CHECKED.  
;  
;*****  
;TEST 127 TEST MODE 2 W/ DOP INSTS.  
;*****  
TS127:  
      CLR      R0          ;R0=0  
      CLR      (R0)       ;LOC. 0=0  
      COM      (R0)       ;LOC. 0=177777  
      MOV      (R0),R4    ;TRY MOVE MODE 2,0
```

8065 007444 005204
8066 007446 001401
(2) 007450 104000
8067 007452 005300
8068 007454 005300
8069 007456 001401
(3) 007460 104000
8070
8071
8072
8073
8074
8075
8076
8077
8078
8079
8080
(2)
(3)
(2) 007462
8081 007462 005000
8082 007464 010010
8083 007466 005110
8084 007470 142010
8085 007472 105737 000001
8086 007476 001401
(2) 007500 104000
8087 007502 105137 000000
8088 007506 001401
(3) 007510 104000
8089
8090
8091
8092
8093
8094
8095
8096
(2)
(3)
(2) 007512
8097 007512 005000
8098 007514 005004
8099 007516 005010
8100 007520 005110
8101 007522 105120
8102 007524 112004
8103 007526 005204
8104 007530 001401
(2) 007532 104000
8105 007534 005740
8106 007536 005700
8107 007540 001401
(3) 007542 104000
8108

```
INC R4 ;CHECK R4
BEQ DOP2
EMT ;RESULT OF MOV INST INCORRECT
DOP2: DEC R0 ;TEST R0 AFTER MODE 2
DEC R0
BEQ TS130
EMT ;REGISTER NOT INCREMENTED IN MODE 2

;*****
;
; THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH ADDRESS
;EVEN BYTES. LOC. 0 IS SET TO -1. R0 IS CLEARED AND USED AS THE
;ADDRESSING REGISTER IN A TEST WHICH TRIES TO CLEAR BYTE 1 USING
;BYTE 0 DATA AND A BICB. UNIQUE IN THIS TEST IS USE OF THE
;SAME ADDRESSING REGISTER FOR BOTH SOURCE AND DESTINATION. THE SOURCE AND
;DESTINATION IS CHECKED TO INSURE PROPER FUNCTIONING.
;
;*****
;TEST 130 TEST MODE 2 - EVEN BYTE W/ DOP INST.
;*****
TS130:
CLR R0 ;R0=0
MOV R0,(R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
BICB (R0), (R0) ;TRY TO CLEAR BYTE 1 FROM BYTE 0 W/ BICB
TST B01 ;CHECK RESULT
BEQ DOPB2A
EMT ;BICB DESTINATION INCORRECT
DOPB2A: COM B00 ;CHECK BICB SOURCE
BEQ TS131
EMT ;BICB SOURCE INCORRECTLY CHANGED

;*****
;
; THIS TEST VERIFIES MODE 2 DOP BYTE INSTRUCTIONS WHICH REFERENCE
;ODD BYTES. R0 IS SET TO 1, LOC. 0 IS SET TO 177400, AND R4 IS CLEARED.
;A MODE 2 MOVB USES R0 TO MOVE BYTE 1 TO R4. AN INCREMENT
;IS USED TO CHECK THAT THE PROPER BYTE WAS MOVED AND SIGN X-TENDED.
;
;*****
;TEST 131 TEST MODE 2 - ODD BYTE W/ DOP INST.
;*****
TS131:
CLR R0 ;R0=0
CLR R4 ;R4=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
COMB (R0), R4 ;LOC 0=177400; R0=1
MOVB (R0)+, R4 ;TRY DOP MODE 2 W/ ODD BYTE
INC R4 ;CHECK RESULT OF MOVB
BEQ DOPB28
EMT ;RESULT OF MOVB INCORRECT
DOPB28: TST -(R0) ;BUMP R0 DOWN BY 2
TST R0 ;CHECK R0
BEQ TS132
EMT ;MODE 2 BYTE DID NOT INCREMENT REG. CORRECTLY

;*****
```

8109
8110
8111
8112
8113
8114
8115
8116
(2)
(3)
(2) 007544
8117 007544 012737 052525 000000
8118 007552 012700 125252
8119 007556 053700 000000
8120 007562 005200
8121 007564 001401
(3) 007566 104000
8122
8123
8124
8125
8126
8127
8128
8129
(2)
(3)
(2) 007570
8130 007570 012737 052652 000000
8131 007576 005000
8132 007600 153700 000000
8133 007604 022700 000252
8134 007610 001401
(3) 007612 104000
8135
8136
8137
8138
8139
8140
8141
8142
(2)
(3)
(2) 007614
8143 007614 012737 052652 000000
8144 007622 005000
8145 007624 153700 000001
8146 007630 022700 000125
8147 007634 001401
(3) 007636 104000
8148
8149
(2)
(3)
(2) 007640

```
;  
; THIS TEST VERIFIES MODE 3 DOUBLE-OPERAND INSTRUCTIONS.  
; LOC. 0 IS LOADED WITH ALTERNATING ZEROES AND ONES; AND RO IS LOADED  
; WITH ALTERNATING ONES AND ZEROES. A MODE 3 BIS IS USED TO SET RO  
; TO -1 BY USING LOC. 0 AS THE SOURCE TO BIS THE ZEROES IN RO. THE  
; RESULT IS TESTED BY INCREMENTING RO AND CHECKING FOR ZERO.  
;.....  
;TEST 132 TEST MODE 3 W/ DOP INSTS.  
;.....  
TS132:  
MOV #052525,RO ;MOVE 52525 TO LOC. 0  
MOV #125252,RO ;SET ALT. ONE AND ZERO IN RO  
BIS RO,RO ;TRY TO SET ALL OTHER BITS W/ MODE 3  
INC RO ;TEST RESULT  
BEQ TS133  
EOT ;BIS W/ MODE 3 INCORRECT RESULT  
;.....  
;  
; THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS WHICH  
; ADDRESS EVEN BYTES. BYTE 0 IS SET TO ALTERNATING 1'S AND 0'S; BYTE 1,  
; ALTERNATING 0'S AND 1'S. RO IS CLEARED AND A BISB IS USED TO  
; SET THE LOW BYTE OF RO TO 252.  
;.....  
;TEST 133 TEST MODE 3 - EVEN BYTE W/ DOP INSTS.  
;.....  
TS133:  
MOV #52652,RO ;MOVE 1'S AND 0' PATTERN TO LOC. 0  
CLR RO ;RO=0  
BISB RO,RO ;TRY RO=252 W/ MODE 3 - EVEN BYTE  
CMP #252,RO ;BISB W/ EVEN BYTE SUCCESSFUL?  
BEQ TS134  
EOT ;BISB W/ MODE 3 - EVEN BYTE FAILED  
;.....  
;  
; THIS TEST VERIFIES MODE 3 DOUBLE OPERAND BYTE INSTRUCTIONS  
; WHICH ADDRESS ODD BYTES. THE SAME PROCEDURE USED IN PREVIOUS  
; TEST IS USED HERE. THIS TIME BYTE 1 IS USED AS THE SOURCE BYTE.  
; THE EXPECTED RESULT IS: RO = 125.  
;.....  
;TEST 134 TEST MODE 3 - ODD BYTE W/ DOP INSTS.  
;.....  
TS134:  
MOV #52652,RO ;MOVE 1'S AND 0'S PATTERN TO LOC 0  
CLR RO ;RO=0  
BISB #1,RO ;TRY RO=152 W/ MODE 3 - ODD BYTE  
CMP #125,RO ;RO=125?  
BEQ TS135  
EOT ;BISB W/ MODE 3 - ODD BYTE FAILED  
;.....  
;TEST 135 TEST DEST. MODE 0-BYTE W/ DOP NON-MODIFYING MST  
;.....  
TS135:
```

8150 007640 005000
8151 007642 105100
8152 007644 000263
8153 007646 132700 000200
8154 007652 001403
8155 007654 102402
8156 007656 103001
8157 007660 100401
(1) 007662
(2) 007662 104000
8158 007664 105100
8159 007666 001401
(3) 007670 104000

CLR R0 ;RO=0
COMB R0 ;RO=377
+SEC!SEV ;SET C AND V BITS
BITB #200,R0 ;TRY DOPNM DEST. MODE 0 BYTE
BEQ DNMBOA ;BR TO ERROR IF Z BIT SET
BVS DNMBOA ;BR TO ERROR IF V BIT SET
BCC DNMBOA ;BR TO ERROR IF C BIT CLEAR.
BMI DNMBOB
DNMBOA: EMT ;CC'S INCORRECT
DNMBOB: COMB R0 ;CHECK DESTINATION DATA
BEQ TS136
EMT ;DEST. DATA MODIFIED

8160
8161
(2)
(3)
(2) 007672
8162 007672 005000
8163 007674 005010
8164 007676 000241
8165 007700 032710 177777
8166 007704 100403
8167 007706 102402
8168 007710 103401
8169 007712 001401
(1) 007714
(2) 007714 104000
8170 007716 005710
8171 007720 001401
(3) 007722 104000

;TEST 136 TEST DEST. MODE 1 W/ DOP NON-MODIFYING INST

TS136:
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
CLC ;CLEAR C BIT
BIT #177777,(R0) ;TRY DOPNM DEST. MODE 1
BMI DNM1A ;BR TO ERROR IF N BIT SET
BVS DNM1A ;BR TO ERROR IF V BIT SET
BCS DNM1A ;BR TO ERROR IF C BIT SET
BEQ DNM1B
DNM1A: EMT ;COND. CODES INCORRECT
DNM1B: TST (R0) ;CHECK TEST DATA
BEQ TS137
EMT ;DESTINATION DATA MODIFIED

8172
8173
(2)
(3)
(2) 007724
8174 007724 005000
8175 007726 005010
8176 007730 052710 125252
8177 007734 032720 077777
8178 007740 102402
8179 007742 001401
8180 007744 100001
(1) 007746
(2) 007746 104000
8181 007750 005300
8182 007752 005300
8183 007754 001401
(1) 007756
(2) 007756 104000
8184 007760 022710 125252
8185 007764 001401
(3) 007766 104000

;TEST 137 TEST DEST, MODE 2 W/ DOP NON-MODIFYING INST.

TS137:
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
BIS #125252,(R0) ;LOC. 0=125252
BIT #77777,(R0) ;TRY DOPNM INST W/ MODE 2
BVS DNM2A ;BR TO ERROR IF V BIT SET
BEQ DNM2A ;BR TO ERROR IF Z-BIT SET
BPL DNM2B
DNM2A: EMT ;COND. CODES INCORRECT
DNM2B: DEC R0 ;DECREMENT R0 TO CHECK IT.
DEC R0
BEQ DNM2D
DNM2C: EMT ;MODE 2 REGISTER NOT INCREMENTED BY 2
DNM2D: CMP #125252,(R0) ;CHECK DEST. DATA
BEQ TS140
EMT ;DEST. DATA MODIFIED

8186
8187
(2)

;TEST 140 TEST DEST. MODE 2-BYTE, W/DOP NON-MODIFYING INST

15

```
(3)
(2) 007770
8188 007770 005000
8189 007772 005010
8190 007774 052710 052652
8191 010000 000263
8192 010002 132720 000201
8193 010006 001403
8194 010010 103002
8195 010012 102401
8196 010014 100401
(1) 010016
(2) 010016 104000
8197 010020 005300
8198 010022 001401
(2) 010024 104000
8199 010026 005200
8200 010030 132720 000201
8201 010034 001402
8202 010036 102401
8203 010040 100001
(1) 010042
(2) 010042 104000
8204 010044 005300
8205 010046 005300
8206 010050 001401
(2) 010052 104000
8207 010054 022710 052652
8208 010060 001401
(3) 010062 104000
8209
8210
8211
(2)
(3)
(2) 010064
8212 010064 005000
8213 010066 005010
8214 010070 052710 125125
8215 010074 105100
8216 010076 005200
8217 010100 005010
8218 010102 000263
8219 010104 132730 000201
8220 010110 001403
8221 010112 102402
8222 010114 103001
8223 010116 100001
(1) 010120
(2) 010120 104000
8224 010122 022700 000402
8225 010126 001401
(2) 010130 104000
8226 010132 005200
8227 010134 005200
8228 010136 132730 000201
```

```
*****
TS140:
      CLR      RO          ;RO=0
      CLR      (RO)       ;LOC. 0=0
      BIS      #52652,(RO) ;LOC. 0=52652
      +SEC!SEV
      BITB     #201,(RO)+ ;SET C AND V BITS
      BEQ      DNMB2A     ;TRY DOPNM INST. W/ MODE 2 EVEN BYTE
      BCC      DNMB2A     ;BR TO ERROR IF Z-BIT SET
      BVS      DNMB2A     ;BR TO ERROR IF C-BIT CLEAR
      BMI      DNMB2B     ;BR TO ERROR IF V-BIT SET

DNMB2A:
      EMT
DNMB2B: DEC      RO          ;COND. CODES INCORRECT
      BEQ      DNMB2C     ;CHECK DEST. REGISTER.
      EMT
DNMB2C: INC      RO          ;DEST. REGISTER NOT INCREMENTED B. 1
      BITB     #201,(RO)+ ;RO=1
      BEQ      DNMB2D     ;TRY DOPNM INST. W/MODE 2-ODD BYTE
      BVS      DNMB2D     ;BR TO ERROR IF Z-BIT SET
      BPL      DNMB2E     ;BR TO ERROR IF V-BIT SET

DNMB2D:
      EMT
DNMB2E: DEC      RO          ;COND. CODES INCORRECT
      DEC      RO          ;DEC RO TO CHECK IT.
      BEQ      DNMB2F
      EMT
DNMB2F: CMP      #52652,(RO) ;DEST. REGISTER NOT INCREMENTED BY 1
      BEQ      TS141      ;CHECK DEST. DATA IS UNMODIFIED
      EMT
                               ;DEST. DATA WAS MODIFIED.
*****
```

```
*****
;TEST 141 TEST DEST. MODE 3-BYTES W/DOP NON-MODIFYING INST.
*****
TS141:
      CLR      RO          ;RO=0
      CLR      (RO)       ;LOC. 0=0
      BIS      #125125,(RO) ;LOC. 0=125125
      COMB     RO          ;RO=377
      INC      RO          ;RO=400
      CLR      (RO)       ;LOC. 400=0
      +SEC!SEV
      BITB     #201,(RO)+ ;C-BIT=V-BIT=1
      BEQ      DNMB3A     ;TRY DOPNM W/MODE 3-EVEN BYTE
      BVS      DNMB3A     ;BR TO ERROR IF Z BIT SET
      BCC      DNMB3A     ;BR TO ERROR IF V BIT SET
      BPL      DNMB3B     ;BR TO ERROR IF C BIT CLEAR

DNMB3A:
      EMT
DNMB3B: CMP      #402,RO     ;COND. CODES INCORRECT
      BEQ      DNMB3C     ;CHECK DEST. REGISTER INC. BY 2 AND INC BY 2 AGAIN
      EMT
DNMB3C: INC      RO          ;DEST. REGISTER NOT INCREMENTED BY 2
      INC      RO          ;RO=404
      BITB     #201,(RO)+ ;TRY DOPNM DEST MODE 3 BYTE(ODD)
*****
```

8229 010142 001402
8230 010144 102401
8231 010146 100401
(1) 010150
(2) 010150 104000
8232 010152 005004
8233 010154 022714 125125
8234 010160 001401
(3) 010162 104000
8235
8236
(2)
(3)
(2) 010164

BEQ DNMB3D ;BR TO ERROR IF Z BIT SET
BVS DNMB3D ;BR TO ERROR IF V BIT SET
BMI DNMB3E
DNMB3D: EMT ;COND. CODES INCORRECT
DNMB3E: CLR R4 ;R4=0
CMP #125125,(R4) ;CHECK DEST. DATA
BEQ TS142
EMT ;DEST. DATA MODIFIED

;TEST 142 TEST DEST. MODE 4 W/DOP NON-MODIFYING INST.

;*****
TS142:

8237 010164 005000
8238 010166 005010
8239 010170 052710 125252
8240 010174 052700 000002
8241 010200 000277
8242 010202 032740 020000
8243 010206 100403
8244 010210 102402
8245 010212 103001
8246 010214 001001
(1) 010216
(2) 010216 104000
8247 010220 005700
8248 010222 001401
(2) 010224 104000
8249 010226 022737 125252 000000
8250 010234 001401
(3) 010236 104000
8251
8252
(2)
(3)
(2) 010240

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
BIS #125252,(R0) ;LOC. 0=125125
BIS #2,R0 ;R0=2
SCC ;SET ALL COND. CODE BITS
BIT #200C0,-(R0) ;TRY DOPNM W/ MODE 4
BMI DNMAA ;BR TO ERROR IF N-BIT SET
BVS DNMAA ;BR TO ERROR IF V-BIT SET
BCC DNMAA ;BR TO ERROR IF C-BIT CHAR
BNE DNMA8
DNMAA: EMT ;COND. CODES INCORRECT
DNMA8: TST R0 ;CHECK DEST. REGISTER
BEQ DNMAC
EMT ;DEST. REGISTER NOT DECREMENTED BY 2
DNMAC: CMP #125252,R0 ;CHECK DEST. DATA
BEQ TS143
EMT ;DEST. DATA MODIFIED

;TEST 143 TEST DEST. MODE 4-BYTE W/ DOP NON-MODIFYING INST.

;*****
TS143:

8253 010240 005000
8254 010242 005010
8255 010244 052710 052652
8256 010250 052700 000002
8257 010254 000257
8258 010256 132740 000201
8259 010262 102403
8260 010264 001402
8261 010266 103401
8262 010270 001001
(1) 010272
(2) 010272 104000
8263 010274 022700 000001
8264 010300 001401
(2) 010302 104000
8265 010304 132740 000201
8266 010310 001401
8267 010312 100401
(1) 010314

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
BIS #52652,(R0) ;LOC. 0=52652
BIS #2,R0 ;R0=2
CCC ;COND. CODES=0
BITB #201,-(R0) ;TRY DOPNM INST W/MODE 4 ODD BYTE
BVS DNMB4A ;BR TO ERROR IF V BIT SET
BEQ DNMB4A ;BR TO ERROR IF Z BIT SET
BCS DNMB4A ;BR TO ERROR IF C BIT SET
BNE DNMB48
DNMB4A: EMT ;COND. CODES INCORRECT
DNMB48: CMP #1,R0 ;CHECK DEST. REGISTER
BEQ DNMB4C
EMT ;DEST REG. NOT DECREMENTED BY 1
DNMB4C: BITB #201,-(R0) ;TRY DOPNM INST. W/MODE 4 EVEN BYTE
BEQ DNMB4D ;BR TO ERROR IF Z-BIT SET
BMI DNMB4E
DNMB4D:
DNMB4E:

(2) 010314 104000
8268 010316 005700
8269 010320 001401
(2) 010322 104000
8270 010324 022710 052652
8271 010330 001401
(3) 010332 104000
8272
8273
(2)
(3)
(2) 010334
8274 010334 005000
8275 010336 005010
8276 010340 052710 100000
8277 010344 052700 000402
8278 010350 000277
8279 010352 032750 100000
8280 010356 102403
8281 010360 103002
8282 010362 001401
8283 010364 100401
(1) 010366
(2) 010366 104000
8284 010370 022700 000400
8285 010374 001401
(2) 010376 104000
8286 010400 022737 100000 000000
8287 010406 001401
(3) 010410 104000
8288
8289
(2)
(3)
(2) 010412
8290 010412 005000
8291 010414 005010
8292 010416 052710 000001
8293 010422 005100
8294 010424 032760 000001 000001
8295 010432 001403
8296 010434 102402
8297 010436 103001
8298 010440 100001
(1) 010442
(2) 010442 104000
8299 010444 022700 177777
8300 010450 001401
(2) 010452 104000
8301 010454 022737 000001 000000
8302 010462 001401
(3) 010464 104000
8303
8304
(2)
(3)

```
EMT ;COND. CODES INCORRECT
DNMB4E: TST R0 ;CHECK DEST. REGISTER
        BEQ DNMB4F
EMT ;DEST. REG. NOT DECREMENTED BY 1
DNMB4F: CMP #52652,(R0) ;CHECK DESTINATION DATA
        BEQ TS144
        EMT ;DEST. DATA MODIFIED

;*****
;TEST 144 TEST DEST MODE 5 W/DOP NON-MODIFYING INST.
;*****
TS144:
        CLR R0 ;R0=0
        CLR (R0) ;LOC 0=0
        BIS #100000,(R0) ;LOC. 0=100000
        BIS #402,R0 ;R0=2
        SCC ;SET ALL COND. CODE BITS
        BIT #100000,B-(R0) ;TRY DOPNM W/MODE 5
        BVS DNMB5A ;BR TO ERROR IF V-BIT SET
        BCC DNMB5A ;BR TO ERROR IF C-BIT CLEAR
        BEQ DNMB5A ;BR TO ERROR IF Z-BIT SET
        BMI DNMB5B
DNMB5A: EMT ;COND. CODES INCORRECT
DNMB5B: CMP #400,R0 ;CHECK DEST. REGISTER
        BEQ DNMB5C
        EMT ;DEST. REGISTER NOT DECREMENTED BY 2
DNMB5C: CMP #100000,B#0 ;CHECK DESTINATION DATA
        BEQ TS145
        EMT ;DEST. DATA INCORRECTLY MODIFIED

;*****
;TEST 145 TEST DEST. MODE 6 W/DOP NON-MODIFYING INST.
;*****
TS145:
        CLR R0 ;R0=0
        CLR (R0) ;LOC 0=0
        BIS #1,(R0) ;LOC. 0=1
        COM R0 ;R0=-1 C-BIT=1
        BIT #1,1(R0) ;TRY DOPNM W/MODE 6
        BEQ DNMB6A ;BR TO ERROR IF Z-BIT SET
        BVS DNMB6A ;BR TO ERROR IF V-BIT SET
        BCC DNMB6A ;BR TO ERROR IF C-BIT CLEAR
        BPL DNMB6B
DNMB6A: EMT ;COND CODES INCORRECT
DNMB6B: CMP #-1,R0 ;CHECK DEST. REGISTER
        BEQ DNMB6C
        EMT ;DEST. REGISTER MODIFIED
DNMB6C: CMP #1,B#0 ;CHECK DEST. DATA
        BEQ TS146
        EMT ;DEST. DATA MODIFIED

;*****
;TEST 146 TEST DEST MODE 7 W/DOP NON-MODIFYING INST.
;*****
```

(2) 010466
8305 010466 005000
8306 010470 005010
8307 010472 052710 125125
8308 010476 052700 000001
8309 010502 132770 000125 000403
8310 010510 102403
8311 010512 100402
8312 010514 103401
8313 010516 001401
(1) 010520
(2) 010520 104000
8314 010522 022700 000001
8315 010526 001401
(2) 010530 104000
8316 010532 022737 125125 000000
8317 010540 001401
(3) 010542 104000

TS146

```
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0 C BIT=0
BIS #125125,(R0) ;LOC. 0=125125
BIS #1,R0 ;RO=1
BITB #125,0403(R0) ;TRY DOPNM W/MODE 7
BVS DNM7A ;BR TO ERROR IF V-BIT SET
BMI DNM7A ;BR TO ERROR IF N-BIT SET
BCS DNM7A ;BR TO ERROR IF C-BIT SET
BEQ DNM7B
```

DNM7A:

```
EMT ;COND. CODES INCORRECT
DNM7B: CMP #1,R0 ;CHECK DEST. REGISTER
BEQ DNM7C
```

DNM7C:

```
EMT ;DESTINATION REGISTER MODIFIED
CMP #125125,040 ;CHECK DEST. DATA
BEQ TS147
EMT ;DEST. DATA INCORRECT
```

: THIS TEST VERIFIES THE MOV DESTINATION MODE 1 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED TO LOC. 0
: USING MOV SRC MODE 0, DEST. MODE 1.
: *****

TEST 147 TEST MOV DESTINATION MODE 1

TS147:

```
CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
COM R0 ;RO=-1
CLR R4 ;R4 POINTS TO LOC. 0
MOV R0,(R4) ;TRY MOVE MODE 0,1
BVS MDM1A ;BR TO ERROR IF V SET
BEQ MDM1A ;BR TO ERROR IF Z SET
BMI MDM1B
```

MDM1A:

```
EMT ;CONDITION CODE NOT CORRECT
```

MDM1B:

```
TST R4
BEQ TS150
```

EMT ;DESTINATION REGISTER INCORRECTLY ALTERED

: THIS TEST VERIFIES THE MOV DESTINATION MODE 2 INSTRUCTION.
: DATA IS SET IN R0 USING SOP INSTRUCTIONS AND THEN MOVED
: TO LOCATION 0 USING MOV SRC MODE 0, DEST. MODE 1.
: *****

TEST 150 TEST MOV DESTINATION MODE 2

TS150:

```
CLR R0 ;RO=0
CLR R1 ;R1=0
CLR (R0) ;LOC.0=0
```

(2) 010544
8326 010544 005000
8327 010546 005010
8328 010550 005100
8329 010552 005004
8330 010554 010014
8331 010556 102402
8332 010560 001401
8333 010562 100401
(1) 010564
(2) 010564 104000
8334 010566 005704
8335 010570 001401
(3) 010572 104000
8336
8337
8338
8339
8340
8341
8342
8343
(2)
(3)
(2) 010574
8344 010574 005000
8345 010576 005001
8346 010600 005010

8347 010602 005110
8348 010604 010120
8349 010606 100402
8350 010610 102401
8351 010612 001401
(1) 010614
(2) 010614 104000
8352 010616 005300
8353 010620 005300
8354 010622 001401
(1) 010624
(2) 010624 104000
8355 010626 005737 000000
8356 010632 001401
(3) 010634 104000

COM (R0) ;LOC. 0=1
MOV R1,(R0) ;TRY MOVE MODE 0,2
BMI MDM2A ;BR TO ERROR IF N SET
BVS MDM2A ;BR TO ERROR IF V SET
BEQ MDM2B
MDM2A: EMT ;CC'S INCORRECT
MDM2B: DEC R0
DEC R0
BEQ MDM2D
MDM2C: EMT ;DESTINATION REGISTER NOT INCREMENTED PROPERLY
MDM2D: TST @R0
BEQ TS151
EMT ;DESTINATION DATA INCORRECT

; THIS TEST VERIFIES DESTINATION MODE 2 W/MOVB INSTS. TWO DIFFERENT MOVB
; INSTRUCTIONS ARE USED TO MOVE A TEST PATTERN FIRST TO BYTE 0 THEN TO BYTE 1.
; *****
; TEST 151 TEST MOV-BYTE DESTINATION MODE 2
; *****
TS151:

(2) 010636
8364 010636 005000
8365 010640 005010
8366 010642 112720 000125
8367 010646 102402
8368 010650 001401
8369 010652 100001
(1) 010654
(2) 010654 104000
8370 010656 022700 000001
8371 010662 001401
(2) 010664 104000
8372 010666 112720 000252
8373 010672 102402
8374 010674 001401
8375 010676 100401
(1) 010700
(2) 010700 104000
8376 010702 022700 000002
8377 010706 001401
(2) 010710 104000
8378 010712 022737 125125 000000
8379 010720 001401
(3) 010722 104000

CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
MOVB @125,(R0) ;TRY DESTINATION MODE 2 W/EVEN BYTE
BVS MBDM2A ;BR TO ERROR IF V SET
BEQ MBDM2A ;BR TO ERROR IF Z SET
BPL MBDM2B
MBDM2A: EMT ;CC'S INCORRECT
MBDM2B: CMP #1,R0
BEQ MBDM2C
EMT ;REGISTER NOT INCREMENTED BY ONE
MBDM2C: MOVB @252,(R0) ;TRY DESTINATION MODE 2 W/ODD BYTE
BVS MBDM2D
BEQ MBDM2D
BMI MBDM2E
MBDM2D: EMT ;CC'S NOT SET CORRECT
MBDM2E: CMP #2,R0
BEQ MBDM2F
EMT ;REGISTER NOT INCREMENTED BY ONE
MBDM2F: CMP #125125,@R0 ;CHECK DATA
BEQ TS152
EMT ;DESTINATION DATA INCORRECT

; THIS TEST VERIFIES MOV DESTINATION MODE 3. R0 IS USED TO PICK UP
; AN ADDRESS AT LOC. 400. LOC 400 POINTS TO LOC. 0 THE EFFECTIVE DEST. ADDR.. ALSO, MOVB
; INST. ARE USED W/ EVEN AND ODD BYTES TO CHECK MOV BYTES INST AND MODE 37 DESTINATIONS.
; *****
; TEST 152 TEST MOV(B) DESTINATION MODE 3

8380
8381
8382
8383
8384
8385
8386
(2)

(3)
(2) 010724
8387 010724 012700 000400
8388 010730 005010
8389 010732 005037 000000
8390 010736 012730 125252
8391 010742 102402
8392 010744 001401
8393 010746 100401
(1) 010750
(2) 010750 104000
8394 010752 022700 000402
8395 010756 001401
(2) 010760 104000
8396 010762 022737 125252 000000
8397 010770 001401
(2) 010772 104000
8398 010774 112737 000125 000000
8399 011002 022737 125125 000000
8400 011010 001401
(2) 011012 104000
8401 011014 112737 000525 000001
8402 011022 022737 052525 000000
8403 011030 001401
(3) 011032 104000
8404
8405
8406
8407
8408
8409
8410
8411
8412
(2)
(3)
(2) 011034
8413 011034 005000
8414 011036 005010
8415 011040 012704 000002
8416 011044 012744 012345
8417 011050 102402
8418 011052 001401
8419 011054 100001
(1) 011056
(2) 011056 104000
8420 011060 005704
8421 011062 001401
(2) 011064 104000
8422 011066 022710 012345
8423 011072 001401
(3) 011074 104000
8424
8425
8426
8427

```
*****  
TS152:  
MOV #400,R0 ;R0=400  
CLR (R0) ;LOC. 400 POINTS TO LOC. 0  
CLR @0 ;LOC. 0=0  
MOV #125252,@(R0)+ ;TRY MOV DESTINATION MODE 2  
BVS MDM3A ;BR TO ERROR IF V SET  
BEQ MDM3A ;BR TO ERROR IF Z SET  
BMI MDM3B  
MDM3A: EMT ;CC'S INCORRECT  
MDM3B: CMP #402,R0 ;CHECK DEST. MODE REGISTER  
BEQ MDM3C  
EMT ;REGISTER NOT INCREMENTED BY 2  
MDM3C: CMP #125252,@0 ;CHECK DESTINATION DATA  
BEQ MDM3D  
EMT ;DESTINATION DATA INCORRECT  
MDM3D: MOVB #125,@0 ;TRY MOVB DESTINATION MODE 2 EVEN BYTE  
CMP #125125,@0 ;CHECK DATA  
BEQ MDM3E  
EMT ;DESTINATION DATA INCORRECT  
MDM3E: MOVB #525,@1 ;TRY MOVB DESTINATION MODE 2 ODD BYTE  
CMP #52525,@0 ;CHECK DATA  
BEQ TS153  
EMT ;  
*****  
; THIS TEST VERIFIES THE MOV DESTINATION MODE 4 INSTRUCTION.  
; SOP INSTRUCTIONS ON R0 ARE USED TO CLEAR TARGET LOCATION 0.  
; R4 IS USED AS THE MODE 4 ADDRESSING REGISTER, AND  
; CONDITIONAL BRANCHES ARE USED TO VERIFY THE DATA.  
*****  
; TEST 153 TEST MOV DESTINATION MODE 4  
*****  
TS153:  
CLR R0 ;R0=0  
CLR (R0) ;LOC 0=0  
MOV #2,R4 ;R4=2  
MOV #12345,-(R4) ;TRY MOV DEST. MODE 4  
BVS MDM4A ;BR TO ERROR IF V-BIT SET  
BEQ MDM4A ;BR TO ERROR IF Z-BIT SET  
BPL MDM4B  
MDM4A: EMT ;CC'S NOT CORRECT  
MDM4B: TST R4 ;CHECK DECREMENTING OF MODE 4 REG.  
BEQ MDM4C  
EMT ;DESTINATION MODE REGISTER NOT DECREMENTED BY 2  
MDM4C: CMP #12345,(R0) ;CHECK DESTINATION DATA  
BEQ TS154  
EMT ;DESTINATION DATA INCORRECT  
*****  
; THIS TEST VERIFIES THE MOVB DESTINATION MODE 4 INSTRUCTION
```

```

8428
8429
8430
8431
8432
8433
(2)
(3)
(2) 011076
8434 011076 005004
8435 011100 005014
8436 011102 012700 000002
8437 011106 112740 125125
8438 011112 020027 000001
8439 011116 001401
(2) 011120 104000
8440 011122 021427 052400
8441 011126 001401
(2) 011130 104000
8442 011132 112740 125125
8443 011136 102402
8444 011140 001401
8445 011142 100001
(1) 011144
(2) 011144 104000
8446 011146 005700
8447 011150 001401
(2) 011152 104000
8448 011154 021427 052525
8449 011160 001401
(3) 011162 104000
8450
8451
8452
8453
8454
8455
8456
8457
8458
8459
8460
(2)
(3)
(2) 011164
8461 011164 005004
8462 011166 005014
8463 011170 012700 000400
8464 011174 012750 004321
8465 011200 102402
8466 011202 001401
8467 011204 100001
(1) 011206
(2) 011206 104000
8468 011210 022700 000376
8469 011214 001401

```

```

; ON BOTH ODD AND EVEN BYTES. SOP INSTRUCTIONS ON R4 ARE
; USED TO CLEAR TARGET LOCATION 0. R0 IS USED AS THE MODE 4
; ADDRESSING REGISTER, AND CMP AND CONDITIONAL BRANCH
; INSTRUCTIONS ARE USED TO VERIFY THE DATA.
;
;.....
;TEST 154 TEST MOVDESTINATION MODE 4
;.....
TS154:
      CLR      R4           ;R4=0
      CLR      (R4)        ;LOC. 0=0
      MOV      #2,R0       ;R0 = 2
      MOVB     #125125,-(R0) ;TRY MOVDEST. MODE 4-ODD BYTE
      CMP      R0,#1       ;CHECK THAT DEST. REG. WAS DECREMENTED
      BEQ      MBDM4A
      EMT
      MBDM4A: CMP      (R4),#52400 ;DESTINATION REG. NOT DECREMENTED BY 1
      BEQ      MBDM4B       ;CHECK DEST. DATA
      EMT
      MBDM4B: MOVB     #125125,-(R0) ;DEST. DATA NOT CORRECT
      BVS      MBDM4C       ;TRY MOVDEST. MODE 4--EVEN BYTE
      BEQ      MBDM4C       ;BR. TO ERROR IF V-BIT SET
      BPL      MBDM4D       ;BR TO ERROR IF Z-BIT SET
      EMT
      MBDM4C: EMT           ;COND. CODES INCORRECT
      MBDM4D: TST      R0     ;CHECK MODE 4 DEST. REGISTER
      BEQ      MBDM4E
      EMT
      MBDM4E: CMP      (R4),#52525 ;DESTINATION REG NOT DECREMENTED BY 1
      BEQ      TS155        ;CHECK DEST. DATA
      EMT                   ;DESTINATION DATA INCORRECT
;.....
;
; THIS TEST VERIFIES THE MOV DESTINATION MODE 5 AND THE MOVDESTINATION
; DESTINATION MODE 5 - EVEN BYTE INSTRUCTIONS. R4 IS A
; POINTER TO TARGET LOCATION 0 AND R0 IS SETUP TO
; POINT TO LOCATION 376 FOR THE MOV, AND LOCATION 404 FOR
; THE MOVDEST INSTRUCTIONS. CMP INSTRUCTIONS ARE USED TO VERIFY
; PROPER ADDRESSING AND DATA.
;
;.....
;TEST 155 TEST MOVDESTINATION MODE 5
;.....
TS155:
      CLR      R4           ;R4=0
      CLR      (R4)        ;LOC. 0 = 0
      MOV      #400,R0     ;R0=400
      MOV      #4321,B-(R0) ;TRY MOV DEST. MODE 5
      BVS      MDMSA       ;BR TO ERROR IF V-BIT SET
      BEQ      MDMSA       ;BR TO ERROR IF Z-BIT SET
      BPL      MDMSB
      EMT
      MDMSA: EMT           ;COND. CODES INCORRECT
      MDMSB: CMP      #376,R0 ;CHECK MODE 5 REG. WAS DECREMENTED
      BEQ      MDMSC

```

(2) 011216 104000
8470 011220 022714 004321
8471 011224 001401
(2) 011226 104000
8472 011230 012700 000406
8473 011234 112750 000377
8474 011240 022700 000404
8475 011244 001401
(2) 011246 104000
8476 011250 022714 177721
8477 011254 001401
(3) 011256 104000

MDM5C: EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
CMP #4321,(R4) ;CHECK DEST. DATA
BEQ MDM5D
EMT ;DEST. DATA INCORRECT
MDM5D: MOV #406,R0 ;RO=406
MOVB #377,B-(R0) ;TRY MOV DEST. MODE 5 - EVEN BYTE
CMP #404,R0 ;CHECK MODE 5 REG.
BEQ MDM5E
EMT ;MODE 5 REGISTER NOT DECREMENTED BY 2
MDM5E: CMP #177721,(R4) ;CHECK DEST. DATA
BEQ T5156
EMT ;DEST. DATA INCORRECT

8478
8479
8480
8481
8482
8483
8484
8485
8486
8487

.....
; THIS TEST VERIFIES THE MOV DESTINATION MODE 6 AND MOVB EVEN BYTE
; DESTINATION MODE 6 INSTRUCTIONS. R0 IS USED TO SETUP TARGET LOC.0
; FOR BOTH TESTS. PATTERNS OF ONES AND ZEROES ARE MOVED INTO LOC.0
; BY MODE 6 INSTRUCTIONS, AND CMP INSTRUCTIONS ARE USED TO VERIFY
; PROPER ADDRESSING AND DATA.
;

(2)
(3)
(2) 011260
8488 011260 005000
8489 011262 005010
8490 011264 005200
8491 011266 012760 052525 177777
8492 011274 102402
8493 011276 001401
8494 011300 100001
(1) 011302
(2) 011302 104000
8495 011304 022700 000001
8496 011310 001401
(2) 011312 104000
8497 011314 022737 052525 000000
8498 011322 001401
(2) 011324 104000
8499 011326 012700 000002
8500 011332 112760 000377 177777
8501 011340 022700 000002
8502 011344 001401
(2) 011346 104000
8503 011350 022737 177525 000000
8504 011356 001401
(3) 011360 104000

.....
; TEST 156 TEST MOV DESTINATION MODE 6
;

T5156: CLR R0 ;R0=0
CLR (R0) ;LOC. 0=0
INC R0 ;R0=1
MOV #052525,-1(R0) ;TRY MOV DEST. MODE 6
BVS MDM6A ;BR TO ERROR IF V-BIT SET
BEQ MDM6A ;BR TO ERROR IF Z-BIT SET
BPL MDM6B

MDM6A: EMT ;COND. CODES INCORRECT
MDM6B: CMP #1,R0 ;CHECK DEST. REGISTER UNALTERED
BEQ MDM6C
EMT ;DEST. REGISTER INCORRECTLY ALTERED
MDM6C: CMP #52525,B#0 ;CHECK DEST. DATA
BEQ MDM6D
EMT ;DEST. DATA INCORRECT
MDM6D: MOV #2,R0 ;RO=2
MOVB #377,-1(R0) ;TRY MOVB DEST. MODE 6
CMP #2,R0 ;CHECK DEST. REGISTER UNALTERED
BEQ MDM6E
EMT ;DEST. REGISTER INCORRECTLY ALTERED
MDM6E: CMP #177525,B#0 ;CHECK DEST. DATA
BEQ T5157
EMT ;DEST. DATA INCORRECT

8505
8506
8507
8508
8509
8510
8511
8512

.....
; THIS TEST VERIFIES THE MOV DESTINATION MODE 7 AND MOVB - ODD BYTE
; DESTINATION MODE 7 INSTRUCTIONS. R4 POINTS TO TARGET LOC.0 AND R0
; IS USED AS THE MODE 7 ADDRESSING REGISTER. CMP INSTRUCTIONS ARE
; USED TO VERIFY PROPER ADDRESSING AND DATA
;

Uf

8513
(2)
(3)
(2) 011362
8514 011362 005004
8515 011364 005014
8516 011366 012700 000403
8517 011372 012770 070707 177777
8518 011400 102402
8519 011402 001401
8520 011404 100001
(1) 011406
(2) 011406 104000
8521 011410 022700 000403
8522 011414 001401
(2) 011416 104000
8523 011420 022737 070707 000000
8524 011426 001401
(2) 011430 104000
8525 011432 112770 107070 000001
8526 011440 022700 000403
8527 011444 001401
(2) 011446 104000
8528 011450 022737 034307 000000
8529 011456 001401
(3) 011460 104000

```
.....  
;TEST 157 TEST MOV DESTINATION MODE 7  
.....  
TS157:  
CLR R4 ;R4=0  
CLR (R4) ;LOC.0=0  
MOV #403,R0 ;R0=403  
MOV #70707,B-1(R0) ;TRY MOV W/DEST MODE 7  
BVS MDM7A ;BR. TO ERROR IF V-BIT SET  
BEQ MDM7A ;BR TO ERROR IF Z-BIT SET  
BPL MDM7B  
  
MDM7A: EMT ;COND. CODES INCORRECT  
MDM7B: CMP #403,R0 ;CHECK DEST. REGISTER  
BEQ MDM7C  
EMT ;DEST. REGISTER INCORRECTLY ALTERED  
MDM7C: CMP #70707,B#0 ;CHECK DEST. DATA  
BEQ MDM7D  
EMT ;DEST. DATA INCORRECT  
MDM7D: MOVB #107070,B1(R0) ;TRY MOVB W/DEST MODE 7--ODD BYTE  
CMP #403,R0 ;CHECK MODE 7 DEST. REG.  
BEQ MDM7E  
EMT ;DEST. DATA INCORRECT  
MDM7E: CMP #34307,B#0 ;CHECK DEST. DATA  
BEQ TS160  
EMT ;DESTINATION DATA INCORRECT
```

8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
(2)
(3)
(2) 011462
8544 011462 012700 011526
8545 011466 014037 011526
8546 011472 064037 011526
8547 011476 144037 011526
8548 011502 154037 011527
8549 011506 024037 011526
8550 011512 001406
(2) 011514
(3) 011514 104000
8551
8552 011516 125252
8553 011520 052652
8554 011522 053125

```
.....  
; THIS TEST VERIFIES MODE 4 DOUBLE OPERAND INSTRUCTIONS.  
; THE TEST USES MODE 4 ADDRESSING WITH REGISTER 0 TO MOVE THRU A  
; TABLE OF OPERANDS. THE TABLE OF OPERANDS AND THE WORK LOCATION IS  
; STORED FOLLOWING THE TEST CODE. A SERIES OF 5 DOP INSTRUCTIONS UTILIZES  
; THE DATA IN THE TABLE TO CYCLE THE WORK LOCATION THRU A SET OF  
; VALUE. THE DATA HAS BEEN CHOSEN TO INSURE THAT NO SINGLE ERROR WILL  
; GO UNDETECTED. WORD AND BYTE INSTRUCTION ACCESSING BOTH EVEN AND  
; ODD ADDRESSES ARE USED IN THE TEST. THE LISTING SHOWS THE  
; EXPECTED INTERMEDIATE RESULT AS EACH INSTRUCTION IS EXECUTED.  
;.....  
;TEST 160 TEST MODE 4 W/ DOP INSTS.  
;.....  
TS160:  
MOV #TBL1,R0 ;INITIALIZE R0  
MOV -(R0),B#TBL1 ;TBL1=125252  
ADD -(R0),B#TBL1 ;TBL1=000377  
BICB -(R0),B#TBL1 ;TBL1=000252  
BISB -(R0),B#TBL1+1 ;TBL1=125252  
CMP -(R0),B#TBL1 ;CHECK RESULT  
BEQ TS161  
  
DOP4: EMT ;RESULT OF MODE 4 INSTS. INCORRECT  
  
125252  
52652  
53125
```

```

8555 011524 125252
8556 011526 000000
8557
8558
8559
8560
8561
8562
8563
8564
8565
8566
8567
(2)
(3)
(2) 011530
8568 011530 012700 011576
8569 011534 015037 011526
8570 011540 065037 011526
8571 011544 145037 011526
8572 011550 155037 011527
8573 011554 025037 011526
8574 011560 001406
(2) 011562
(3) 011562 104000
8575 011564 011516
8576 011566 011520
8577 011570 011521
8578 011572 011522
8579 011574 011524
8580
8581
8582
8583
8584
8585
8586
8587
8588
8589
8590
(2)
(3)
(2) 011576
8591 011576 012700 011522
8592 011602 016037 000002 011526
8593 011610 066037 000000 011526
8594 011616 146037 177777 011526
8595 011624 156037 177776 011527
8596 011632 026037 177774 011526
8597 011640 001401
(3) 011642 104000
8598
8599
8600
8601

```

```

125252
TBL1: 0

```

```

;*****
;
; THIS TEST VERIFIES MODE 5 DOUBLE OPERAND INSTRUCTIONS.
; THE TEST USES AN ADDRESS TABLE STORED FOLLOWING THE TEST CODE.
; THIS TABLE IS SIMPLY A TABLE OF ADDRESS POINTERS WHICH ADDRESS
; THE DATA TABLE USED IN THE PREVIOUS TEST. THE TEST IS IDENTICAL TO
; THE PREVIOUS TEST EXCEPT THE DATA IS REFERENCED USING THIS ADDRESS
; TABLE AND MODE 5 ADDRESSING. (SEE PREVIOUS TEST).
;*****

```

```

;TEST 161 TEST MODE 5 W/ DOP INSTS.
;*****

```

```

TS161:
MOV @TBL2+2,RO ;INITIALIZE RO
MOV @-(RO),@TBL1 ;TBL1=125252
ADD @-(RO),@TBL1 ;TBL1=000377
BICB @-(RO),@TBL1 ;TBL1=000252
BISB @-(RO),@TBL1+1 ;TBL1=125252
CMP @-(RO),@TBL1 ;CHECK RESULT
BEQ TS162

```

```

DOP5:
EMT ;RESULT OF MODE 5 INSTS. INCORRECT

```

```

TBL1-10
TBL1-6
TBL1-5
TBL1-4
TBL2: TBL1-2

```

```

;*****
;
; THIS TEST VERIFIES MODE 6 DOUBLE OPERAND INSTRUCTIONS.
; IT USES THE SAME DATA AS THAT USED IN THE MODE 4 TESTS.
; THIS TIME THE DATA IS ACCESSED USING MODE 6. RO IS SET
; TO POINT TO THE MIDDLE OF THE TABLE. THE TABLE IS ACCESSED FROM
; BOTTOM TO TOP BY VARYING THE OFFSET IN THE MODE 6 INSTRUCTIONS.
; THE DATA RESULTS ARE IDENTICAL TO THOSE EXPECTED IN THE MODE 4
; TESTS.
;*****

```

```

;TEST 162 TEST MODE 6 W/ DOP INSTS.
;*****

```

```

TS162:
MOV @TBL1-4,RO ;INITIALIZE RO
MOV 2(RO),@TBL1 ;TBL1=125252
ADD 0(RO),@TBL1 ;TBL1=000377
BICB -1(RO),@TBL1 ;TBL1=000252
BISB -2(RO),@TBL1+1 ;TBL1=125252
CMP -4(RO),@TBL1 ;CHECK RESULT
BEQ TS163

```

```

EMT ;RESULT OF MODE 6 INSTS. INCORRECT

```

```

;*****
;
; THIS TEST VERIFIES MODE 7 DOUBLE OPERAND INSTRUCTIONS.
; THIS TEST USES THE SAME ADDRESS TABLE AND DATA TABLE USED BY

```


FE

8602
8603
8604
8605
8606
8607
8608
(2)
(3)
(2) 011644
8609 011644 012700 011570
8610 011650 017037 000004 011526
8611 011656 067037 000002 011526
8612 011664 147037 000000 011526
8613 011672 157037 177776 011527
8614 011700 027037 177774 011526
8615 011706 001401
(3) 011710 104000
8616
8617
8618
8619
8620
8621
8622
8623
8624
(2)
(3)
(2) 011712
8625 011712 012700 125252
8626 011716 000261
8627 011720 006100
8628 011722 102004
8629 011724 103003
8630 011726 022700 052525
8631 011732 001401
(1) 011734
(2) 011734 104000
8632 011736 012700 125252
8633 011742 000261
8634 011744 106100
8635 011746 102004
8636 011750 103003
8637 011752 022700 125125
8638 011756 001401
(2) 011760
(3) 011760 104000
8639
8640
8641
8642
8643
8644
8645
8646

;THE MODE 5 TESTS. THIS TIME THE DATA IS ACCESSED USING MODE 7.
;RO IS SET TO POINT TO THE MIDDLE OF THE ADDRESS TABLE IN THE MODE 5
;TEST. THE TABLE IS ACCESSED FROM BOTTOM TO TOP BY VARYING THE OFFSET
;IN THE MODE 7 INSTRUCTIONS. THE DATA RESULTS ARE IDENTICAL TO
;THOSE EXPECTED IN THE MODE 5 TESTS.

;
;*****
;TEST 163 TEST MODE 7 W/ DOP INSTS.
;*****

TS163:
MOV #TBL2-4,RO ;INITIALIZE RO
MOV #4(RO),#TBL1 ;TBL1=125252
ADD #2(RO),#TBL1 ;TBL1=000377
BICB #0(RO),#TBL1 ;TBL1=000252
BISB #2(RO),#TBL1+1 ;TBL1=125252
CMP #4(RO),#TBL1 ;CHECK RESULT
BEQ TS164
EMT ;RESULT OF MODE 7 INSTS INCORRECT

;*****

; THIS TEST VERIFIES THE ROTATE MODE 0 INSTRUCTIONS.
;RO IS LOADED WITH A DATA PATTERN, THE C-BIT IS LOADED, AND
;AN ROL INSTRUCTION IS EXECUTED WITH MODE 0. THE OPERATION IS CHECKED
;BY TESTING THE RESULTING DATA AND THE STATE OF THE C AND V BITS.
;NEXT, THE SAME PROCEDURE IS EXECUTED TO TEST MODE 0 BYTE INSTRUCTIONS.

;
;*****
;TEST 164 TEST ROTATE INSTRUCTIONS OF MODE 0
;*****

TS164:
MOV #125252,RO ;INITIALIZE DATA
SEC ;SET C-BIT
ROL RO ;TRY ROL W/ MODE 0
BVC RTOA ;CC=0011
BCC RTOA
CMP #052525,RO ;CHECK DATA
BEQ ROTOB
RTOA:
EMT ;ROL MODE 0 FAILED
ROTB: MOV #125252,RO ;INITIALIZE DATA
SEC ;SET C-BIT
ROLB RO ;TRY ROL W/ MODE 0 EVEN BYTE
BVC ROTOC ;CC=0011
BCC ROTOC
CMP #125125,RO ;CHECK DATA
BEQ ROTOC
ROTC:
EMT ;ROLB MODE 0 FAILED

;*****

; THIS TEST VERIFIES THE ROTATE MODE 1 INSTRUCTIONS.
;THE DATA TO BE ROTATED IS IN LOC 0. RO IS USED AS THE
;ADDRESSING REGISTER. THE C-BIT IS LOADED AND AN ROL IS EXECUTED.
;THE RESULTS ARE CHECKED BY COMPARING THE DATA RESULTS AND TESTING
;THE C AND V BITS. THIS PROCEDURE IS THEN REPEATED TWICE MORE

(3f)

8647
8648
8649
(2)
(3)
(2) 011762
8650 011762 005000
8651 011764 012710 052525
8652 011770 000241
8653 011772 006110
8654 011774 102005
8655 011776 103404
8656 012000 023727 000000 125252
8657 012006 001401
(1) 012010
(2) 012010 104000
8658 012012 000261
8659 012014 012710 125252
8660 012020 106110
8661 012022 102005
8662 012024 103004
8663 012026 022737 125125 000000
8664 012034 001401
(1) 012036
(2) 012036 104000
8665 012040 012710 125252
8666 012044 005000
8667 012046 005200
8668 012050 000261
8669 012052 106110
8670 012054 102005
8671 012056 103004
8672 012060 022737 052652 000000
8673 012066 001401
(2) 012070
(3) 012070 104000
8674
8675
8676
8677
8678
8679
8680
8681
8682
(2)
(3)
(2) 012072
8683 012072 005000
8684 012074 012710 173737
8685 012100 000241
8686 012102 006120
8687 012104 103007
8688 012106 022737 167676 000000
8689 012114 001003
8690 012116 005300

```
; TO TEST THE BYTE ROTATES. FIRST ON BYTE 0, THEN ON BYTE 1.  
;  
;.....  
;TEST 165 TEST ROTATE INSTRUCTIONS W/ MODE 1  
;.....  
TS165:  
CLR RO ;POINT TO LOC. 0  
MOV #52525,(RO) ;INITIALIZE DATA  
CLC ;CLEAR C-BIT  
ROL (RO) ;TRY ROL W/ MODE 1  
BVC ROT1A ;CC=1010  
BCS ROT1A  
CMP #0,#125252 ;CHECK RESULT  
BEQ ROT1B  
ROT1A: EMT ;ROL MODE 1 FAILED  
ROT1B: SEC  
MOV #125252,(RO) ;INITIALIZE DATA  
ROLB (RO) ;TRY ROLB W/ MODE 1 EVEN BYTE  
BVC ROT1C ;CC=1011  
BCC ROT1C  
CMP #125125,#0 ;TEST RESULT  
BEQ ROT1D  
ROT1C: EMT ;ROLB W/ MODE 1 EVEN BYTE FAILED  
ROT1D: MOV #125252,(RO)  
CLR RO ;POINT TO ODD BYTE  
INC RO  
SEC ;SET C-BIT  
ROLB (RO) ;TRY ROLB W/ MODE 1 ODD BYTE  
BVC ROT1E ;CC=0011  
BCC ROT1E  
CMP #052652,#0 ;CHECK DATA  
BEQ TS166  
ROT1E: EMT ;ROLB W/ MODE 1 ODD BYTE FAILED  
;  
;.....  
; THIS TEST VERIFIES MODE 2 ROTATE INSTRUCTIONS.  
; THE SAME PROCEDURE AS IN THE OTHER ROTATE TESTS ARE USED. RO  
; IS USED AS THE ADDRESSING REGISTER AND IS CHECKED FOR PROPER  
; INCREMENTING. BYTE INSTRUCTIONS ARE ALSO CHECKED.  
;  
;.....  
;TEST 166 TEST ROTATE INSTRUCTIONS W/ MODE 2  
;.....  
TS166:  
CLR RO ;POINT TO LOC 0  
MOV #173737,(RO) ;INITIALIZE DATA  
CLC ;CLEAR C-BIT  
ROL (RO); ;TRY ROL W/ MODE 2  
BCC ROT2A ;CHECK C-BIT  
CMP #167676,#0 ;CHECK DATA  
BNE ROT2A ;BRANCH IF RESULT INCORRECT  
DEC RO ;TEST RO
```

8691 012120 005300 DEC R0
8692 012122 001401 BEQ ROT2B
(1) 012124 ROT2A:
(2) 012124 104000 EMT ;ROL W/ MODE 2 FAILED
8693 012126 005000 ROT2B: CLR R0 ;POINT TO LOC 0
8694 012130 012710 004040 MOV #4040,(R0) ;INITIALIZE DATA
8695 012134 000241 CLC ;CLEAR C-BIT
8696 012136 106120 ROLB (R0)+ ;TRY ROLB W/ MODE 2 EVEN BYTE
8697 012140 103406 BCS ROT2C ;CHECK C-BIT
8698 012142 022737 004100 000000 CMP #4100,#0 ;CHECK DATA
8699 012150 001002 BNE ROT2C ;BRANCH IF DATA INCORRECT
8700 012152 005300 DEC R0 ;CHECK R0
8701 012154 001401 BEQ ROT2D
(1) 012156 ROT2C:
(2) 012156 104000 EMT ;ROLB W/ MODE 2 EVEN BYTE FAILED
8702 012160 005000 ROT2D: CLR R0 ;POINT TO LOC 0
8703 012162 012710 004040 MOV #4040,(R0) ;INITIALIZE DATA
8704 012166 005200 INC R0 ;POINT TO ODD BYTE OF DATA
8705 012170 000261 SEC ;SET C-BIT
8706 012172 106120 ROLB (R0)+ ;TRY ROL W/ MODE 2 ODD BYTE
8707 012174 103407 BCS ROT2E ;CHECK C-BIT
8708 012176 022737 010440 000000 CMP #10440,#0 ;CHECK DATA
8709 012204 001003 BNE ROT2E ;BRANCH IF DATA INCORRECT
8710 012206 005300 DEC R0 ;CHECK R0
8711 012210 005300 DEC R0
8712 012212 001401 BEQ TS167
(2) 012214 ROT2E:
(3) 012214 104000 EMT ;ROLB W/ MODE 2 ODD BYTE FAILED

: THIS TEST VERIFIES MODE 3 ROTATE INSTRUCTIONS.
: THIS TEST USES THE SAME PROCEDURES AS IN THE OTHER ROTATE
: TESTS. THE DATA IS STORED IN LOC. 0 AND IS ADDRESSED USING
: MODE 37. BYTE ADDRESSING IS ALSO CHECKED FOR EVEN AND ODD BYTES.
: *****

TEST 167 TEST ROTATE INSTRUCTIONS /W MODE 3

TS167:
8722 012216 012737 052525 000000 MOV #52525,#0 ;INITIALIZE DATA IN LOC 0
8723 012224 000261 SEC ;SET C-BIT
8724 012226 006137 000000 ROL #0 ;TRO ROL W/ MODE 3
8725 012232 103404 BCS ROT3A ;CHECK C-BIT
8726 012234 022737 125253 000000 CMP #125253,#0 ;CHECK DATA
8727 012242 001401 BEQ ROT3B
(1) 012244 ROT3A:
(2) 012244 104000 EMT ;ROL W/ MODE 3 FAILED
8728 012246 012737 125252 000000 ROT3B: MOV #125252,#0 ;INITIALIZE DATA
8729 012254 000241 CLC ;CLEAR C-BIT
8730 012256 106137 000000 ROLB #0 ;TRY ROL W/ MODE 3 EVEN BYTE
8731 012262 103004 BCC ROT3C ;CHECK C-BIT
8732 012264 023727 000000 125124 44: CMP #0,#125124 ;CHECK DATA
8733 012272 001401 BEQ ROT3D
(1) 012274 ROT3C:
(2) 012274 104000 EMT ;ROL W/ MODE 3 EVEN BYTE FAILED

8734 012276 012737 125252 000000
 8735 012304 000261
 8736 012306 106137 000001
 8737 012312 103004
 8738 012314 022737 052652 000000
 8739 012322 001401
 (2) 012324
 (3) 012324 104000

ROT3D: MOV #125252,B#0 ;INITIALIZE DATA IN LOC. 0
 SEC ;SET C-BIT
 ROLB #01 ;TRY ROL W/ MODE 3 ODD BYTE
 BCC ROT3E ;CHECK C-BIT
 CMP #052652,B#0 ;CHECK DATA
 BEQ TS170
 ROT3E: EMT ;ROL W/ MODE 3 ODD BYTE FAILED

8740
8741
8742
8743
8744
8745
8746
8747
8748
8749

```

;*****
;
; THIS TEST VERIFIES MODE 4 ROTATE INSTRUCTIONS. THE DATA IS
; STORED IN LOC. 0. RO IS SET TO 2 AND THE CARRY IS SET. AN ROL MODE 4
; IS USED TO ROTATE LOCATION 0 USING RO. THE DATA IS CHECKED
; AND THE C AND V BITS ARE TESTED. THE PROPER DECREMENTING OF
; RO IS VERIFIED.
;
;*****

```

(2)
(3)
(2) 012326
8750 012326 012737 070707 000000
8751 012334 012700 000002
8752 012340 000261
8753 012342 006140
8754 012344 103406
8755 012346 022737 161617 000000
8756 012354 001002
8757 012356 005700
8758 012360 001401
(2) 012362
(3) 012362 104000

```

;*****
; TEST 170 TEST MODE 4 W/ ROTATE INSTRUCTIONS
;*****
TS170:
MOV #070707,B#0 ;INITIALIZE DATA IN LOC. 0
MOV #2,RO ;INITIALIZE RO AS POINTER
SEC ;SET C-BIT
ROL -(RO) ;TRY ROL W/ MODE 4
BCS ROT4 ;CHECK C-BIT
CMP #161617,B#0 ;CHECK DATA
BNE ROT4 ;BRANCH IF DATA INCORRECT
TST RO ;CHECK MODE 4 REGISTER
BEQ TS171
ROT4: EMT ;ROL MODE 4 FAILED

```

8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769

```

;*****
;
; THIS TEST VERIFIES MODE 5 ROTATE INSTRUCTIONS.
; THE DATA IS STORED IN A WORK LOCATION (ROTX) AT THE END OF THE
; TEST CODE. LOC. 0 IS LOADED WITH THE ADDRESS OF THE DATA (ROTX).
; RO IS SET TO 2. THE CARRY IS CLEARED AND A MODE 5 ROL
; IS EXECUTED USING RO AS AN ADDRESSING REGISTER. THE DATA IS
; CHECKED, THE C AND V BITS TESTED, AND RO CHECKED FOR PROPER
; DECREMENTING.
;
;*****

```

8770
(2)
(3)
(2) 012364
8771 012364 012737 012430 000000
8772 012372 012700 000002
8773 012376 012767 107070 000024
8774 012404 000241
8775 012406 006150
8776 012410 103006
8777 012412 022737 016160 012430
8778 012420 001002
8779 012422 005700

```

;*****
; TEST 171 TEST MODE 5 W/ ROTATE INSTRUCTIONS
;*****
TS171:
MOV #ROTX,B#0 ;MOVE POINTER TO LOC. 0
MOV #2,RO ;SET MODE 5 REG. TO LOC. 0
MOV #107070,ROTX ;INITIALIZE DATA
CLC ;CLEAR C-BIT
ROL B-(RO) ;TRY ROL W/ MODE 5
BCC ROT5 ;CHECK C-BIT
CMP #016160,B#ROTX ;CHECK DATA
BNE ROT5 ;BRANCH IF DATA INCORRECT
TST RO ;CHECK MODE 5 REGISTER

```

J6

8780 012424 001402
 (2) 012426
 (3) 012426 104000
 8781 012430 000000
 8782
 8783
 8784
 8785
 8786
 8787
 8788
 8789
 8790
 (2)
 (3)
 (2) 012432
 8791 012432 012737 125252 012430
 8792 012440 000261
 8793 012442 006167 177762
 8794 012446 103004
 8795 012450 022737 052525 012430
 8796 012456 001401
 (2) 012460
 (3) 012460 104000
 8797
 8798
 8799
 8800

BEQ TS172
 ROT5:
 EMT ;ROL MODE 5 FAILED
 ROTX: 0

.....
 ;
 ; THIS TEST VERIFIES MODE 6 ROTATE INSTRUCTIONS.
 ; IT USES THE SAME PROCEDURE AS THE ABOVE TEST EXCEPT THE
 ; ROTATE INSTRUCTION USES MODE 6 ADDRESSING WITH REGISTER 7.
 ; THE DATA IS STILL OPERATED ON IN LOC. ROTX (SEE PREVIOUS TEST).
 ;

.....
 ; TEST 172 TEST MODE 6 W/ ROTATE INSTRUCTIONS
 ;

TS172:
 MOV #125252, @ROTX ;INITIALIZE DATA
 SEC ;SET C-BIT
 ROL ROTX ;TRY ROL W/ MODE 6
 BCC ROT6 ;CHECK C-BIT
 CMP #52525, @ROTX ;CHECK DATA
 BEQ TS173

ROT6:
 EMT ;ROL W/ MODE 6 FAILED

.....
 ;
 ; THIS TEST VERIFIES MODE 7 ROTATE INSTRUCTIONS.

8802
8803
8804
8805
8806
(2)
(3)
(2) 012462
8807 012462 012737 052525 012430
8808 012470 012737 012430 012520
8809 012476 000241
8810 012500 006177 000014
8811 012504 103404
8812 012506 023727 012430 125252
8813 012514 001402
(2) 012516
(3) 012516 104000
8814 012520 000000
8815
8816
8817
8818
8819
8820
8821
8822
8823
8824
(2)
(3)
(2) 012522
8825 012522 012700 177400
8826 012526 000300
8827 012530 100401
(2) 012532 104000
8828 012534 022700 000377
8829 012540 001401
(3) 012542 104000
8830
8831
8832
8833
8834
8835
8836
8837
8838
(2)
(3)
(2) 012544
8839 012544 012737 125652 000000
8840 012552 005000
8841 012554 000310
8842 012556 022737 125253 000000
8843 012564 001401
(3) 012566 104000

; THE DATA IS SET IN LOC. ROTX, (SEE PREVIOUS TEST) THE ROL INSTRUCTION
; ADDRESSES IT INDIRECTLY USING MODE 7 AND INDIRECT ADDRESS LOCATION
; (ROTXAD) FOLLOWING THE TEST CODE.

; TEST 173 TEST MODE 7 W/ ROTATE INSTRUCTIONS

TS173:
MOV #52525, @ROTX ; INITIALIZE DATA
MOV @ROTX, @ROTXAD ; INITIALIZE ADDRESS POINTER
CLC ; CLEAR C-BIT
ROL @ROTXAD ; TRY ROL W/ MODE 7
BCS ROT7 ; CHECK C-BIT
CMP @ROTX, #125252 ; CHECK DATA
BEQ TS174

ROT7: EMT ; ROL W/ MODE 7 FAILED
ROTXAD: 0

; THIS TEST VERIFIES MODE 0 SWAB INSTRUCTION. RO IS SET TO
; 177400. A SWAB MODE 0 IS EXECUTED AND THE CONDITIONAL BRANCH
; IS USED TO CHECK THE SIGN OF THE RESULT. ALSO, A COMPARISON
; IS MADE TO CHECK THE DATA RESULTS.

; TEST 174 TEST MODE 0 W/ SWAB INST.

TS174:
MOV #177400, RO ; MOVE TEST PATTERN TO RO
SWAB RO ; TRY SWAB MODE 0
BMI SBO
EMT ; SWAB DID NOT SET CC'S CORRECT
SBO: CMP #377, RO ; CHECK RESULT
BEQ TS175
EMT ; RESULT OF SWAB MODE 0 FAILED

; THIS TEST VERIFIES MODE 1 SWAB INSTRUCTION. THE TEST
; PATTERN IS MOVED TO LOC 0. RO IS CLEARED AND USED AS THE ADDRESSING
; REGISTER IN THE MODE 1 SWAB. THE DATA RESULTS ARE CHECKED WITH
; A COMPARE.

; TEST 175 TEST MODE 1 W/ SWAB INST

TS175:
MOV #125652, @0 ; MOVE TEST PATTERN TO LOC. 0
CLR RO ; RO=0
SWAB (RO) ; TRY SWAB MODE 1
CMP #125253, @0 ; CHECK RESULT
BEQ TS176
EMT ; RESULT OF SWAB MODE 1 FAILED

```

8844
8845
8846
8847
8848
8849
8850
8851
8852
8853
(2)
(3)
(2) 012570
8854 012570 012737 125152 000000
8855 012576 005000
8856 012600 000320
8857 012602 022737 065252 000000
8858 012610 001401
(2) 012612 104000
8859 012614 162700 000002
8860 012620 001401
(3) 012622 104000
8861
8862
8863
8864
8865
8866
8867
8868
8869
8870
(2)
(3)
(2) 012624
8871 012624 012737 000377 000000
8872 012632 000337 000000
8873 012636 022737 177400 000000
8874 012644 001401
(3) 012646 104000
8875
8876
8877
8878
8879
8880
8881
8882
8883
8884
(2)
(3)
(2) 012650
8885 012650 012737 125652 000000
8886 012656 012700 000002
8887 012662 000340

```

```

;*****
;
; THIS TEST VERIFIES MODE 2 SWAB INSTRUCTION. THE TEST
; PATTERN IS MOVED TO LOC 0. RO IS CLEARED AND USED AS THE MODE
; 2 ADDRESSING REGISTER. THE RESULTS ARE CHECKED WITH A COMPARE.
; RO IS CHECKED FOR PROPER DECREMENTING.
;*****

```

```

;TEST 176 TEST MODE 2 W/ SWAB INST
;*****

```

```

TS176:
MOV #125152,RO ;MOVE TEST PATTERN TO LOC. 0
CLR RO ;RO=0
SWAB (RO); ;TRY SWAB MODE 2
CMP #65252,RO ;CHECK RESULT
BEQ S82
EMT ;RESULT OF SWAB MODE 0 FAILED
S82: SUB #2,RO ;CHECK EFFECT OF REG.
BEQ TS177
EMT ;REGISTER VALUE INCORRECT

```

```

;*****
;
; THIS TEST VERIFIES MODE 3 SWAB INSTRUCTION. THE TEST
; PATTERN IS MOVED TO LOC 0. A MODE 3 SWAB INSTRUCTION IS EXECUTED
; USING R7 AS THE ADDRESSING REGISTER. A COMPARE VERIFIES THE
; DATA RESULTS.
;*****

```

```

;TEST 177 TEST MODE 3 W/SWAB INST.
;*****

```

```

TS177:
MOV #377,RO ;MOVE TEST PATTERN TO LOC. 0
SWAB RO ;TRY SWAB W/ MODE 3
CMP #177400,RO ;CHECK RESULT
BEQ TS200
EMT ;RESULT OF SWAB INCORRECT

```

```

;*****
;
; THIS TEST VERIFIES MODE 4 SWAB INSTRUCTIONS. THE DATA
; IS MOVED TO LOC 0. RO IS SET TO 2 AND USED AS THE MODE 4 ADDRESSING
; REGISTER. THE DATA IS CHECKED WITH A COMPARE AND RO IS CHECKED
; FOR PROPER DECREMENTING.
;*****

```

```

;TEST 200 TEST MODE 4 W/ SWAB INST
;*****

```

```

TS200:
MOV #125652,RO ;MOVE TEST PATTERN TO LOC. 0
MOV #2,RO ;SET UP REGISTER POINTER
SWAB -(RO) ;TRY SWAB MODE 4

```

8888 012664 022737 125253 000000
 8889 012672 001401
 (2) 012674 104000
 8890 012676 005700
 8891 012700 001401
 (3) 012702 104000

CMP #125253,800 ;CHECK RESULT
 BEQ SB4
 EMT ;RESULT OF SWAB INCORRECT
 ;CHECK EFFECT ON REG.
 SB4: TST R0
 BEQ TS201
 EMT ;REGISTER VALUE INCORRECT

8892
 8893
 8894
 8895
 8896
 8897
 8898
 8899
 8900
 8901
 8902
 8903
 (2)
 (3)
 (2) 012704

 ; THIS TEST VERIFIES MODE 5 SWAB INSTRUCTION. THE TEST USES
 ; TWO LOCATIONS FOLLOWING THE TEST CODE. SB5X HOLDS THE DATA;
 ; SB5XAD IS A POINTER TO THE DATA LOCATION. THE DATA IS MOVED TO
 ; SB5X AND R0 IS SET TO TWO PLUS THE ADDRESS OF SB5XAD. FOLLOWING
 ; THE MODE 5 SWAB SB5X IS CHECKED FOR THE PROPER DATA. R0 IS
 ; CHECKED TO SEE THAT IT WAS DECREMENTED PROPERLY.
 ;
 ;*****
 ;TEST 201 TEST MODE 5 W/ SWAB INST.
 ;*****
 TS201:

8904 012704 012700 012746
 8905 012710 012767 125125 000024
 8906 012716 000350
 8907 012720 022767 052652 000014
 8908 012726 001401
 (2) 012730 104000
 8909 012732 020027 012744
 8910 012736 001403
 (2) 012740
 (3) 012740 104000
 8911 012742 000000
 8912 012744 012742

MOV #SB5XAD+2,R0 ;SET UP POINTER TO WORK LOCATION
 MOV #125125,SB5X ;MOVE PATTERN TO WORK LOCATION
 SWAB 8-(R0) ;TRY SWAB MODE 5
 CMP #52652,SB5X ;CHECK RESULT
 BEQ SB5A
 EMT ;RESULT OF SWAB INCORRECT
 ;CHECK RESULT OF REG.
 SB5A: CMP R0,#SB5XAD
 BEQ TS202
 SB5: EMT ;REGISTER VALUE INCORRECT
 SB5X: 0 ;WORK LOCATION
 SB5XAD: SB5X

8913
 8914
 8915
 8916
 8917
 8918
 8919
 8920
 8921
 8922
 8923
 8924
 (2)
 (3)
 (2) 012746

 ; THIS TEST VERIFIES MODE 6 SWAB INSTRUCTION. THIS TEST
 ; USES A WORK LOCATION (SB6X) FOLLOWING THE TEST CODE. TEST DATA
 ; IS LOADED INTO THE WORK LOCATION. R0, THE ADDRESSING REGISTER
 ; IS LOADED WITH 6 LESS THEN THE ADDRESS OF THE WORK LOCATION.
 ; THE MODE 6 SWAB IS EXECUTED WITH A +6 OFFSET. THE DATA IS
 ; VERIFIED WITH A COMPARE.
 ;
 ;*****
 ;TEST 202 TEST MODE 6 W/ SWAB INST.
 ;*****
 TS202:

8925 012746 012767 125125 000022
 8926 012754 012700 012770
 8927 012760 000360 000006
 8928 012764 022760 052652 000006
 8929 012772 001402
 (2) 012774
 (3) 012774 104000
 8930 012776 000000

MOV #125125,SB6X ;MOVE PATTERN TO WORK LOCATION
 MOV #SB6X-6,R0 ;MOVE OFFSET POINTER TO R0
 SWAB 6(R0) ;TRY SWAB W/ MODE 6
 CMP #52652,6(R0) ;CHECK RESULT
 BEQ TS203
 SB6: EMT ;RESULT OF SWAB INCORRECT
 SB6X: 0 ;WORK LOCATION

NE

```

8931
8932
8933
8934
8935
8936
8937
8938
8939
8940
8941
8942
8943
(2)
(3)
(2) 013000
8944 013000 012767 177400 000022
8945 013006 012700 012740
8946 013012 000370 000072
8947 013016 027027 000072 000377
8948 013024 001403
(2) 013026
(3) 013026 104000
8949 013030 000000
8950 013032 013030
8951
8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978
8979
8980
8981

```

```

;*****
;
; THIS TEST VERIFIES MODE 7 SWAB INSTRUCTION. THIS TEST
; USES TWO LOCATIONS FOLLOWING THE TEST CODE: A WORK LOCATION
; (SB7X) AND A POINTER TO THE WORK LOCATION (SB7XAD). DATA IS MOVED
; TO THE WORK LOCATION. R0 IS LOADED WITH 72 LESS THAN THE ADDRESS
; OF THE ADDRESS POINTER. THE DATA IS SWAB'ED USING A MODE 7
; INSTRUCTION WITH AN OFFSET OF +72. THE DATA IS VERIFIED WITH A
; COMPARE.
;
;*****
; TEST 203 TEST MODE 7 W/ SWAB INST.
;*****
TS203:
MOV #177400,SB7X ;MOVE PATTERN TO WORK LOCATION
MOV #SB7XAD-72,R0 ;MOVE OFFSET POINTER TO R0
SWAB 872(R0) ;TRY SWAB MODE 7
CMP 872(R0),#377 ;CHECK RESULTS
BEQ TS204

SB7:
EMT ;RESULT OF SWAB INCORRECT
SB7X: 0 ;WORK LOCATION
SB7XAD: SB7X ;POINTER TO WORK LOCATION

```

```

;*****
;
; THIS TEST VERIFIES ALL LEGAL MODES OF THE JMP INSTRUCTION.
; BECAUSE OF THE NATURE OF THE INSTRUCTION UNDER TEST, THIS TEST
; UTILIZES SEVERAL DIFFERENT TECHNIQUES. THE CODE IS NOT EXECUTED
; IN A LINEAR FASHION. THE DIFFERENT MODES ARE EXECUTED IN ORDER
; FROM 1-7; HOWEVER, THE CODE IS ARRANGED SO THAT CONTROL LEAP
; FROGS THRU THE TEST CODE. THE ORDER OF APPEARANCE OF THE CODE
; IS:
;
; JMP MODE 1
; JMP MODE 3
; JMP MODE 2
; JMP MODE 4
; JMP MODE 6
; JMP MODE 5
; JMP MODE 7
;
; AN INTERNAL SEQUENCE TEST (JMPSEQ) IS USED TO INSURE THAT THE
; JUMPS ARE OCCURRING IN THE PROGRAMMED SEQUENCE.
; THE TEST IS MADE UP OF SEVERAL BLOCKS OF CODE. EACH CODE
; BEGINS WITH A LABEL WHICH INDICATES THE MODE BEING EXECUTED IN
; THAT BLOCK. A SIMPLE PROCEDURE IS FOLLOWED IN EACH BLOCK. FOR
; EXAMPLE THE CODE BEGINNING AT JMP3 WILL FIRST COMPARE THE RESULTS
; OF THE PREVIOUS MODE 2 JMP. (ANY REGISTER CHANGES ARE VERIFIED
; AND THE SEQUENCE CHECK IS MADE). THEN THE REGISTERS ARE SETUP
; FOR A MODE 3 JUMP TO THE NEXT TEST BLOCK (HERE, JMP4), THE SEQUENCE
; CHECKER IS UPDATED AND THE JUMP IS EXECUTED.
; IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
; DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT
; THEN THE ERROR DETECTED WAS A MODE FAILURE (E.G. FAILURE OF THE

```

```

8982 ;REGISTER TO BE INCREMENTED IN MODE 2 JUMP.)
8983 ;
8984 ;*****
(2) ;TEST 204 TEST THE JMP INSTRUCTION IN ALL MODES
(3) ;*****
(2) TS204:
8985 013034 005067 000240 CLR JMPSEQ ;ESTABLISH A SEQUENCE CHECKER
8986 013040 012700 013104 MOV @JMP2,RO ;SET RO=JUMP TARGET
8987 013044 000110 JMP (RO) ;TRY JMP MODE 1
8988 013046 022700 013050 JMP3: CMP @.2,RO ;CHECK RESULT OF MODE 2 JUMP
8989 013052 001401 BEQ JMP3A
(2) 013054 104000 EMT ;REGISTER VALUE AFTER JMP MODE 2 INCORRECT
8990 013056 026727 000216 000001 JMP3A: CMP JMPSEQ,#1 ;MAKE SURE JMPs ARE IN SEQUENCE: JMPSEQ=1?
8991 013064 001401 BEQ JMP3B
(2) 013066 104000 EMT ;SHOULD BE HERE FROM JMP MODE 2 ONLY
8992 013070 012700 013102 JMP3B: MOV @IJMP4,RO ;POINT RO TO INDIRECT JMP ADDR.
8993 013074 005267 000200 INC JMPSEQ ;UPDATE SEQUENCE CHECKER
8994 013100 000130 JMP B(RO). ;TRY JMP MODE 3
8995 013102 013126 IJMP4: JMP4 ;ADDRESS INDIRECT JUMP
8996
8997 013104 005767 000170 JMP2: TST JMPSEQ ;CHECK THAT JMPs ARE IN SEQUENCE: JMPSEQ=0?
8998 013110 001401 BEQ JMP2A
(2) 013112 104000 EMT ;SHOULD BE HERE FROM JMP MODE 1 ONLY
8999 013114 005267 000160 JMP2A: INC JMPSEQ ;UPDATE SEQUENCE CHECKER
9000 013120 012700 013046 MOV @JMP3,RO ;SET RO=JUMP TARGET
9001 013124 000120 JMP (RO). ;TRY A JMP MODE 2 TO "JMP3"
9002 013126 022700 013104 JMP4: CMP @IJMP4+2,RO ;CHECK RESULT OF REGISTER IN MODE 3 JUMP
9003 013132 001401 BEQ JMP4A
(2) 013134 104000 EMT ;REGISTER VALUE AFTER MODE 3 JUMP INCORRECT
9004 013136 022767 000002 000134 JMP4A: CMP #2,JMPSEQ ;CHECK JUMP SEQUENCE: JMPSEQ=2?
9005 013144 001401 BEQ JMP4B
(2) 013146 104000 EMT ;SHOULD BE ONLY FROM MODE 3 JUMP
9006 013150 012700 013212 JMP4B: MOV @JMP5+2,RO ;SET UP POINTER TO JUMP TARGET
9007 013154 005267 000120 INC JMPSEQ ;UPDATE SEQUENCE CHECKER
9008 013160 000140 JMP -(RO) ;TRY JMP MODE 4 TO "JMP4"
9009
9010 013162 022767 000004 000110 JMP6: CMP #4,JMPSEQ ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=4?
9011 013170 001401 BEQ JMP6A
(2) 013172 104000 EMT ;SHOULD BE HERE ONLY FROM MODE 5 JUMP
9012 013174 012700 013634 JMP6A: MOV @JMP7+376,RO ;SET UP OFFSET POINTER TO JUMP TARGET
9013 013200 005267 000074 INC JMPSEQ ;UPDATE JUMP SEQUENCE
9014 013204 000160 177402 JMP -376(RO) ;TRY MODE 6 JUMP
9015
9016 013210 022767 000003 000062 JMP5: CMP #3,JMPSEQ ;CHECK THAT JUMPS ARE IN SEQUENCE: JMPSEQ=3?
9017 013216 001401 BEQ JMP5A
(2) 013220 104000 EMT ;SHOULD ONLY BE HERE FROM MODE 4 JUMP
9018 013222 012700 013236 JMP5A: MOV @IJMP5+2,RO ;SET UP POINTER TO INDIRECT JUMP ADDR.
9019 013226 005267 000046 INC JMPSEQ ;UPDATE JUMP SEQUENCE
9020 013232 000150 JMP B-(RO) ;TRY JMP MODE 5 TO "JMP6"
9021 013234 013162 IJMP5: JMP6 ;INDIRECT ADDRESS POINTER
9022
9023 013236 022767 000005 000034 JMP7: CMP #5,JMPSEQ ;CHECK JUMPS IN SEQUENCE: JMPSEQ=5?
9024 013244 001401 BEQ JMP7A
(2) 013246 104000 EMT ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
9025 013250 012700 013274 JMP7A: MOV @IJMP+10,RO ;SET UP OFFSET POINTER TO INDIRECT ADDR
9026 013254 005267 000020 INC JMPSEQ ;UPDATE JUMP SEQUENCE

```

9027 013260 000170 177770
9028 013264 013266
9029
9030 013266 026727 000006 000006
9031 013274 001402
(3) 013276 104000
9032 013300 000000
9033
9034
9035
9036
9037
9038
9039
9040
9041
9042
9043
9044
9045
9046
9047
9048
9049
9050
(2)
(3)
(2) 013302
9051 013302 000402
9052 013304 000137 013666
9053
9054 013310 012706 001000
9055 013314 012700 013406
9056 013320 005037 013646
9057 013324 005001
9058 013326 005101
9059 013330 004110
9060
9061
9062 013332
(2) 013332 104000
9063
9064 013334 022737 000001 013646
9065 013342 001014
9066 013344 020127 013462
9067 013350 001011
9068 013352 022706 000776
9069 013356 001006
9070 013360 022716 125252
9071 013364 001003
9072 013366 022700 013336
9073 013372 001401
(1) 013374
(2) 013374 104000
9074 013376 005237 013646
9075 013402 004137 013462

JMP B 10(90) ;TRY MODE 7 JUMP
I JMP: JMPCK ;INDIRECT ADDRESS
JMPCK: CMP JMPSEQ,#6 ;CHECK JUMPS IN SEQUENCE: JMPSEQ
BEQ TS205 ;SHOULD ONLY BE HERE FROM MODE 6 JUMP
EMT
JMPSEQ: 0
;.....
; THIS TEST VERIFIES ALL LEGAL MODES OF THE JSR INSTRUCTION.
; THE CONCEPT OF LEAP FROGGING AND SEQUENCE CHECKING (JSRSEQ) IS
; IDENTICAL TO THAT USED IN JMP TEST (SEE PREVIOUS TEST). EACH
; BLOCK OF CODE VERIFIES THE PREVIOUS JSR BY CHECKING THE SEQUENCE,
; CHECKING THAT THE PC WAS SAVED IN THE SPECIFIED REGISTER, CHECKING
; THAT THE SP WAS DECREMENTED, CHECKING THAT THE REGISTER WAS
; SAVED ON THE STACK, AND FINALLY CHECKING THAT ANY MODE ADDRESS
; REGISTER ALTERATIONS (E.G. INCREMENT REGISTER IN MODE 2) WERE
; SUCCESSFUL. R1 IS USED AS THE REGISTER IN ALL JSR INSTRUCTIONS.
; IF A FAILURE OCCURS, THE SEQUENCE CHECKER WILL ASSIST IN
; DETERMINING JUST WHICH MODE FAILED. IF THE SEQUENCE IS CORRECT
; THEN THE ERROR DETECTED WAS A FUNCTIONAL FAILURE (E.G., INCORRECT
; REGISTER SAVED).
;.....
;TEST 205 TEST JSR INSTRUCTION W/ ALL MODES
;.....
TS205:
JSR0: BR JSR1
JMP @JSRCK1
JSR1: MOV @STBOT,R6 ;SET STACK POINTER
MOV @JSR2,R0 ;SET TARGET ADDRESS
CLR @JSRSEQ ;INITIALIZE SEQUENCE CHECKER
CLR R1 ;INITIALIZE R1
COM R1
JSR R1,(R0) ;TRY JSR MODE 1
; TO SCOPE: REPLACE THE MOVE INSTRUCTION (....
; FOLLOWING W/ 774 (....
JSR1A: EMT ;JSR MODE 1 FAILED
JSR3: CMP #1,@ JSRSEQ ;CHECK SEQUENCE: JSRSEQ=1?
BNE JSR3A ;BRANCH IF OUT OF SEQUENCE
CMP R1,@JSR4 ;PROPER PC SAVED?
BNE JSR3A ;BRANCH IF PC WRONG
CMP @STBOT-2,R6 ;STACK POINTER DECREMENTED?
BNE JSR3A ;BRANCH IF SP WRONG
CMP #125252,(R6) ;REG SAVED ON STACK?
BNE JSR3A ;BRANCH IF REG. NOT SAVED
CMP @JSR3+2,R0 ;MODE 2 INCREMENT CORRECT?
BEQ JSR3B
JSR3A: EMT ;JSR MODE 3 MALFUNCTIONED
JSR3B: INC @JSRSEQ ;UPDATE SEQUENCE CHECKER
JSR R1,@JSR4 ;TRY JSR MODE 4

9076										
9077	013406	005737	013646		JSR2:	TST	@JSRSEQ			;CHECK SEQUENCE: JSRSEQ=0?
9078	013412	001011				BNE	JSR2A			;BRANCH IF OUT OF SEQUENCE
9079	013414	020127	013332			CMP	R1,@JSR1A			;PROPER PC SAVED?
9080	013420	001006				BNE	JSR2A			;BRANCH IF PC WRONG
9081	013422	022706	000776			CMP	@STBOT-2,R6			;R6 DECREMENT?
9082	013426	001003				BNE	JSR2A			;BRANCH IF R6 IS INCORRECT
9083	013430	021627	177777			CMP	(R6),# 1			;REGISTER SAVED?
9084	013434	001401				BEQ	JSR2B			
(1)	013436				JSR2A:					
(2)	013436	104000				EMT				;JSR MODE 1 MALFUNCTIONED
9085	013440	012706	001000		JSR2B:	MOV	@STBOT,R6			;INITIALIZE R6
9086	013444	012701	125252			MOV	@125252,R1			;INITIALIZE R1
9087	013450	005237	013646			INC	@JSRSEQ			;UPDATE SEQUENCE CHECKER
9088	013454	012700	013334			MOV	@JSR3,R0			;SET TARGET ADDRESS
9089	013460	004120				JSR	R1,(R0)			;TRY JSR MODE 2
9090										
9091	013462	022737	000002	013646	JSR4:	CMP	@2,@JSRSEQ			;CHECK SEQUENCE: JSRSEQ=2?
9092	013470	001003				BNE	JSR4A			;BRANCH IF OUT OF SEQUENCE
9093	013472	022701	013406			CMP	@JSR2,R1			;PROPER PC SAVED?
9094	013476	001401				BEQ	JSR4B			
(1)	013500				JSR4A:					
(2)	013500	104000				EMT				;JSR MODE 3 MALFUNCTIONED
9095	013502	005237	013646		JSR4B:	INC	@JSRSEQ			;UPDATE SEQUENCE CHECKER
9096	013506	012700	013554			MOV	@JSR5+2,R0			;SET TARGET ADDRESS
9097	013512	004140				JSR	R1, -(R0)			;TRY JSR MODE 4
9098										
9099	013514	022767	000004	000124	JSR6:	CMP	@4,JSRSEQ			;CHECK SEQUENCE: JSRSEQ=4?
9100	013522	001006				BNE	JSR6A			;BRANCH IF OUT OF SEQUENCE
9101	013524	022701	013612			CMP	@JSR7,R1			;PROPER PC SAVED?
9102	013530	001003				BNE	JSR6A			;BRANCH IF PC WRONG
9103	013532	022700	013642			CMP	@JSR6A0,R0			;MODE 5 REGISTER CORRECT?
9104	013536	001401				BEQ	JSR6B			
(1)	013540				JSR6A:					
(2)	013540	104000				EMT				;JSR MODE 5 FAILED
9105	013542	005237	013646		JSR6B:	INC	@JSRSEQ			;UPDATE SEQUENCE CHECKER
9106	013546	004167	000040			JSR	R1,JSR7			;TRY JSR MODE 6
9107	013552	022767	000003	000066	JSR5:	CMP	@3,JSRSEQ			;CHECK SEQUENCE: JSRSEQ=3?
9108	013560	001006				BNE	JSR5A			;BRANCH IF OUT OF SEQUENCE
9109	013562	022701	013514			CMP	@JSR6,R1			;PROPER PC SAVED?
9110	013566	001003				BNE	JSR5A			;BRANCH IF PC WRONG
9111	013570	022700	013552			CMP	@JSR5,R0			;CHECK MODE 4 REGISTER
9112	013574	001401				BEQ	JSR5B			
(1)	013576				JSR5A:					
(2)	013576	104000				EMT				;JSR MODE 4 MALFUNCTIONED
9113	013600	005237	013646		JSR5B:	INC	@JSRSEQ			;UPDATE SEQUENCE CHECKER
9114	013604	012700	013644			MOV	@JSR6A0+2,R0			;POINT R0 TO TARGET ADDRESS
9115	013610	004150				JSR	R1,@-(R0)			;TRY JSR MODE 5
9116										
9117	013612	022737	000005	013646	JSR7:	CMP	@5,@JSRSEQ			;CHECK SEQUENCE: JSRSEQ=5?
9118	013620	001003				BNE	JSR7A			;BRANCH IF OUT OF SEQUENCE
9119	013622	022701	013552			CMP	@JSR5,R1			;PROPER PC SAVED?
9120	013626	001401				BEQ	JSR7B			
(1)	013630				JSR7A:					
(2)	013630	104000				EMT				;JSR MODE 6 FAILED
9121	013632	005237	013646		JSR7B:	INC	@JSRSEQ			;UPDATE SEQUENCE CHECKER

9122 013636 004177 000002
9123
9124 013642 013514
9125 013644 013650
9126 013646 000000
9127
9128 013650 022767 000006 177770
9129 013656 001003
9130 013660 022701 013642
9131 013664 001401
(2) 013666
(3) 013666 104000

JSR R1,BJSRCKAD ;TRY JSR MODE 7
JSR6AD: JSR6 ;MODE 5 TARGET ADDRESS
JSRCKAD: JSRCK ;MODE 7 TARGET ADDRESS
JSRSEQ: 0 ;SEQUENCE CHECKER
JSRCK: CMP #6,JSRSEQ ;CHECK SEQUENCE: JSRSEQ=6?
BNE JSRCK1 ;BRANCH IF OUT OF SEQUENCE
CMP #JSR6AD,R1 ;PROPER PC SAVED?
BEQ TS206
JSRCK1: EMT ;JSR MODE 7 MALFUNCTIONED

9132
9133
9134
9135
9136
9137
9138
9139
9140
9141
(2)
(3)
(2) 013670
9142 013670 012706 001000
9143 013674 012746 052525
9144 013700 012700 013710
9145 013704 000200
9146
9147
9148 013706 104000
9149 013710 022700 052525
9150 013714 001401
(3) 013716 104000

; THIS TEST VERIFIES THE RTS INSTRUCTION. THE STACK POINTER
; IS INITIALIZED AND A TEST PATTERN STORED ON STACK. R0 IS LOADED
; WITH RETURN ADDRESS. AN RTS IS EXECUTED, AND, AT THE TARGET
; ADDRESS, A CHECK IS MADE THAT R0 WAS PROPERLY RESTORED FROM THE
; STACK.

; TEST 206 TEST RTS INSTRUCTION

TS206:
MOV #STBOT,R6 ;INITIALIZE STACK POINTER
MOV #52525,-(R6) ;INITIALIZE TOP OF STACK
MOV #RTS1,R0 ;INITIALIZE RETURN REGISTER
RTS R0 ;TRY RTS THROUGH R0
; TO SCOPE: REPLACE THE MOVE INSTRUCTION <****
; FOLLOWING W/ 770 <****
EMT ;RTS FAILED
RTS1: CMP #52525,R0 ;CHECK THAT R0 RESTORED FROM STACK
BEQ TS207
EMT ;RTS MALFUNCTIONED

9151
9152
9153
9154
9155
9156
9157
9158
9159
9160
9161
9162
9163
9164
9165
(2)
(3)
(2) 013720
9166 013720 000277
9167 013722 000251
9168 013724 012700 100000

; THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF A GROUP
; OF FOUR INSTRUCTIONS. THE GROUP CONSISTS OF THE INSTRUCTIONS:
; MOV, BIC, BIT, AND BIS. THESE INSTRUCTIONS ARE SIMILAR IN THE
; WAY THEY EFFECT THE C AND V BITS. THEY ALL LEAVE THE V-BIT
; CLEAR AND THE C-BIT UNAFFECTED.
; THE TEST PROCEDURE IS AS FOLLOWS: THE N, Z, AND V BITS
; ARE LOADED WITH THE COMPLEMENT OF THE EXPECTED RESULTS, THE C-BIT
; IS LOADED WITH THE DESIRED RESULT. THE INSTRUCTION IS EXECUTED
; WITH DIFFERENT DATA PATTERNS AND THE RESULTS ARE VERIFIED WITH
; A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS. THE DATA IS CHOSEN
; TO PRODUCT ALL POSSIBLE COMBINATIONS OF THE C AND V BITS.

; TEST 207 TEST MOV INSTRUCTION

TS207:
SCC ;CC=0110
+CLN!CLC
MOV #100000,R0 ;CC=1000

9169 013730 101402
9170 013732 102401
9171 013734 100401
(1) 013736
(2) 013736 104000
9172
9173 013740 000277
9174 013742 000244
9175 013744 012700 000000
9176 013750 101002
9177 013752 102401
9178 013754 100001
(2) 013756
(3) 013756 104000
9179
(2)
(3)
(2) 013760
9180 013760 012700 100001
9181 013764 000277
9182 013766 000251
9183 013770 032700 100000
9184 013774 101402
9185 013776 102401
9186 014000 100401
(1) 014002
(2) 014002 104000
9187
9188 014004 000277
9189 014006 000244
9190 014010 032700 077776
9191 014014 101002
9192 014016 102401
9193 014020 100001
(2) 014022
(3) 014022 104000
9194
(2)
(3)
(2) 014024
9195 014024 012700 177777
9196 014030 000277
9197 014032 000251
9198 014034 042700 077777
9199 014040 101402
9200 014042 102401
9201 014044 100401
(1) 014046
(2) 014046 104000
9202 014050 000277
9203 014052 000244
9204 014054 042700 100000
9205 014060 101002
9206 014062 102401
9207 014064 100001
(2) 014066

BLOS MOV1
BVS MOV1
BMI MOV2
MOV1: EMT ;MOV DID NOT SET CC'S CORRECTLY
MOV2: SCC ;CC=1011
CLZ
MOV #0,R0 ;CC=0101
BHI MOV3 ;C OR Z = 0?
BVS MOV3 ;V=1?
BPL TS210
MOV3: EMT ;MOV DID NOT SET CC'S CORRECTLY
;*****
;TEST 210 TEST BIT INSTRUCTION
;*****
TS210: MOV #100001,R0
SCC ;CC=0110
+CLN!CLC
BIT #100000,R0 ;CC=1000
BLOS BITST1
BVS BITST1
BMI BITST2
BITST1: EMT ;BIT DID NOT SET CC'S CORRECTLY
BITST2: SCC ;CC=1011
CLZ
BIT #77776,R0 ;CC=0101
BHI BITST3
BVS BITST3
BPL TS211
BITST3: EMT ;BIT DID NOT SET CC'S CORRECTLY
;*****
;TEST 211 TEST BIC INSTRUCTION
;*****
TS211: MOV #177777,R0
SCC ;CC=0110
+CLN!CLC
BIC #77777,R0 ;CC=1000
BLOS BIC1
BVS BIC1
BMI BIC2
BIC1: EMT ;BIC DID NOT SET CC'S CORRECTLY
BIC2: SCC ;CC=1011
CLZ
BIC #100000,R0 ;CC=0101
BHI BIC3
BVS BIC3
BPL TS212
BIC3:

(3) 014066 104000
9208
(2)
(3)
(2) 014070
9209 014070 005000
9210 014072 000277
9211 014074 000251
9212 014076 002700 000000
9213 014102 103403
9214 014104 102402
9215 014106 100401
9216 014110 001401
(1) 014112
(2) 014112 104000
9217 014114 000277
9218 014116 000250
9219 014120 052700 177777
9220 014124 103003
9221 014126 102402
9222 014130 001401
9223 014132 100401
(2) 014134
(3) 014134 104000
9224
9225
9226
9227
9228
9229
9230
9231
9232
9233
9234
9235
9236
(2)
(3)
(2) 014136
9237 014136 012700 077777
9238 014142 000257
9239 014144 000264
9240 014146 005200
9241 014150 101402
9242 014152 100001
9243 014154 102401
(1) 014156
(2) 014156 104000
9244 014160 052700 077777
9245 014164 000261
9246 014166 000244
9247 014170 005200
9248 014172 100403
9249 014174 102402
9250 014176 103001

```
EMT ;BIC DID NOT SET CC S CORRECTLY
;*****
;TEST 212 TEST BIC INSTRUCTION
;*****
TS212:
CLR RO ;RO=0
SCC ;CC=1010
*CLN!CLC
BIS #0,RO ;CC=0100 RO=0
BCS BIS1
BVS BIS1
BMI BIS1
BEQ BIS2

BIS1: EMT ;BIS DID NOT SET CC S CORRECTLY
BIS2: SCC ;CC=0111
CLN
BIS #177777,RO ;CC=1001
BCC BIS3
BVS BIS3
BEQ BIS3
BMI TS213

BIS3: EMT ;BIS DID NOT SET CC'S CORRECTLY
;*****
;
; THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE INC AND
; DEC INSTRUCTIONS. THESE INSTRUCTIONS BOTH EFFECT THE C AND V
; BITS THE SAME; THE C-BIT IS LEFT UNCHANGED AND THE V-BIT IS DEPENDENT
; UPON THE DATA RESULTS. THE SAME PROCEDURE IS USED. THE CONDITION
; CODE BITS ARE INITIALIZED, THE INSTRUCTION IS EXECUTED AND THE
; RESULTS ARE VERIFIED WITH A SERIES OF CONDITIONAL BRANCH INSTRUCTIONS.
; THIS PROCEDURE IS REPEATED WITH SEVERAL DATA PATTERNS TO PRODUCE
; DIFFERENT COMBINATIONS OF THE C AND V BITS.
;*****
;TEST 213 TEST INC INSTRUCTION
;*****
TS213:
MOV #077777,RO ;RO=077777
CCC ;CC=0100
SEZ
INC RO ;CC=1010 RO=10000
BLOS INC1
BPL INC1
BVS INC2

INC1: EMT ;INC DID NOT SET CC'S CORRECTLY
INC2: BIS #77777,RO ;RO=177777
SEC ;CC=1011
CLZ
INC RO ;CC=0101 RO=0
BMI INC3
BVS INC3
BCC INC3
```

9251 014200 001401
(1) 014202
(2) 014202 104000
9252
9253 014204 000277
9254 014206 000241
9255 014210 005200
9256 014212 101402
9257 014214 100401
9258 014216 100001
(2) 014220
(3) 014220 104000
9259
9260
(2)
(3)
(2) 014222
9261 014222 012700 000002
9262 014226 000277
9263 014230 005300
9264 014232 100403
9265 014234 001402
9266 014236 102401
9267 014240 103401
(1) 014242
(2) 014242 104000
9268 014244 000261
9269 014246 000244
9270 014250 005300
9271 014252 101002
9272 014254 100401
9273 014256 102001
(1) 014260
(2) 014260 104000
9274 014262 000277
9275 014264 000251
9276 014266 005300
9277 014270 101402
9278 014272 102401
9279 014274 100401
(1) 014276
(2) 014276 104000
9280 014300 042700 077777
9281 014304 000277
9282 014306 000252
9283 014310 005300
9284 014312 100403
9285 014314 001402
9286 014316 102001
9287 014320 103401
(2) 014322
(3) 014322 104000
9288
9289
9290
9291

INC3: BEQ INC4
EMT ;INC DID NOT SET CC'S CORRECTLY
INC4: SCC ;CC=1110
CLC
INC RO ;CC=0000 RO=1
BLOS INC5
BMI INC5
BPL TS214
INC5: EMT ;INC DID NOT SET CC'S CORRECTLY

;TEST 214 TEST DEC INSTRUCTION

TS214:

MOV #2,RO ;RO=2
SCC ;CC=1111
DEC RO ;CC=0001 RO=1
BMI DEC1
BEQ DEC1
BVS DEC1
BCS DEC2
DEC1: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC2: SEC ;CC=1011
CLZ
DEC RO ;CC=0101 RO=0
BHI DEC3
BMI DEC3
BVC DEC4
DEC3: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC4: SCC ;CC=0110
+CLN!CLC
DEC RO ;CC=1000 RO=17777
BLOS DEC5
BVS DEC5
BMI DEC6
DEC5: EMT ;DEC DID NOT SET CC'S CORRECTLY
DEC6: BIC #77777,RO ;RO=100000
SCC ;CC=0101
+CLN!CLV
DEC RO ;CC=1011 RO=77777
BMI DEC7
BEQ DEC7
BVC DEC7
BCS TS215
DEC7: EMT ;DEC DID NOT SET CC'S CORRECTLY

;

9292
9293
9294
9295
9296
9297
9298
9299
(2)
(3)
(2) 014324
9300 014324 000277
9301 014326 000244
9302 014330 005000
9303 014332 100403
9304 014334 102402
9305 014336 103401
9306 014340 001401
(2) 014342
(3) 014342 104000
9307
9308
(2)
(3)
(2) 014344
9309 014344 000277
9310 014346 000244
9311 014350 005700
9312 014352 100403
9313 014354 102402
9314 014356 103401
9315 014360 001401
(1) 014362
(2) 014362 104000
9316 014364 005300
9317 014366 000277
9318 014370 000250
9319 014372 005700
9320 014374 101402
9321 014376 102401
9322 014400 100401
(2) 014402
(3) 014402 104000
9323
(2)
(3)
(2) 014404
9324 014404 012700 170000
9325 014410 000277
9326 014412 000250
9327 014414 000300
9328 014416 101402
9329 014420 102401
9330 014422 100401
(1) 014424
(2) 014424 104000

THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE CLR,
TST, AND SWAB INSTRUCTIONS. THESE THREE INSTRUCTIONS ALL LEAVE
THE C AND V BITS CLEARED. AGAIN, THE CONDITION CODES ARE PRESET,
THE INSTRUCTION EXECUTED AND THE RESULTS CHECKED WITH CONDITIONAL
BRANCH INSTRUCTIONS. THE PROCEDURE IS REPEATED TO PRODUCE OTHER
COMBINATIONS OF CONDITION CODES.

;TEST 215 TEST CLR INSTRUCTION

TS215:
SCC ;CC=1011
CLZ
CLR RO ;CC=0100 RO=0
BMI CLR1
BVS CLR1
BCS CLR1
BEQ TS216

CLR1: EMT ;CLR DID NOT SET CC'S CORRECTLY

;TEST 216 TEST TST INSTRUCTION

TS216:
SCC ;CC=1011
CLZ
TST RO ;CC=0100
BMI TEST1
BVS TEST1
BCS TEST1
BEQ TEST2

TEST1: EMT ;TEST DID NOT SET CC'S CORRECTLY
TEST2: DEC RO ;MAKE RO NEGATIVE
SCC ;CC=0111
CLN
TST RO ;CC=1000
BLOS TEST3
BVS TEST3
BMI TS217

TEST3: EMT ;TEST DID NOT SET CC'S CORRECTLY

;TEST 217 TEST SWAB INSTRUCTION

TS217:
MOV #170000,RO ;RO=170000
SCC ;CC=0111
CLN
SWAB RO ;CC=1000 RO=360
BLOS SWB1
BVS SWB1
BMI SWB2

SWB1: EMT ;SWAB DID NOT SET CC'S CORRECTLY

9331 014426 000277
 9332 014430 000244
 9333 014432 000300
 9334 014434 102403
 9335 014436 103402
 9336 014440 100401
 9337 014442 001401
 (2) 014444
 (3) 014444 104000
 9338
 9339
 9340
 9341
 9342
 9343
 9344
 9345
 9346
 9347
 9348
 9349
 (2)
 (3)
 (2) 014446
 9350 014446 012700 040000
 9351 014452 000277
 9352 014454 062700 030000
 9353 014460 101402
 9354 014462 102401
 9355 014464 100001
 (1) 014466
 (2) 014466 104000
 9356 014470 000264
 9357
 9358 014472 062700 010000
 9359 014476 101402
 9360 014500 102001
 9361 014502 100401
 (1) 014504
 (2) 014504 104000
 9362 014506 000257
 9363 014510 000270
 9364 014512 062700 100000
 9365 014516 101002
 9366 014520 102001
 9367 014522 100001
 (1) 014524
 (2) 014524 104000
 9368 014526 062700 177777
 9369 014532 101402
 9370 014534 102401
 9371 014536 100401
 (1) 014540
 (2) 014540 104000
 9372 014542 000277
 9373 014544 000245

SWB2: SCC ;CC=1011
 CLZ
 SWAB RO ;CC=0100 RO=170000
 BVS SWB3
 BCS SWB3
 BMI SWB3
 BEQ TS220
 SWB3: EMT ;

 :
 : THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE ADD AND
 : ADC INSTRUCTIONS. BOTH OF THESE INSTRUCTIONS HANDLE THE C AND
 : V BITS IDENTICALLY. THE PROCEDURE IS TO PRESET THE CONDITION
 : CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
 : THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
 : BRANCHES. THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
 : DATA TO PRODUCE EVERY COMBINATION OF C AND V BITS.
 :
 : *****

 : TEST 220 TEST ADD INSTRUCTION
 : *****

TS220:
 MOV #40000,RO ;RO=40000
 SCC ;CC=1111
 ADD #30000,RO ;CC=0000 RO=70000
 BLOS ADD1
 BVS ADD1
 BPL ADD2
 ADD1: EMT ;ADD DID NOT SET CC'S CORRECTLY
 ADD2: SEZ ;CC=0100
 ADD #10000,RO ;CC=1010 RO=100000
 BLOS ADD3
 BVC ADD3
 BMI ADD4
 ADD3: EMT ;ADD DID NOT SET CC'S CORRECTLY
 ADD4: CCC ;CC=1000
 SEN
 ADD #100000,RO ;CC=0111 RO=0
 BMI ADD5
 BVC ADD5
 BPL ADD6
 ADD5: EMT ;ADD DID NOT SET CC'S CORRECTLY
 ADD6: ADD #177777,RO ;CC=1000 RO=177777
 BLOS ADD7
 BVS ADD7
 BMI ADD8
 ADD7: EMT ;ADD DID NOT SET CC'S CORRECTLY
 ADD8: SCC ;CC=1010
 +CLC!CLZ

9374 014546 062700 000001
9375 014552 102403
9376 014554 103002
9377 014556 100401
9378 014560 001401
(2) 014562
(3) 014562 104000
9379
9380
(2)
(3)
(2) 014564
9381 014564 012700 077777
9382 014570 000277
9383 014572 000252
9384 014574 005500
9385 014576 101402
9386 014600 102001
9387 014602 100401
(1) 014604
(2) 014604 104000
9388 014606 052700 077777
9389 014612 000277
9390 014614 000244
9391 014616 005500
9392 014620 101002
9393 014622 102401
9394 014624 100001
(1) 014626
(2) 014626 104000
9395 014630 000277
9396 014632 000245
9397 014634 005500
9398 014636 102403
9399 014640 103402
9400 014642 100401
9401 014644 001401
(2) 014646
(3) 014646 104000
9402
9403
9404
9405
9406
9407
9408
9409
9410
9411
9412
9413
(2)
(3)
(2) 014650
9414 014650 012700 000001
9415 014654 000277

```
ADD #1,R0 ;CC=0101 R=0
BVS ADD9
BCC ADD9
BMI ADD9
BEQ TS221
ADD9: EMT ;ADD DID NOT SET CC'S CORRECTLY
;*****
;TEST 221 TEST ADC INSTRUCTION
;*****
TS221: MOV #077777,R0
      SCC ;CC=0101
      +CLN!CLV
      ADC R0 ;CC=1010
      BLOS ADC1
      BVC ADC1
      BMI ADC2
ADC1: EMT ;ADC DID NOT SET CC'S CORRECTLY
ADC2: BIS #77777,R0
      SCC ;CC=1011
      CLZ
      ADC R0 ;CC=0101 R0=0
      BHI ADC3
      BVS ADC3
      BPL ADC4
ADC3: EMT ;ADC DID NOT SET CC'S CORRECTLY
ADC4: SCC
      +CLZ!CLC ;CC=1010
      ADC R0 ;CC=0100
      BVS ADC5
      BCS ADC5
      BMI ADC5
      BEQ TS222
ADC5: EMT ;ADC DID NOT SET CC'S CORRECTLY
;*****
;
; THESE NEXT THREE TESTS VERIFY THE FUNCTIONING OF THE NEG,
; CMP, AND COM INSTRUCTIONS. EACH OF THESE INSTRUCTIONS GENERATE
; THE C AND V BITS IDENTICALLY. THE CONDITION CODES ARE PRESET,
; THE INSTRUCTIONS EXECUTED, AND THE RESULTS CHECKED WITH A SERIES
; OF CONDITIONAL BRANCH INSTRUCTIONS. THIS PROCEDURE IS REPEATED
; SEVERAL TIMES WITH DIFFERENT DATA IN ORDER TO GENERATE DIFFERENT
; COMBINATIONS OF THE C AND V BITS.
;*****
;TEST 222 TEST NEG INSTRUCTION
;*****
TS222: MOV #1,R0
      SCC ;CC=0110
```

9416	014656	000251				.CLN!CLC		
9417	014660	005400				NEG	RO	;CC=1001 RO=177777
9418	014662	103003				BCC	NEG1	
9419	014664	102402				BVS	NEG1	
9420	014666	001401				BEQ	NEG1	
9421	014670	100401				BMI	NEG2	
(1)	014672			NEG1:				
(2)	014672	104000				EMT		;NEG DID NOT SET CC'S CORRECTLY
9422	014674	042700	077777	NEG2:		BIC	#77777,RO	
9423	014700	000257				CCC		;CC=0100
9424	014702	000264				SEZ		
9425	014704	005400				NEG	RO	;CC=1011 RO=100000
9426	014706	102003				BVC	NEG3	
9427	014710	103002				BCC	NEG3	
9428	014712	001401				BEQ	NEG3	
9429	014714	100401				BMI	NEG4	
(1)	014716			NEG3:				
(2)	014716	104000				EMT		;NEG DID NOT SET CC'S CORRECTLY
9430	014720	005000		NEG4:		CLR	RO	
9431	014722	000277				SCC		;CC=1011
9432	014724	000244				CLZ		
9433	014726	005400				NEG	RO	;CC=0100 RO=0
9434	014730	102403				BVS	NEG5	
9435	014732	103402				BCS	NEG5	
9436	014734	001001				BNE	NEG5	
9437	014736	100001				BPL	TS223	
(2)	014740			NEG5:				
(3)	014740	104000				EMT		;NEG DID NOT SET CC'S CORRECTLY

9438								
9439								
(2)								
(3)								
(2)	014742							
9440	014742	012700	000005			MOV	#5,RO	
9441	014746	000257				CCC		;CC=1010
9442	014750	000271				.SEN!SEC		
9443	014752	022700	000005			CHP	#5,RO	;CC=0101
9444	014756	101002				BHI	CHP1	
9445	014760	102401				BVS	CHP1	
9446	014762	100001				BPL	CHP2	
(1)	014764			CMP1:				
(2)	014764	104000				EMT		;CMP DID NOT SET CC'S CORRECTLY
9447	014766	012700	100000	CMP2:		MOV	#100000,RO	
9448	014772	000277				SCC		;CC=1101
9449	014774	000242				CLV		
9450	014776	020027	077777			CHP	RO,#77777	;CC=0010
9451	015002	101402				BLOS	CMP3	
9452	015004	102001				BVC	CMP3	
9453	015006	100001				BPL	CMP4	
(1)	015010			CMP3:				
(2)	015010	104000				EMT		;CMP DID NOT SET CC'S CORRECTLY
9454	015012	052700	040000	CMP4:		BIS	#40000,RO	;RO=140000
9455	015016	000257				CCC		;CC=0100
9456	015020	000264				SEZ		
9457	015022	022700	040000			CHP	#40000,RO	;CC=1011
9458	015026	102003				BVC	CMP5	

9459 015030 103002
 9460 015032 001401
 9461 015034 100401
 (1) 015036
 (2) 015036 104000
 9462 015040 042700 040000
 9463 015044 000277
 9464 015046 022700 177777
 9465 015052 101402
 9466 015054 102401
 9467 015056 100001
 (2) 015060
 (3) 015060 104000
 9468
 9469
 (2)
 (3)
 (2) 015062
 9470 015062 012700 177777
 9471 015066 000257
 9472 015070 000265
 9473 015072 005100
 9474 015074 101002
 9475 015076 102401
 9476 015100 100001
 (2) 015102
 (3) 015102 104000
 9477
 9478
 9479
 9480
 9481
 9482
 9483
 9484
 9485
 9486
 9487
 9488
 9489
 (2)
 (3)
 (2) 015104
 9490 015104 012700 125252
 9491 015110 000257
 9492 015112 000271
 9493 015114 162700 125252
 9494 015120 101002
 9495 015122 102401
 9496 015124 100001
 (1) 015126
 (2) 015126 104000
 9497 015130 052700 100000
 9498 015134 000277
 9499 015136 000242
 9500 015140 162700 077777

BCC CMP5
 BEQ CMP5
 BMI CMP6
 CMP5: EMT ;CMP DID NOT SET CC S CORRECTLY
 CMP6: BIC #40000,R0
 SCC ;CC=1111
 CMP #-1,R0 ;CC=0000
 BLOS CMP7
 BVS CMP7
 BPL TS224
 CMP7: EMT ;CMP DID NOT SET CC'S CORRECTLY

 ;TEST 224 TEST COM INSTRUCTION
 ;*****

TS224: MOV #-1,R0
 CCC ;CC=1010
 +SEC!SEZ
 COM R0 ;CC=0101
 BHI COM1
 BVS COM1
 BPL TS225

COM1: EMT ;COM DID NOT SET CC'S CORRECTLY

 ;
 ; THESE NEXT TWO TESTS VERIFY THE FUNCTIONING OF THE SUB
 ; AND SBC INSTRUCTIONS. BOTH OF THESE INSTRUCTIONS HANDLE THE
 ; C AND V BITS IDENTICALLY. THE PROCEDURE IS TO PRESET THE CONDITION
 ; CODES, EXECUTE THE INSTRUCTION WITH A PARTICULAR SET OF DATA, AND
 ; THEN CHECK THE RESULTS BY EXECUTING A SERIES OF CONDITIONAL
 ; BRANCHES. THIS PROCEDURE IS REPEATED SEVERAL TIMES WITH DIFFERENT
 ; DATA PATTERNS TO PROVIDE EVERY COMBINATION OF THE C AND V BITS.
 ;

 ;TEST 225 TEST SUB INSTRUCTION
 ;*****

TS225: MOV #125252,R0
 CCC ;CC=1010
 +SEN!SEC
 SUB #125252,R0 ;CC=0101 R0=0
 BHI SUB1
 BVS SUB1
 BPL SUB2

SUB1: EMT ;SUB DID NOT SET CC'S CORRECTLY
 SUB2: BIS #100000,R0
 SCC ;CC=1101
 CLV
 SUB #77777,R0 ;CC=0010 R0=1

N

9501 015144 101402
 9502 015146 102001
 9503 015150 100001
 (1) 015152
 (2) 015152 104000
 9504 015154 005100
 9505 015156 000277
 9506
 9507 015160 162700 100000
 9508 015164 101402
 9509 015166 102401
 9510 015170 100001
 (1) 015172
 (2) 015172 104000
 9511 015174 000257
 9512 015176 000264
 9513 015200 162700 140000
 9514 015204 102003
 9515 015206 103002
 9516 015210 001401
 9517 015212 100401
 (2) 015214
 (3) 015214 104000
 9518
 9519
 (2)
 (3)
 (2) 015216
 9520 015216 012700 000001
 9521 015222 000277
 9522 015224 000244
 9523 015226 005600
 9524 015230 103403
 9525 015232 102402
 9526 015234 100401
 9527 015236 001401
 (1) 015240
 (2) 015240 104000
 9528 015242 000277
 9529 015244 000245
 9530 015246 005600
 9531 015250 103403
 9532 015252 102402
 9533 015254 100401
 9534 015256 001401
 (1) 015260
 (2) 015260 104000
 9535 015262 000277
 9536 015264 000250
 9537 015266 005600
 9538 015270 103003
 9539 015272 102402
 9540 015274 001401
 9541 015276 100401
 (1) 015300
 (2) 015300 104000

BLOS SUB3
 BVC SUB3
 BPL SUB4
 SUB3: EMT
 SUB4: COM RO ;RO=177777
 SCC ;CC=11111
 SUB #100000,RO ;CC=0000 RO=77777
 BLOS SUB5
 BVS SUB5
 BPL SUB6
 SUB5: EMT ;SUB DID NOT SET CC'S CORRECTLY
 SUB6: CCC ;CC=0100
 SEZ
 SUB #140000,RO ;CC=1011
 BVC SUB7
 BCC SUB7
 BEQ SUB7
 BMI TS226
 SUB7: EMT ;
 ;*****
 ;TEST 226 TEST SBC INSTRUCTION
 ;*****
 TS226:
 MOV #1,RO
 SCC ;CC=1011
 CLZ
 SBC RO ;CC=0100 R=0
 BCS SBC1
 BVS SBC1
 BMI SBC1
 BEQ SBC2
 SBC1: EMT ;SBC DID NOT SET CC'S CORRECTLY
 SBC2: SCC ;CC=1010
 +CLZ!CLC
 SBC RO ;CC=0100 R=0
 BCS SBC3
 BVS SBC3
 BMI SBC3
 BEQ SBC4
 SBC3: EMT ;SBC DID NOT SET CC'S CORRECTLY
 SBC4: SCC ;CC=0111
 CLN
 SBC RO ;CC=1001 RO=177777
 BCC SBC5
 BVS SBC5
 BEQ SBC5
 BMI SBC6
 SBC5: EMT ;SBC DID NOT SET CC'S CORRECTLY

68

9542 015302 042700 077777
9543 015306 000277
9544 015310 000242
9545 015312 005600
9546 015314 101402
9547 015316 102001
9548 015320 100001
(2) 015322
(3) 015322 104000
9549
9550
9551
9552
9553
9554
9555
9556
9557
9558
9559
9560
(2)
(3)
(2) 015324
9561 015324 012700 144000
9562 015330 000257
9563 015332 000266
9564 015334 006100
9565 015336 103003
9566 015340 102402
9567 015342 001401
9568 015344 100401
(1) 015346
(2) 015346 104000
9569 015350 000277
9570 015352 000243
9571 015354 006100
9572 015356 103003
9573 015360 102002
9574 015362 001401
9575 015364 100001
(1) 015366
(2) 015366 104000
9576 015370 000277
9577 015372 000250
9578 015374 006100
9579 015376 101402
9580 015400 102401
9581 015402 100001
(1) 015404
(2) 015404 104000
9582 015406 000257
9583 015410 000265
9584 015412 006100
9585 015414 101405
9586 015416 102004

SBC6: BIC #77777,RO ;RO=100000
SCC ;CC=1101
CLV
SBC RO ;CC=0010
BLOS SBC7
BVC SBC7
BPL TS227
SBC7: EMT ;SBC DID NOT SET CC'S CORRECTLY

.....
; THESE NEXT FOUR TESTS VERIFY THE FUNCTIONING OF THE ROL,
; ROR, ASL AND ASR INSTRUCTIONS. SPECIAL DATA PATTERNS ARE LOADED
; AND ROTATED SEVERAL TIMES FOR EACH TEST. THE CONDITION CODES
; ARE PRESET BEFORE EACH ROTATION AND THE CONDITION CODES ARE
; CHECKED AFTER EACH ROTATION. THE FINAL CHECK IN EACH TEST IS
; TO VERIFY THE CUMULATIVE DATA RESULT. THE DATA PATTERNS HAVE
; BEEN SELECTED TO PRODUCE ALL COMBINATIONS OF THE C AND V BITS.
;

TEST 227 TEST ROL INSTRUCTION
.....
TS227:
MOV #144000,RO ;RO=144000
CCC ;CC=0110
+SEZ!SEV
ROL RO ;CC=1001 RO=110000
BCC ROL1
BVS ROL1
BEQ ROL1
BMI ROL2
ROL1: EMT ;
ROL2: SCC ;CC=1100
+CLV!CLC
ROL RO ;CC=0011 RO=020000
BCC ROL3
BVC ROL3
BEQ ROL3
BPL ROL4
ROL3: EMT ;ROL DID NOT SET CC'S CORRECTLY
ROL4: SCC ;CC=0111
CLN
ROL RO ;CC=0000 RO=040001
BLOS ROL5
BVS ROL5
BPL ROL6
ROL5: EMT ;ROL DID NOT SET CC'S CORRECTLY
ROL6: CCC ;CC=0101
+SEZ!SEC
ROL RO ;CC=1010 RO=100003
BLOS ROL7
BVC ROL7

08

9587 015420 100003
9588 015422 022700 100003
9589 015426 001401
(2) 015430
(3) 015430 104000
9590
(2)
(3)
(2) 015432
9591 015432 012700 000023
9592 015436 000277
9593 015440 000250
9594 015442 006000
9595 015444 102403
9596 015446 103002
9597 015450 001401
9598 015452 100401
(1) 015454
(2) 015454 104000
9599 015456 000257
9600 015460 000274
9601 015462 006000
9602 015464 102003
9603 015466 103002
9604 015470 001401
9605 015472 100001
(1) 015474
(2) 015474 104000
9606 015476 000277
9607 015500 000241
9608 015502 006000
9609 015504 101403
9610 015506 102402
9611 015510 001401
9612 015512 100001
(1) 015514
(2) 015514 104000
9613 015516 000257
9614 015520 000265
9615 015522 006000
9616 015524 101402
9617 015526 102001
9618 015530 100401
(2) 015532
(3) 015532 104000
9619
(2)
(3)
(2) 015534
9620 015534 012700 144000
9621 015540 000257
9622 015542 000271
9623 015544 006300
9624 015546 103003
9625 015550 102402
9626 015552 001401

BPL ROL 7
CMP #100003,RO
BEQ TS230
ROL 7:
EMT ;ROL MALFUNCTIONED
;.....
;TEST 230 TEST ROR INSTRUCTION
;.....
TS230:
MOV #23,RO ;RO=23
SCC ;CC=0111
CLN
ROR RO ;CC=1001 RO=100011
BVS ROR1
BCC ROR1
BEQ ROR1
BMI ROR2
ROR1:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR2:
CCC ;CC=1100
+SEN!SEZ
ROR RO ;CC=0011 RO=040004
BVC ROR3
BCC ROR3
BEQ ROR3
BPL ROR4
ROR3:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR4:
SCC ;CC=1110
CLC
ROR RO ;CC=0000 RO=020002
BLOS ROR5
BVS ROR5
BEQ ROR5
BPL ROR6
ROR5:
EMT ;ROR DID NOT SET CC'S CORRECTLY
ROR6:
CCC ;CC=0101
+SEC!SEZ
ROR RO ;CC=1010 RO=110001
BLOS ROR7
BVC ROR7
BMI TS231
ROR7:
EMT ;ROR DID NOT PRODUCE CORRECT RESULTS
;.....
;TEST 231 TEST ASL INSTRUCTION
;.....
TS231:
MOV #144000,RO ;RO=14000
CCC ;CC=0110
+SEN!SEC
ASL RO ;CC=1001 RO=110000
BCC ASL1
BVS ASL1
BEQ ASL1

9627 015554 100401
(1) 015556
(2) 015556 104000
9628 015560 000277
9629 015562 000243
9630 015564 006300
9631 015566 103003
9632 015570 102002
9633 015572 001401
9634 015574 100001
(1) 015576
(2) 015576 104000
9635 015600 000277
9636 015602 000250
9637 015604 006300
9638 015606 101402
9639 015610 102401
9640 015612 100001
(1) 015614
(2) 015614 104000
9641 015616 000257
9642 015620 000265
9643 015622 006300
9644 015624 103406
9645 015626 001405
9646 015630 102004
9647 015632 100003
9648 015634 022700 100000
9649 015640 001401
(2) 015642
(3) 015642 104000
9650
(2)
(3)
(2) 015644
9651 015644 012700 100023
9652 015650 000277
9653 015652 000250
9654 015654 006200
9655 015656 102403
9656 015660 103002
9657 015662 001401
9658 015664 100401
(1) 015666
(2) 015666 104000
9659 015670 042700 100000
9660 015674 000277
9661 015676 000243
9662 015700 006200
9663 015702 102003
9664 015704 103002
9665 015706 001401
9666 015710 100001
(1) 015712
(2) 015712 104000
9667 015714 000277

BMI ASL 2
ASL 1: EMT
ASL 2: SCC ;CC=1100
-CLV!CLC
ASL RO ;CC=0011 RO=020000
BCC ASL 3
BVC ASL 3
BEQ ASL 3
BPL ASL 4
ASL 3: EMT
ASL 4: SCC ;ASL DID NOT SET CC'S CORRECTLY
CLN ;CC=0111
ASL RO ;CC=0000 RO=040000
BLOS ASL 5
BVS ASL 5
BPL ASL 6
ASL 5: EMT
ASL 6: CCC ;ASL DID NOT SET CC'S CORRECTLY
-SEZ!SEC ;CC=0101
ASL RO ;CC=1010 RO=100000
BCS ASL 7
BEQ ASL 7
BVC ASL 7
BPL ASL 7
CMP #100000,RO
BEQ TS232
ASL 7: EMT ;ASL MALFUNCTIONED
;*****
;TEST 232 TEST ASR INSTRUCTION
;*****
TS232:
MOV #100023,RO ;RO=100023
SCC ;CC=0110
CLN
ASR RO ;CC=1001 RP=140011
BVS ASR 1
BCC ASR 1
BEQ ASR 1
BMI ASR 2
ASR 1: EMT
ASR 2: BIC #100000,RO ;ASR DID NOT SET CC'S CORRECTLY
SCC ;RO=40011
-CLV!CLC ;CC=1100
ASR RO ;CC=0011 RO=020004
BVC ASR 3
BCC ASR 3
BEQ ASR 3
BPL ASR 4
ASR 3: EMT
ASR 4: SCC ;ASR DID NOT SET CC'S CORRECTLY
;CC=1111

```

9668
9669 015716 006200 ASR RO ;CC=0000 RO=010002
9670 015720 101403 BLOS ASR5
9671 015722 102402 BVS ASR5
9672 015724 001401 BEQ ASR5
9673 015726 100001 BPL ASR6
(1) 015730 ASR5:
(2) 015730 104000 EMT ;ASR DID NOT SET CC'S CORRECTLY
9674 015732 052700 100000 ASR6: BIS #100000,RO ;RO=110002
9675 015736 000257 CCC ;CC=0101
9676 015740 000265 .SEZ!SEC
9677 015742 006200 ASR RO ;C=1010 RO=144001
9678 015744 101406 BLOS ASR7
9679 015746 102005 BVC ASR7
9680 015750 100004 BPL ASR7
9681 015752 001403 BEQ ASR7
9682 015754 022700 144001 CMP #144001,RO ;CHECK RESULT OF ASR'S
9683 015760 001401 BEQ TS233
(2) 015762 ASR7:
(3) 015762 104000 EMT ;ASR DID NOT FUNCTION CORRECTLY

```

```

9684
9685
9686
(2)
(3)

```

```

(2) 015764
9687 015764 112701 000004 TS233: MOVB #4,R1 ;LOAD REGISTER
9688 015770 000257 CCC ;CLEAR ALL FLAGS
9689 015772 106001 RORB R1 ;SHIFT BYTE RIGHT
9690 015774 106001 RORB R1 ;SHIFT BYTE RIGHT
9691 015776 122701 000001 CMPB #1,R1 ;CHECK RESULT
9692 016002 001401 BEQ RORB1
(2) 016004 104000 EMT ;RORB DID NOT FUNCTION CORRECTLY
9693 016006 106001 RORB1: RORB R1 ;SHIFT BYTE RIGHT
9694 016010 100403 BMI RORB2 ;CC=7?
9695 016012 001002 BNE RORB2
9696 016014 102001 BVC RORB2
9697 016016 103401 BCS RORB3
(1) 016020 RORB2:
(2) 016020 104000 EMT ;RORB DID NOT SET CC'S CORRECTLY
9698 016022 106001 RORB3: RORB R1 ;SHIFT BYTE RIGHT
9699 016024 100002 BPL RORB4 ;CC=12
9700 016026 101401 BLOS RORB4
9701 016030 102401 BVS RORB5
(1) 016032 RORB4:
(2) 016032 104000 EMT ;RORB DID NOT SET CC CORRECTLY
9702 016034 122701 000200 RORB5: CMPB #200,R1 ;CHECK RESULT
9703 016040 001401 BEQ RORB7
(2) 016042 104000 EMT ;RORB DID NOT FUNCTION CORRECTLY
9704 016044 RORB7:
9705 ;ROTATE ODD BYTE
9706 016044 005000 CLR RO ;MAKE RO ZERO
9707 016046 012710 025125 MOV #025125,(RO) ;PUT STARTING VALUE IN LOC. 0
9708 016052 005200 INC RO ;MAKE RO POINT TO ODD BYTE
9709 016054 000257 CCC ;CLEAR ALL CC
9710 016056 000261 SEC ;SEC CARRY BIT

```

9711	016060	106010				RORB	(R0)	;SHIFT BYTE RIGHT
9712	016062	100002				BPL	RORB10	;CC=127
9713	016064	101401				BLOS	RORB10	
9714	016066	102401				BVS	RORB11	
(1)	016070				RORB10:			
(2)	016070	104000				EMT		;RORB DID NOT SET CC'S CORRECTLY
9715	016072	022737	112525	000000	RORB11:	CMP	#112525,0#0	;CHECK RESULT
9716	016100	001401				BEQ	RORB12	
(2)	016102	104000				EMT		;RORB DID NOT FUNCTION CORRECTLY
9717	016104	106010			RORB12:	RORB	(R0)	;SHIFT BYTE RIGHT
9718	016106	100403				BMI	RORB13	;CC=37
9719	016110	001402				BEQ	RORB13	
9720	016112	102001				BVC	RORB13	
9721	016114	103401				BCS	RORB14	
(1)	016116				RORB13:			
(2)	016116	104000				EMT		;RORB DID NOT SET CC CORRECTLY
9722	016120	022737	045125	000000	RORB14:	CMP	#045125,0#0	;CHECK RESULT
9723	016126	001401				BEQ	TS234	
(3)	016130	104000				EMT		;RORB DID NOT FUNCTION CORRECTLY

9724
9725
9726
(2)
(3)
;*****
;TEST 234 TEST ASLB INSTRUCTION
;*****

(2)	016132				TS234:			
9727	016132	112701	000040			MOVB	#40,R1	;LOAD REGISTER
9728	016136	000257				CCC		;CLEAR ALL CONDITION CODES
9729	016140	106301				ASLB	R1	;SHIFT BYTE LEFT
9730	016142	106301				ASLB	R1	;SHIFT BYTE LEFT
9731	016144	100002				BPL	ASLB2	;CHECK CC=12
9732	016146	101401				BLOS	ASLB2	
9733	016150	102401				BVS	ASLB3	
(1)	016152				ASLB2:			
(2)	016152	104000				EMT		;ASLB DID NOT SET CONDITION CODE CORRECTLY
9734	016154	022701	000200		ASLB3:	CMP	#200,R1	;CHECK RESULT
9735	016160	001401				BEQ	ASLB1	
(2)	016162	104000				EMT		;ASLB DID NOT FUNCTION CORRECTLY
9736	016164	106301			ASLB1:	ASLB	R1	;SHIFT BYTE LEFT
9737	016166	100403				BMI	ASLB4	;CHECK CC=7?
9738	016170	001002				BNE	ASLB4	
9739	016172	102001				BVC	ASLB4	
9740	016174	103401				BCS	TS235	
(2)	016176				ASLB4:			
(3)	016176	104000				EMT		;ASLB DID NOT SET CC'S CORRECTLY

9741
9742
9743
(2)
(3)
;*****
;TEST 235 TEST ASRB INSTRUCTION
;*****

(2)	016200				TS235:			
9744	016200	112701	000004			MOVB	#4,R1	;SET UP STARTING DATA
9745	016204	000257				CCC		;CLEAR ALL CONDITION CODES
9746	016206	106201				ASRB	R1	;SHIFT BYTE RIGHT
9747	016210	106201				ASRB	R1	;SHIFT BYTE RIGHT
9748	016212	122701	000001			CMPB	#1,R1	;CHECK DATA
9749	016216	001401				BEQ	ASRB1	

(2) 014220 104000
9750 016222 106201
9751 016224 100403
9752 016226 001002
9753 016230 102001
9754 016232 103401
(1) 016234
(2) 016234 104000
9755 016236 106201
9756 016240 103401
9757 016242 001401
(1) 016244
(2) 016244 104000
9758 016246 112701 000202
9759 016252 106201
9760 016254 106201
9761 016256 100003
9762 016260 001402
9763 016262 102401
9764 016264 103401
(1) 016266
(2) 016266 104000
9765 016270 122701 000340
9766 016274 001401
(3) 016276 104000
9767
9768
9769
9770
9771
9772
9773
9774
9775
9776
(2)
(3)
(2) 016300
9777 016300 005000
9778 016302 000277
9779 016304 000244
9780 016306 006700
9781 016310 100006
9782 016312 001405
9783 016314 102404
9784 016316 103003
9785 016320 022700 177777
9786 016324 001401
(1) 016326
(2) 016326 104000
9787 016330 005000
9788 016332 005010
9789 016334 005110
9790 016336 000257
9791 016340 000266
9792 016342 006710

EMT ;ASRB DID NOT SHIFT DATA CORRECTLY
ASRB1: ASRB R1 ;SHIFT BYTE RIGHT
BMI ASRB2 ;CHECK CONDITION CODE = 7?
BNE ASRB2
BVC ASRB2
BCS ASRB3
ASRB2:
EMT ;ASRB DID NOT SET CC'S CORRECTLY
ASRB3: ASRB R1 ;SHIFT BYTE RIGHT
BCS ASRB4 ;CHECK CC=4
BEQ ASRB5
ASRB4:
EMT ;ASRB DID NOT SET CC'S CORRECTLY
ASRB5: MOVB #202,R1 ;PUT STARTING DATA IN REGISTER
ASRB R1 ;SHIFT BYTE RIGHT
ASRB R1 ;SHIFT BYTE RIGHT
BPL ASRB6 ;CHECK CC'S =11?
BEQ ASRB6
BVS ASRB6
BCS ASRB7
ASRB6:
EMT ;ASRB DID NOT SET CC'S CORRECTLY
ASRB7: CMPB #340,R1 ;CHECK RESULT
BEQ TS236
EMT ;ASRB DID NOT SHIFT DATA CORRECTLY

; THIS TEST VERIFIES THE SXT INSTRUCTION. CONDITION CODES
; ARE PRESET IN EACH OF THE TWO POSSIBLE CASES. WITH THE N-BIT SET,
; THE TEST CHECKS FOR ALL ONES IN THE DESTINATION. WITH THE N-BIT
; CLEAR, THE DESTINATION SHOULD CONTAIN ALL ZEROES. THE DATA
; IS VERIFIED BY CONDITIONAL BRANCHES.
; *****
; TEST 236 TEST THE SXT INSTRUCTION
; *****
TS236:

CLR R0 ;SET CC=1011
SCC
CLZ
SXT R0 ;TRY SXT
BPL SXT0 ;TEST CC=1001
BEQ SXT0
BVS SXT0
BCC SXT0
CMP #-1,R0 ;CHECK DATA RESULT
BEQ SXT1
SXT0:
EMT ;RESULTS OF SXT INCORRECT
SXT1: CLR R0 ;RO=0
CLR (R0) ;LOC. 0=0
COM (R0) ;LOC. 0=177777
CCC ;SET CC=0110
+SEZ!SEV
SXT (R0)

HS

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07 JAN-85 09:05

MAC:11 30(1046) 07 JAN-85 09:28 PAGE 9 23
T236 TEST THE SXT INSTRUCTION

SEQ 0098

9793 016344 001005
9794 016346 103404
9795 016350 102403
9796 016352 100402
9797 016354 005710
9798 016356 001401
(2) 016360
(3) 016360 104000
9799
9800

BNE SXT2 ;TEST CC=0100
BCS SXT2
BVS SXT2
BMI SXT2
TST (R0)
BEQ TS237

SXT2: EMT ;RESULTS OF SXT INCORRECT
;.....
;

9802
9803
9804
9805
9806
9807
9808
(2)
(3)
(2) 016362
9809 016362 012700 007463
9810 016366 012701 031525
9811 016372 000277
9812 016374 000241
9813 016376 074100
9814 016400 101406
9815 016402 102405
9816 016404 001404
9817 016406 100403
9818 016410 022700 036146
9819 016414 001401
(1) 016416
(2) 016416 104000
9820 016420 010104
9821 016422 000261
9822 016424 000241
9823 016426 074400
9824 016430 101406
9825 016432 102405
9826 016434 001404
9827 016436 100403
9828 016440 022700 007463
9829 016444 001401
(2) 016446
(3) 016446 104000
9830
9831
9832
9833
9834
9835
9836
9837
(2)
(3)
(2) 016450
9838 016450 012700 000525
9839 016454 010004
9840 016456 000277
9841 016460 101002
9842 016462 100001
9843 016464 102401
(1) 016466
(2) 016466 104000
9844 016470 005304
9845 016472 000277

```
THIS TEST VERIFIES THE XOR INSTRUCTION. UNIQUE PATTERNS  
OF ONES AND ZEROS ARE MOVED TO DATA REGISTERS R0 AND R1.  
AFTER THE FIRST XOR INSTRUCTION R0=36146. AN XOR IS THEN  
EXECUTED WITH THIS NEW VALUE AND THE CONTENTS OF R1 TO  
REPRODUCE THE ORIGINAL VALUE IF R0=31525.  
;*****  
;TEST 237 TEST THE XOR INSTRUCTION  
;*****  
TS237:  
MOV #7463,R0 ;SET UP R0  
MOV #31525,R1 ;SET UP R1  
SCC ;SET CC=1110  
CLC  
XOR R1,R0 ;TRY XOR  
BLOS XOR1 ;CC=0000?  
BVS XOR1  
BEQ XOR1  
BMI XOR1  
CMP #36146,R0 ;DATA RESULT CORRECT?  
BEQ XOR2  
XOR1:  
EMT ;  
XOR2: MOV R1,R4  
SEC ;CC=1110  
CLC  
XOR R4,R0 ;TRY XOR MODE 0,0  
BLOS XOR3 ;CC=0000?  
BVS XOR3  
BEQ XOR3  
BMI XOR3  
CMP #7463,R0  
BEQ TS240  
XOR3: EMT ;RESULT OF XOR INCORRECT  
;*****  
; THIS TEST VERIFIES THE SOB INSTRUCTION. R4 IS USED AS A  
;COUNTER WHILE R0 IS THE ADDRESS REGISTER. CONDITIONAL  
;BRANCHES ARE USED TO VERIFY PROPER TRANSFER OF CONTROL  
;WHILE R4 IS CHECKED TO INSURE PROPER DECREMENTING OF R0.  
;*****  
;TEST 240 TEST SOB INSTRUCTION  
;*****  
TS240:  
MOV #525,R0  
MOV R0,R4  
SCC ;SET CC=1111  
SOB1: BHI SOB2 ;CC=1111?  
BPL SOB2  
BVS SOB3  
SOB2: EMT ;  
SOB3: DEC R4 ;COUNT ITERATIONS  
SCC ;CC=1111
```

9846 016474 077007
 9847 016476 101004
 9848 016500 100003
 9849 016502 102002
 9850 016504 005704
 9851 016506 001401
 (2) 016510
 (3) 016510 104000
 9852
 9853
 9854
 9855
 9856
 9857
 9858
 9859
 (2)
 (3)
 (2) 016512
 9860 016512 012706 001000
 9861 016516 012746 125252
 9862 016522 162706 000074
 9863 016526 012705 016544
 9864 016532 012746 006436
 9865 016536 000277
 9866 016540 000116
 9867 016542 104000
 9868 016544 101010
 9869 016546 100007
 9870 016550 102006
 9871 016552 020527 125252
 9872 016556 001003
 9873 016560 022706 001000
 9874 016564 001401
 (1) 016566
 (2) 016566 104000
 9875 016570 012746 052525
 9876 016574 012746 006400
 9877 016600 010605
 9878 016602 004737 016612
 9879 016606 000137 016616
 9880 016612 000205
 9881 016614 104000
 9882 016616 022706 001000
 9883 016622 001003
 9884 016624 022705 052525
 9885 016630 001401
 (2) 016632
 (3) 016632 104000
 9886 177776
 9887
 9888
 9889
 9890
 9891
 9892

S0B R0,S0B1 ;DO SOB W/ R0
 BMI S0B4 ;CHECK CC=1111
 BPL S0B4
 BVC S0B4
 TST R4 ;ITERATION COUNT OK?
 BEQ TS241
 S0B4:
 EMT ;INCORRECT # OF BRANCHES OR CC S CHANGED
 ;*****
 ;
 ; THIS TEST VERIFIES THE MARK INSTRUCTION. THE EFFECTS
 ; OF THE MARK INSTRUCTION ARE SIMULATED BY THE PROGRAM INSTRUCTIONS.
 ; THE CONTENTS OF R5 AND THE STACK POINTER ARE CHECKED AFTER EACH
 ; OF THE TWO ROUTINES IN THE TEST.
 ;
 ;*****
 ;TEST 241 TEST MARK INSTRUCTION
 ;*****
 TS241:
 MOV #STBOT,SP ;PUT R5 VALUE ON STACK
 MOV #125252,-(SP) ;EFFECTIVELY PUT 36 ARGUMENTS ON STACK
 SUB #74,SP ;SET NEW PC IN R5
 MOV #MRK1,R5 ;PUT MARK 36 INST. ON STACK
 MOV #6436,-(SP) ;SET CC=1111
 SCC ;XFER CONTL TO MARK 36 INST. ON STACK
 JMP (SP) ;MARK INST. SHOULD HAVE JUMPED TO MRK1
 EMT ;TEST CC UNAFFECTED
 MRK1: BMI MRK2 ;IE. CC=1111
 BPL MRK2
 BVC MRK2
 CMP R5,#125252 ;CHECK R5 RESTORED FROM STACK
 BNE MRK2
 CMP #STBOT,R6 ;CHECK STACK POINTER READJUSTED CORRECTLY.
 BEQ MRK3
 MRK2: EMT ;RESULTS OF MARK INCORRECT
 MRK3: MOV #52525,-(SP) ;PUT MARK 0 INST. ON STACK
 MOV #6400,-(SP) ;SET ADDR. OF MARK INST. IN R5
 MOV SP,R5 ;DO JSR
 JSR PC,@MRK4 ;
 MRK4: S R5 ;DO RTS WITH R5 TO MARK INST ON STACK
 EMT ;RTS,MARK SEQUENCE FAILED
 MRK5: CMP #STBOT,R6 ;STACK ADJUSTED CORRECTLY
 BNE MRK6 ;IF NOT: BR
 CMP #52525,R5 ;CHECK IF R5 RESTORED FROM STACK
 BEQ TS242
 MRK6: EMT ;RESULTS OF MARK INCORRECT
 PS=177776
 ;*****
 ;
 ; THESE NEXT SEVEN TESTS VERIFY THE MTPS INSTRUCTION IN ALL
 ; MODES. THE PSW IS DEFINED BY AN EQUATE STATEMENT BEFORE THE
 ; FIRST MTPS TEST. IN EACH TEST A PATTERN OF ONES AND
 ; ZEROES IS SET IN A DATA REGISTER AND MOVED TO THE PSW.
 ;

9893
9894
9895
9896
(2)
(3)
(2) 016634
9897 016634 012700 000377
9898 016640 000257
9899 016642 106400
9900 016644 022767 000357 161124
9901 016652 001401
(2) 016654 104000
9902 016656 005000
9903 016660 005010
9904 016662 000277
9905 016664 106410
9906 016666 100403
9907 016670 102402
9908 016672 103401
9909 016674 001001
(2) 016676
(3) 016676 104000
9910
9911
(2)
(3)
(2) 016700
9912 016700 005000
9913 016702 012710 177777
9914 016706 005037 177776
9915 016712 106420
9916 016714 022737 000357 177776
9917 016722 001401
(2) 016724 104000
9918 016726 022700 000001
9919 016732 001401
(3) 016734 104000
9920
9921
(2)
(3)
(2) 016736
9922 016736 012700 000402
9923 016742 005010
9924 016744 012737 052652 000000
9925 016752 005037 177776
9926 016756 106430
9927 016760 022737 000252 177776
9928 016766 001401
(2) 016770 104000
9929 016772 022700 000404
9930 016776 001401
(3) 017000 104000
9931
9932

```

; THE DATA IN THE PSW, AND THE DATA REGISTER ADDRESS,
; ARE CHECKED TO VERIFY PROPER EXECUTION OF THE INSTRUCTION.
;
;*****
;TEST 242 TEST MTPS INSTRUCTION
;*****
TS242:
MOV #377,R0
CCC
MTPS R0
CMP #357,PS
BEQ MTPS1
EMT ;MTPS FAILED
MTPS1: CLR R0
CLR (R0)
SCC ;CC=1111
MTPS (R0) ;TRY MTPS MODE 1
BMI MTPS1A ;CHECK PS
BVS MTPS1A
BCS MTPS1A
BNE TS243
MTPS1A: EMT ;MTPS FAILED
;*****
;TEST 243 TEST MTPS MODE 2
;*****
TS243:
CLR R0 ;R0=0
MOV #-1,(R0) ;LOC. 0=-1
CLR B#PS ;PS=0
MTPS (R0) ;TRY MTPS W/MODE 2
CMP #357,B#PS ;CHECK DATA
BEQ MTPS2
EMT ;DEST. DATA INCORRECT
MTPS2: CMP #1,R0 ;CHECK DEST. REGISTER.
BEQ TS244
EMT ;DEST REGISTER NOT INCREMENTED BY 1
;*****
;TEST 244 TEST MTPS MODE 3
;*****
TS244:
MOV #402,R0 ;R0=402
CLR (R0) ;LOC. 402=0
MOV #52652,B#0 ;LOC. 0=52652
CLR B#PS ;PS=0
MTPS B(R0) ;TRY MTPS W/MODE 3
CMP #252,B#PS ;CHECK DEST. DATA
BEQ MTPS3
EMT ;DEST. DATA INCORRECT
MTPS3: CMP #404,R0 ;CHECK MODE 3 REGISTER.
BEQ TS245
EMT ;MODE 3 REGISTER INCORRECT
;*****
```


(2)
(3)
(2) 017002
9933 017002 012700 000001
9934 017006 012737 125125 000000
9935 017014 005037 177776
9936 017020 106440
9937 017022 022737 000105 177776
9938 017030 001401
(2) 017032 104000
9939 017034 005700
9940 017036 001401
(3) 017040 104000
9941
9942
(2)
(3)
(2) 017042
9943 017042 012700 000404
9944 017046 012737 177400 000000
9945 017054 000277
9946 017056 106450
9947 017060 005737 177776
9948 017064 001401
(2) 017066 104000
9949 017070 022700 000402
9950 017074 001401
(3) 017076 104000
9951
9952
(2)
(3)
(2) 017100
9953 017100 012737 052652 000000
9954 017106 012700 000406
9955 017112 005037 177776
9956 017116 106460 177372
9957 017122 022737 000252 177776
9958 017130 001401
(2) 017132 104000
9959 017134 022700 000406
9960 017140 001401
(3) 017142 104000
9961
9962
(2)
(3)
(2) 017144
9963 017144 012737 052652 000000
9964 017152 012700 000410
9965 017156 005037 177776
9966 017162 106470 177776
9967 017166 022737 000105 177776
9968 017174 001401
(2) 017176 104000
9969 017200 022700 000410

```
; TEST 245          TEST MTPS MODE 4
;*****
TS245:
      MOV          #1,R0          ;R0=1
      MOV          #125125,R0    ;LOC. 0 = 125125
      CLR          @R0           ;PS=0
      MTPS         -(R0)         ;TRY MTPS W/MODE 4
      CMP          #105,@R0      ;CHECK DEST. DATA
      BEQ          MTPS4
      EMT
MTPS4: TST          R0           ;DEST. DATA INCORRECT
      BEQ          TS246         ;CHECK MODE 4 REGISTER
      EMT                       ;MODE 4 REGISTER NOT DECREMENTED BY 1
;*****
; TEST 246          TEST MTPS MODE 5
;*****
TS246:
      MOV          #404,R0       ;R0=404
      MOV          #177400,R0    ;LOC. 0=177400
      SCC
      MTPS         @-(R0)        ;SET ALL COND. CODES
      TST          @R0           ;TRY MTPS W/MODE 5
      BEQ          MTPS5         ;CHECK DEST. DATA.
      EMT                       ;DESTINATION DATA INCORRECT
MTPS5: CMP          #402,R0      ;CHECK MODE 5 REGISTER
      BEQ          TS247
      EMT                       ;MODE 5 REGISTER NOT DECREMENTED BY 2
;*****
; TEST 247          TEST MTPS MODE 6
;*****
TS247:
      MOV          #52652,R0     ;LOC. 0=52652
      MOV          #406,R0       ;R0=406
      CLR          @R0           ;PS=0
      MTPS         -406(R0)      ;TRY MTPS W/MODE 6
      CMP          #252,@R0      ;CHECK DEST. DATA
      BEQ          MTPS6
      EMT                       ;DEST. DATA INCORRECT
MTPS6: CMP          #406,R0      ;CHECK MODE 6 REGISTER
      BEQ          TS250
      EMT                       ;MODE 6 REGISTER MODIFIED
;*****
; TEST 250          TEST MTPS MODE 7
;*****
TS250:
      MOV          #52652,R0     ;LOC. 0=52652
      MOV          #410,R0       ;R0=410
      CLR          @R0           ;PS=0
      MTPS         @-2(R0)       ;TRY MTPS W/MODE 7
      CMP          #105,@R0      ;CHECK DEST. DATA
      BEQ          MTPS7
      EMT                       ;DESTINATION DATA INCORRECT
MTPS7: CMP          #410,R0      ;CHECK MODE 7 REGISTER
```

9970 017204 001401
(3) 017206 104000
9971
9972
9973
9974
9975
9976
9977
9978
9979
9980
(2)
(3)
(2) 017210
9981 017210 012737 000377 177776
9982 017216 106700
9983 017220 022700 177757
9984 017224 001401
(2) 017226 104000
9985
9986 017230 005000
9987 017232 012737 177777 000000
9988 017240 005037 177776
9989 017244 106710
9990 017246 105737 000000
9991 017252 001401
(3) 017254 104000
9992
9993
(2)
(3)
(2) 017256
9994 017256 005000
9995 017260 005010
9996 017262 012737 000377 177776
9997 017270 106720
9998 017272 103003
9999 017274 102402
10000 017276 001401
10001 017300 100401
(1) 017302
(2) 017302 104000
10002 017304 022737 000357 000000
10003 017312 001401
(2) 017314 104000
10004 017316 022700 000001
10005 017322 001401
(3) 017324 104000
10006
10007
(2)
(3)
(2) 017326
10008 017326 012700 000406
10009 017332 005037 000000

```
BEQ TS251  
EMT ;MODE 7 REGISTER MODIFIED  
;*****  
; THESE NEXT SEVEN TESTS VERIFY THE MFPS INSTRUCTION IN ALL  
; MODES. IN EACH TEST, A PATTERN OF ONES AND ZEROES IS MOVED TO THE  
; PSW, AND AN MFPS INSTRUCTION MOVES THE DATA TO A LOCATION SETUP  
; BY R0, EITHER DIRECTLY OR INDIRECTLY. CONDITIONAL BRANCHES ARE  
; USED TO CHECK PROPER ADDRESSING AND DATA.  
;*****  
;TEST 251 TEST MFPS INSTRUCTION  
;*****  
TS251:  
MOV #377,0#PS  
MFPS R0  
CMP #177757,R0  
BEQ MFPS1  
EMT ;MFPS FAILED  
MFPS1: CLR R0  
MOV #-1,0#0  
CLR 0#PS  
MFPS (R0)  
TS18 0#0  
BEQ TS252  
EMT ;MFPS FAILED  
;*****  
;TEST 252 TEST MFPS MODE 2  
;*****  
TS252:  
CLR R0 ;R0=0  
CLR (R0) ;LOC. 0=0  
MOV #377,0#PS ;SET PS=357  
MFPS (R0); ;TRY MFPS W/MODE 2  
BCC MFPS2A ;BR TO ERROR IF C BIT CLEAR  
BVS MFPS2A ;BR TO ERROR IF V BIT SET  
BEQ MFPS2A ;BR TO ERROR IF Z BIT SET  
BMI MFPS2B  
MFPS2A: EMT ;COND. CODES INCORRECT  
MFPS2B: CMP #357,0#0 ;CHECK DEST. DATA  
BEQ MFPS2C  
EMT ;DEST. DATA INCORRECT  
MFPS2C: CMP #1,R0 ;CHECK MODE Z REGISTER  
BEQ TS253  
EMT ;MODE 2 REGISTER NOT INCREMENTED 1  
;*****  
;TEST 253 TEST MFPS MODE 3  
;*****  
TS253:  
MOV #406,R0 ;R0=406  
CLR 0#0 ;LOC. 0=0
```

```
10010 017336 012737 000252 177776      MOV    #252,0#PS      ;PS=252
10011 017344 106730      MFPS   @-(R0)         ;TRY MFPS WITH MODE 3
10012 017346 103403      BCS   MFPS3A         ;BR TO ERROR IF C-BIT SET
10013 017350 102402      BVS   MFPS3A         ;BR TO ERROR IF V-BIT SET
10014 017352 001401      BEQ   MFPS3A         ;BR TO ERROR IF Z-BIT SET
10015 017354 100401      BMI   MFPS3B
      MFPS3A:
      EMT
      MFPS3B: CMP    #125000,0#0      ;CONDITION CODES INCORRECT
      BEQ   MFPS3C         ;CHECK DEST. DATA
      EMT
      MFPS3C: CMP    R0,#410      ;DEST DATA INCORRECT
      BEQ   TS254         ;CHECK MODE 3 REGISTER.
      EMT
      ;MODE 3 REGISTER NOT INCREMENTED BY 2

;*****
;TEST 254 TEST MFPS MODE 4
;*****
TS254:
      MOV    #2,R0      ;R0=2
      CLR    0#0        ;LOC. 0=0
      MOV    #125,0#PS  ;PS=125
      MFPS   -(R0)      ;TRY MFPS W/MODE 4
      BCC   MFPS4A      ;BR TO ERROR IF C-BIT CLEAR
      BVS   MFPS4A      ;BR TO ERROR IF V-BIT SET
      BEQ   MFPS4A      ;BR TO ERROR IF Z-BIT SET
      BPL   MFPS4B
      MFPS4A:
      EMT
      MFPS4B: CMP    #42400,0#0    ;COND. CODES INCORRECT
      BEQ   MFPS4C         ;CHECK DEST. DATA
      EMT
      MFPS4C: CMP    R0,#1      ;DEST. DATA INCORRECT
      BEQ   TS255         ;CHECK MODE 4 REGISTER
      EMT
      ;MODE 4 REGISTER NOT DECREMENTED BY 1

;*****
;TEST 255 TEST MFPS MODE 5
;*****
TS255:
      MOV    #410,R0    ;R0=410
      MOV    #-1,0#0    ;LOC. 0=-1
      CLR    0#PS      ;PS=0
      MFPS   @-(R0)    ;TRY MFPS W/MODE 5
      BCS   MFPS5A      ;BR TO ERROR IF C-BIT SET
      BVS   MFPS5A      ;BR TO ERROR IF V-BIT SET
      BMI   MFPS5A      ;BR TO ERROR IF N-BIT SET
      BEQ   MFPS5B
      MFPS5A:
      EMT
      MFPS5B: CMP    #377,0#0    ;COND. CODES INCORRECT
      BEQ   MFPS5C         ;CHECK DEST. DATA
      EMT
      MFPS5C: CMP    R0,#406    ;DEST DATA INCORRECT
      BEQ   TS256         ;CHECK MODE 5 REGISTER
      EMT
      ;MODE 5 REGISTER NOT DECREMENTED BY 2
```

```

10048
10049
  (2)
  (3)
  (2) 017532
10050 017532 012700 000401
10051 017536 005037 000000
10052 017542 012737 000252 177776
10053 017550 106760 177377
10054 017554 102403
10055 017556 103402
10056 017560 001401
10057 017562 100401
  (1) 017564
  (2) 017564 104000
10058 017566 022737 000252 000000
10059 017574 001401
  (2) 017576 104000
10060 017600 022700 000401
10061 017604 001401
  (3) 017606 104000
10062
10063
  (2)
  (3)
  (2) 017610
10064 017610 012700 000777
10065 017614 005037 000000
10066 017620 012737 000125 177776
10067 017626 106770 177407
10068 017632 102403
10069 017634 103002
10070 017636 001401
10071 017640 100001
  (1) 017642
  (2) 017642 104000
10072 017644 022737 042400 000000
10073 017652 001401
  (2) 017654 104000
10074 017656 022700 000777
10075 017662 001401
  (3) 017664 104000
10076
10077
10078
10079
10080
10081
10082
10083
10084
10085
  (2)
  (3)
  (2) 017666
10086 017666 032737 000001 001020

```

```

;.....
;TEST 256 TEST MFPS MODE 6
;.....
TS256:
      MOV     #401,R0      ;R0=410
      CLR     #00         ;LOC. 0=0
      MOV     #252,#PS    ;PS=252
      MFPS   -401(R0)     ;TRY MFPS W/MODE 6
      BVS    MFPS6A       ;BR TO ERROR IF V-BIT SET
      BCS    MFPS6A       ;BR TO ERROR IF C-BIT SET
      BEQ    MFPS6A       ;BR TO ERROR IF Z-BIT SET
      BMI    MFPS6B
MFPS6A:
      EMT
MFPS6B: CMP     #252,#00   ;COND. CODES INCORRECT
      BEQ    MFPS6C       ;CHECK DEST. DATA
      EMT
MFPS6C: CMP     #401,R0   ;DEST. DATA INCORRECT
      BEQ    TS257       ;CHECK DEST. REGISTER
      EMT                ;DEST. DATA INCORRECT
;.....
;TEST 257 TEST MFPS MODE 7
;.....
TS257:
      MOV     #777,R0     ;R0=777
      CLR     #00         ;LOC. 0=0
      MOV     #125,#PS    ;PS=125
      MFPS   #371(R0)    ;TRY MFPS W/MODE 7
      BVS    MFPS7A       ;BR TO ERROR IF V-BIT SET
      BCC    MFPS7A       ;BR TO ERROR IF C-BIT SET
      BEQ    MFPS7A       ;BR TO ERROR IF Z-BIT SET
      BPL    MFPS7B
MFPS7A:
      EMT
MFPS7B: CMP     #42400,#00 ;CONDITION CODE INCORRECT
      BEQ    MFPS7C       ;CHECK DESTINATION DATA
      EMT
MFPS7C: CMP     #777,R0   ;DEST. DATA INCORRECT
      BEQ    TS260       ;CHECK MODE 7 REGISTER
      EMT                ;MODE 7 REGISTER MODIFIED
;.....
;
; THIS TEST VERIFIES THAT RESET DOES NOT CLEAR THE PSW.
; THE PSW IS LOADED WITH ONES, A RESET IS ISSUED, AND THE
; CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT THEY HAVE NOT
; CHANGED. THIS TEST IS EXECUTED ONLY ONCE EVERY 240 (DECIMAL)
; ITERATIONS OF PROGRAM.
;.....
;TEST 260 TEST THAT RESET DOES NOT CLEAR PSW
;.....
TS260:
      BIT     #1,#ENV     ;ARE WE RUNNING UNDER APT

```

```

(1) 017674 001403
(1) 017676 005737 001006
(2) 017702 001011
(1) 017704
10087 017704 012737 000357 177776
10088 017712 000005
10089 017714 022737 000357 177776
10090 017722 001401
(3) 017724 104000
10091 017726
10092
10093
10094
10095
10096
10097
10098
(2)
(3)
(2) 017726
10099 017726 052767 140000 160042
10100 017734 012706 000001
10101 017740 000241
10102 017742 006106
10103 017744 103376
10104 017746 001404
10105 017750 042767 140000 160020
10106 017756 104000
10107 017760 042767 140000 160010
10108
10109
10110
10111
10112
10113
10114
10115
10116
10117
(2)
(3)
(2) 017766
10118 017766 052767 140000 160002
10119 017774 012706 177777
10120 020000 022706 177777
10121 020004 001404
10122 020006 042767 140000 157762
10123 020014 104000
10124 020016 042767 140000 157752
10125 020024 022706 177777
10126 020030 001001
(2) 020032 104000
10127 020034 005006
10128 020036 052767 140060 157732
10129 020044 022706 177777
10130 020050 042767 140000 157720

```

```

BEQ 708 ;IF NO THEN DO TEST
TST @R6PASS ;IS THIS FIRST PASS
BNE TS261 ;IF NO THEN SHIP TO NEXT TEST

708:
MOV @R57,@R6PS ;MOV ONES TO PSW
RESET ;
CMP @R57,@R6PS ;PSW CORRECT?
BEQ TS261
EML ;RESET ALTERED PSW

REST:
;.....
;
; THE FOLLOWING TEST CHECKS THE INDEPENDENT FUNCTIONING OF BASIC
; DATA PATH COMPONENTS WITH USER MODE SET.
;.....
; TEST 261 TEST USER MODE R6 CAN HOLD A ONE IN EVERY POSITION
;.....
TS261:
BIS @USR,PS ;SET USER MODE
MOV @1,R6 ;SET BIT0
CLC ;CLEAR C-BIT
USP1: ROL R6 ;ROTATE 1 POSITION
BCC USP1 ;BR IF NOT ALL DONE
BEQ USP1A ;BR IF NO BITS PICKED
BIC @USR,PS ;CLEAR USER MODE
EML ;USER MODE R6 PICKED A BIT
USP1A: BIC @USR,PS ;CLEAR USER MODE

;.....
;
; THIS TEST CHECKS THE INDEPENDENT FUNCTIONING OF THE USER
; AND KERNEL MODE R6'S. R6 IS SETUP AND ADDRESSED IN EACH
; OF THE TWO MODES TO VERIFY THAT THE TWO R6'S ARE INDEPENDENT
; OF EACH OTHER.
;.....
; TEST 262 TEST INDEPENDENCE OF USER AND KERNEL MODE R6'S
;.....
TS262:
BIS @USR,PS ;SET USER MODE
MOV @-1,R6 ;SET USER R6 TO ALL ONES
CMP @-1,R6 ;READ AND CHECK USER R6
BEQ USP2 ;BR IF NO ERROR
BIC @USR,PS ;CLEAR USER MODE
EML ;USER R6 WILL NOT HOLD ALL ONES
USP2: BIC @USR,PS ;SET KERNEL MODE
CMP @-1,R6 ;KERNEL MODE R6 ADDR. FROM USER MODE?>>
BNE USP3
EML ;DUAL ADDRESSING ERROR USER/KERNEL R6
USP3: CLR R6 ;CLEAR KERNEL MODE SP
BIS @USR,PS ;SET USER MODE
CMP @-1,R6 ;CHECK USER R6 NOT ADDR. FROM KERNEL MODE
BIC @USR,PS ;CLEAR USER MODE

```

```
10131 020056 001401          BEQ     USP4          ;BR IF NO ERROR
10132 020060 104000          EMT                    ;DUAL ADDRESSING ERROR OR SEQUENCE ERROR
10133 020062 012706 001000    USP4:  MOV     #STBOT,R6 ;RESTORE SP USER
10134 020066 042767 140000    157702 BIC     @USRM,PS     ;SET KERNEL MODE
10135 020074 012705 001000          MOV     #STBOT,R6     ;RESTORE SP KERNEL
10136
10137 ;.....
10138 ;
10139 ;   THESE NEXT TWO TESTS VERIFY MFPI AND MTPI INSTRUCTIONS
10140 ;WITH R6 IN MODE 0.
10141 ;
10142 ;.....
10143 (2) ;TEST 263      TEST MFPI WITH R6 IN MODE 0
10144 (3) ;.....
10145 (2) 020100 TS263:
10146 020100 012706 001000          MOV     #STBOT,R6     ;INITIALIZE KERNEL STACK POINTER
10147 020104 012767 140000    157664 MOV     @USRM,PS     ;SET USER MODE/PREVIOUS KERNEL
10148 020112 012706 000600          MOV     #USESTK,R6   ;INITIALIZE USER STACK POINTER
10149 020116 006506          MFPI    R6           ;TRY MFPI WITH MODE 0
10150 020120 022767 140000    157650 CMP     @140000,PS   ;CHECK PSW
10151 020126 001404          BEQ     MFPI0        ;BR IF NO ERROR
10152 020130 042767 140000    157640 BIC     @USRM,PS     ;CLEAR USER MODE
10153 020136 104000          EMT                    ;INCORRECT PSW FROM MFPI
10154 020140 042767 140000    157630 MFPI0: BIC     @USRM,PS ;CLEAR USER MODE
10155 020146 022767 001000    160422 CMP     #STBOT,USESTK-2 ;CHECK DATA ON STACK
10156 020154 001401          BEQ     MFPI0A       ;BR IF NO ERROR
10157 020156 104000          EMT                    ;INCORRECT DATA FROM MFPI
10158 MFPI0A:
10159 ;.....
10160 (2) ;TEST 264      TEST MTPI WITH R6 IN MODE 0
10161 (3) ;.....
10162 (2) 020160 TS264:
10163 020160 005067 157612          CLR     PS           ;SET KERNEL MODE
10164 020164 005006          CLR     R6          ;INITIALIZE KERNEL R6
10165 020166 012767 140000    157602 MOV     @USRM,PS     ;SET USER MODE/PREVIOUS KERNEL
10166 020174 012706 000600          MOV     #USESTK,R6   ;INITIALIZE USER STACK POINTER
10167 020200 012746 001000          MOV     #STBOT,-(R6) ;SET UP TARGET DATA
10168 020204 006606          MTPI   R6           ;TRY MODE 0 MTPI
10169 020206 022767 140000    157562 CMP     @USRM,PS     ;CHECK PSW
10170 020214 001404          BEQ     MTPI0        ;BR IF NO ERROR
10171 020216 042767 140000    157552 BIC     @USRM,PS     ;CLEAR USER MODE
10172 020224 104000          EMT                    ;PS INCORRECT FOLLOWING MTPI
10173 020226 005067 157544    MTPI0: CLR     PS           ;SET KERNEL MODE
10174 020232 020627 001000          CMP     R6,#STBOT    ;CHECK TARGET DATA
10175 020236 001401          BEQ     TS265        ;DATA INCORRECT FOLLOWING MTPI
10176 020240 104000          EMT
10177
10178 ;.....
10179 ; THE FOLLOWING TEST VERIFIES THAT NO DUAL ADDRESSING OF THE GENERAL
;REGISTERS OCCURS. ALL REGISTERS ARE CLEARED, AND A UNIQUE BIT IS SET
;IN EACH. CMP INSTRUCTIONS CHECK THAT ONLY ONE BIT IS SET IN EACH
;REGISTER.
```

```

10180
10181
(2)
(3)
(2) 020242
10182 020242 005000
10183 020244 005001
10184 020246 005002
10185 020250 005003
10186 020252 005004
10187 020254 005005
10188 020256 005006
10189 020260 052700 000001
10190 020264 052701 000002
10191 020270 052702 000004
10192 020274 052703 000010
10193 020300 052704 000020
10194 020304 052705 000040
10195 020310 052706 000100
10196 020314 022706 000100
10197 020320 001022
10198 020322 022705 000040
10199 020326 001017
10200 020330 022704 000020
10201 020334 001014
10202 020336 022703 000010
10203 020342 001011
10204 020344 022702 000004
10205 020350 001006
10206 020352 022701 000002
10207 020356 001003
10208 020360 022700 000001
10209 020364 001401
(1) 020366
(2) 020366 104000
10210 020370 012702 001004
10211
10212
10213
10214
10215
10216
10217
10218
(2)
(3)
(2) 020374
10219 020374 052737 170357 177776
10220 020402 105037 177776
10221 020406 013700 177776
10222 020412 032700 170000
10223 020416 001003
10224 020420 005037 177776
10225 020424 104000
10226 020426 005037 177776
10227

```

```

;
;*****
;TEST 265      DUAL REGISTER ADDRESSING TEST
;*****
TS265:
BITCLR: CLR      R0      ;INITIALIZE ALL REGISTERS
        CLR      R1
        CLR      R2
        CLR      R3
        CLR      R4
        CLR      R5
        CLR      R6
BITSET: BIS      #1,R0   ;SET R0=1
        BIS      #2,R1   ;R1=2
        BIS      #4,R2   ;R2=4
        BIS      #10,R3  ;R3=10
        BIS      #20,R4  ;R4=20
        BIS      #40,R5  ;R5=40
        BIS      #100,R6 ;R6=100
BITCHK: CMP      #100,R6 ;TEST THAT NO DUAL ADDRESSING OCCURRED
        BNE      DAERR   ;BR TO ERROR HALT IF ANY OTHER BITS ARE SET
        CMP      #40,R5
        BNE      DAERR
        CMP      #20,R4
        BNE      DAERR
        CMP      #10,R3
        BNE      DAERR
        CMP      #4,R2
        BNE      DAERR
        CMP      #2,R1
        BNE      DAERR
        CMP      #1,R0
        BEQ      BITCON
DAERR:  EMT
BITCON: MOV      #TESTN,R2 ;DUAL ADDRESSING ERROR
        ;RESTORE POINTER
;*****
;      THIS TEST VERIFIES THAT THE UPPER BYTE OF THE PSW IS NOT AFFECTED
;      WHEN THE PRIORITY LEVEL OR CC'S ARE CHANGED. ALL BITS ARE
;      INITIALLY SET IN THE PSW, AND THE LOW BYTE IS CLEARED. A BIT
;      INSTRUCTION VERIFIES THE DATA.
;*****
;TEST 266      TEST BYTE INSTRUCTION ON PSW
;*****
TS266:
        BIS      #170357,#PS ;SET ALL POSSIBLE BITS IN PSW
        CLR      #PS        ;CLR PR LEVEL AND CC'S
        MOV      #PS,R0     ;COPY CONTENTS OF PSW
        BIT      #170000,R0 ;TEST THAT UPPER BYTE IS UNAFFECTED
        BNE      BITCON    ;CONTINUE IF OK
BTERR:  CLR      #PS        ;RETURN TO KERNEL MODE
        EMT
BITCON: CLR      #PS        ;BYTE INSTRUCTION ALTERED PSW
        ;RETURN TO KERNEL MODE

```

10228
10229
10230
10231
10232
10233
10234
(2)
(3)
(2) 020432
:0235 020432 000277
10236 020434 000252
10237 020436 000167 00000C
10238 020442 100403
10239 020444 001002
10240 020446 102401
10241 020450 103401
(2) 020452
(3) 020452 104000
10242
10243
10244
10245
10246
10247
10248
10249
10250
10251
10252
10253
10254
10255
10256
(2)
(3)
(2) 020454
10257 020454 012767 000240 000024
10258 020462 012767 000017 000032
10259 020470 012767 000261 000074
10260 020476 012767 000001 000102
10261 020504 000277
10262 020506 000000
10263 020510 013704 177776
10264 020514 042704 177760
10265 020520 022704
10266 020522 000000
10267 020524 001401
(2) 020526 104000
10268 020530 005367 177766
10269 020534 005267 177746
10270 020540 026727 177742 000257
10271 020546 003756
10272 020550 026727 177732 000260
10273 020556 001004
10274 020560 012767 000017 177734

```
.....  
: THIS TEST VERIFIES THAT A JMP INSTRUCTION DOES NOT ALTER THE  
; CONDITION CODES IN THE PSW. THE CC'S ARE PRESET, THE JMP IS  
; EXECUTED, AND CONDITIONAL BRANCHES VERIFY THE STATE OF THE CC'S.  
: .....
```

TS267: TEST THAT JMP INSTRUCTION DOES NOT AFFECT CONDITION CODES
:

```
TS267:      SCC          ;CC=0101  
           +CLN!CLV      ;JUMP TO TEST PSW  
JMP:      JMP          ;BR TO ERROR HALT IF N-BIT IS SET  
           BMI          ;BR TO ERROR HALT IF Z-BIT IS CLEAR  
           BNE          ;BR TO ERROR HALT IF V-BIT IF SET  
           BVS          ;BR TO ERROR HALT IF V-BIT IF SET  
           BCS          ;BR TO ERROR HALT IF V-BIT IF SET  
           TS270  
JMPERR:   EMT          ;JMP INSTRUCTION AFFECTED CC'S  
: .....
```

THIS TEST VERIFIES THE SET AND CLEAR CONDITION CODE INSTRUCTIONS.
THE TEST CONSISTS OF TWO ROUTINES, ONE TO TEST ALL CLEAR CC
INSTRUCTIONS, AND THE SECOND TO TEST ALL SET CC INSTRUCTIONS. ALL
POSSIBLE COMBINATIONS OF CONDITION CODES ARE TESTED, INCLUDING NOP'S.
TO TEST THE CLEAR CC INSTRUCTIONS, ALL CONDITION CODES ARE
INITIALLY SET. THE INSTRUCTION IS EXECUTED, AND THE PSW IS CHECKED
TO VERIFY THE PROPER COMBINATION OF CONDITION CODES.
TO TEST THE SET CC INSTRUCTIONS, THE CONDITION CODES ARE
INITIALLY CLEARED, AND ONLY THE REQUIRED BITS ARE SET BY THE SET CC
INSTRUCTION. THE CONTENTS OF THE PSW ARE CHECKED TO VERIFY THAT
ONLY THE REQUIRED BITS WERE SET.
:

TS270: TEST SET CC AND CLEAR CC INSTRUCTIONS
:

```
TS270:      MOV          #240,CC3      ;INITIALIZE CLR CC INSTRUCTION CODES  
           MOV          #17,CC2       ;INITIALIZE OCTAL MAP  
           MOV          #261,SC3      ;INITIALIZE SET CC INSTRUCTION CODES  
           MOV          #1,SC4        ;INITIALIZE OCTAL MAP  
CLRCD:     SCC          ;SET ALL CONDITION CODES  
CC3:       0            ;CONDITION CODE INSTRUCTION  
           MOV          @PS,R4        ;COPY THE PSW  
           BIC          #177760,R4    ;ISOLATE CONDITION CODES  
           CMP          (PC)+,R4      ;CHECK THAT PROPER CC'S WERE CLEARED  
CC2:       0            ;OCTAL REPRESENTATION OF CC'S  
           BEQ          CON1  
           EMT  
CON1:     DEC          CC2            ;CLEAR CC INSTRUCTION FAILED  
           INC          CC3            ;SET NEXT OCTAL MAP OF CC'S  
           CMP          CC3,#257      ;GET NEXT CLEAR CC INSTRUCTION  
           BLE          CLRCD         ;TEST FOR CCC INSTRUCTION  
           CMP          CC3,#260      ;GO TEST NEXT INSTRUCTION IF NOT FOUND  
           BNE          SETCD         ;CHECK FOR NOP=260  
           MOV          #17,CC2       ;GO TEST SET CC INSTRUCTIONS  
           MOV          #17,CC2       ;SET OCTAL MAP TO TEST NOP
```


CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07 JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 10 11
T270 TEST SET CC AND CLEAR CC INSTRUCTIONS

SEQ 0110

10275	020566	000746				BR	CLRCD		;GO TEST NOP
10276	020570	000257				SETCD:	CCC		;CLEAR ALL CONDITION CODES
10277	020572	000000				SC3:	0		;CONDITION CODE INSTRUCTION
10278	020574	013704	177776				MOV	@PS,R4	;COPY PSW
10279	020600	042704	177760				BIC	@177760,R4	;CLEAR AWAY UNWANTED BITS
10280	020604	022704					CMP	(PC),R4	;CHECK THAT PROPER CC'S WERE SET
10281	020606	000000				SC4:	0		;OCTAL REPRESENTATION OF CC'S
10282	020610	001401					BEQ	CON2	
(1)	020612					CCERR:			
(2)	020612	104000					EMT		;SET CC FAILED OR SEQUENCE ERROR
10283	020614	005267	177766			CON2:	INC	SC4	;SET NEXT OCTAL MAP
10284	020620	005267	177746				INC	SC3	;PREPARE NEXT SET CC INSTRUCTION
10285	020624	026727	177742	000277			CMP	SC3,@277	;FINISHED?
10286	020632	003756					BLE	SETCD	;BR IF NO
10287	020634	000167	000006				JMP	MORO	;JUMP TO NEXT TESTS

```

10289
10290
10291
10292
10293
10294
10295
10296
10297
10298 020640 000000 000000 000000
10299 020646
(3)
(2)
(3)
(2) 020646
10300 020646 005037 020640
10301 020652 012700 020640
10302 020656 060020
10303
10304 020660 022700 020642
10305 020664 001401
(2) 020666 104000
10306
10307 020670 022737 020642 020640 MOR1:
10308
10309
10310 020676 001401
(3) 020700 104000
10311
10312
(2)
(3)
(2) 020702
10313 020702 005037 020640
10314 020706 012700 020642
10315 020712 060040
10316
10317 020714 022700 020640
10318 020720 001401
(2) 020722 104000
10319
10320 020724 022737 020640 020640 MOR2:
10321
10322
10323 020732 001401
(3) 020734 104000
10324
10325
(2)
(3)
(2) 020736
10326 020736 005037 020640
10327 020742 005037 020644
10328 020746 012737 020640 020642
10329 020754 012700 020642
10330 020760 060030

```

```

;*****
;SBTTL TEST INSTRUCTIONS USING SAME REGISTER FOR SOURCE & DESTINATION
;
;IN AUTO INCREMENT (DECREMENT) MODES AND
;AUTO INCREMENT (DECREMENT) DEFERRED MODES,
;CONTENTS OF THE REGISTER IN USED ARE
;INCREMENTED (DECREMENTED) BY 2
;BEFORE USED AS THE SOURCE OPERAND.
;
A: .WORD 0,0,0
MOR0:
;*****
;TEST 271 TEST AUTO-INCREMENT MODE, USING RO
;*****
TS271:
CLR @A ;CLEAR LOC A
MOV @A,RO ;RO STORES ADDR OF A
ADD RO,(RO)+ ;CHECK THAT RO IS INCR BY 2 BEFORE
;BEING USED AS THE SOURCE OPERAND
;RO INCR BY 2?
CMP @A+2,RO
BEQ MOR1
EMT ;RO WAS NOT INCREMENTED BY 2
;
MOR1: CMP @A+2,@A ;CHECK CONTENT OF RO WAS INCR BY 2 BEFORE
;BEING USED IN THE "ADD" INSTR
;LOC A CONTAINS (A+2)?
BEQ TS272
EMT ;WRONG SUM IN LOC A
;
;*****
;TEST 272 AUTO-DECREMENT MODE, USING RO
;*****
TS272:
CLR @A ;CLEAR LOC A
MOV @A+2,RO ;RO STORES ADDR OF A+2
ADD RO,-(RO) ;CHECK THAT RO IS DECR BY 2 BEFORE
;BEING USED AS THE SOURCE OPERAND
;RO DECR BY 2?
CMP @A,RO
BEQ MOR2
EMT ;RO WAS NOT DECREMENTED BY 2
;
MOR2: CMP @A,@A ;CONTENT OF RO WAS DECR BY 2 BEFORE
;BEING USED IN THE "ADD" INSTR
;LOC A CONTAINS (RO)
BEQ TS273
EMT ;WRONG SUM IN LOC A
;
;*****
;TEST 273 TEST AUTO-INCREMENT DEFERRED MODE, USING RO
;*****
TS273:
CLR @A ;CLEAR LOC A
CLR @A+4 ;CLEAR LOC A+4
MOV @A,@A+2 ;STORE ADDR A IN LOC A+2
MOV @A+2,RO ;RO STORES ADDR A+2
ADD RO,@(RO)+ ;CHECK THAT RO IS INCR BY 2 BEFORE

```


10367
10368
10369
10370
10371
10372
10373
10374
(2)
(3)
(2) 021116
10375 021116 012700 177777
10376 021122 010700
10377 021124 022700 021124
10378 021130 001401
(3) 021132 104000
10379
10380
(2)
(3)
(2) 021134
10381 021134 012700 020640
10382 021140 010760 000004
10383 021144 022737 021144 020644
10384 021152 001401
(3) 021154 104000
10385
10386
(2)
(3)
(2) 021156
10387 021156 012737 020640 020644
10388 021164 012700 020640
10389 021170 010770 000004
10390 021174 022737 021174 020640
10391 021202 001401
(3) 021204 104000
10392
10393
(2)
(3)
(2) 021206
10394 021206 012737 020642 020640
10395 021214 010777 177420
10396 021220 022737 021220 020642
10397 021226 001401
(3) 021230 104000
10398
10399
(2)
(3)
(2) 021232
10400 021232 005037 020640
10401 021236 010767 177376
10402 021242 022737 021242 020640
10403 021250 001401

```
.....  
;SBTTL INSTRUCTION USING PC AS SOURCE REGISTER  
;  
;IN INDEX, INDEX DEFERRED, RELATIVE, AND  
;RELATIVE DEFERRED MODES, DESTINATION WILL CONTAIN  
;THE PC COUNT OF THE CURRENT INSTRUCTION +4.  
;  
.....  
;TEST 275 TEST PC AS SOURCE IN MODE 0, USING RO  
;.....  
TS275:  
PCN01: MOV #1,RO ;SET ALL 1 IN RO  
MOV PC,RO ;STORES PC IN RO  
CMP #PCN01+2,RO ;RO STORES PC+2?  
BEQ TS276  
EMT ;RO STORED WRONG VALUE  
.....  
;TEST 276 TEST PC AS SOURCE IN MODE 6, USING RO  
;.....  
TS276:  
PCN2: MOV #A,RO ;RO STORES ADDR A  
MOV PC,4(RO) ;EFFECTIVE ADDR IS A+4  
CMP #PCN2+4,#A+4 ;LOC A+4 STORES PC+4?  
BEQ TS277  
EMT ;LOC A+4 STORED WRONG VALUE  
.....  
;TEST 277 TEST PC AS SOURCE IN MODE 7, USING RO  
;.....  
TS277:  
PCN3: MOV #A,#A+4 ;LOC A+4 STORES ADDR A  
MOV #A,RO ;RO STORES ADDR A  
MOV PC,#A(RO) ;EFFECTIVE ADDR IS A  
CMP #PCN3+4,#A ;LOC A STORES PC+4?  
BEQ TS300  
EMT ;LOC A STORED WRONG VALUE  
.....  
;TEST 300 TEST PC AS SOURCE IN RELATIVE DEFERRED MODE ,USING RO  
;.....  
TS300:  
PCN4: MOV #A+2,#A ;LOC A STORES ADDR A+2  
MOV PC,#A ;EFFECTIVE ADDR IS A+2  
CMP #PCN4+4,#A+2 ;LOC A+2 STORES PC+4?  
BEQ TS301  
EMT ;LOC A+2 STORED WRONG VALUE  
.....  
;TEST 301 TEST PC AS SOURCE IN RELATIVE MODE ,USING RO  
;.....  
TS301:  
PCN5: CLR #A ;CLEAR A  
MOV PC,A ;EFFECTIVE ADDR IS A  
CMP #PCN5+4,#A ;LOC A STORES PC+4?  
BEQ TS302
```

(3) 021252 104000
10404
10405
10406
10407
10408
10409
(2)
(3)
(2) 021254
10410 000007
10411 021254 012706 001000
10412 021260 000007
10413 021262 022700 000003
10414 021266 001401
(3) 021270 104000
10415
10416
10417
10418
10419
10420
(2)
(3)
(2) 021272
10421 021272 012737 052525 000000
10422 021300 012701 050505
10423 021304 005000
10424 021306 160120
10425 021310 022737 002020 000000
10426 021316 001401
(3) 021320 104000
10427
10428
(2)
(3)
(2) 021322
10429 021322 012737 052525 000000
10430 021330 005000
10431 021332 012767 170000 156436
10432 021340 012706 000600
10433 021344 106520
10434 021346 005067 156424
10435 021352 022767 052525 157216
10436 021360 001401
(3) 021362 104000
10437
10438
(2)
(3)
(2) 021364
10439 021364 012767 170000 156404
10440 021372 012706 000600
10441 021376 012746 125252
10442 021402 012737 000000 000000
10443 021410 005000

```
EMT ;LOCATION A STORED WRONG VALUE
;*****
;THIS TESTS THE MOVE FROM PROCESSOR TYPE INSTRUCTION(MFPT)
;UPON EXECUTION R0 WILL RECIEVE THE PROCESSOR MODEL CODE
;WHICH IS "000003" FOR THE DCF11-AA
;*****
;TEST 302 TEST MFPT
;*****
TS302:
MFPT=000007
MOV #STBOT,SP ;INITIALIZE STACK POINT IN CASE OF TRAP
MFPT ;GET MODEL CODE.IF THIS TRAPS AN ERROR WILL BE REPORTED
CMP #3,R0 ;CHECK IF CORRECT CODE RETURNED
BEQ TS303
EMT ;WRONG CODE RETURNED

;*****
.SBTL THE NEXT THREE TESTS EXERCISE MASKING ACTION OF MICROCODES.
;*****
;TEST 303 TEST SUB INSTRUCTION, SM=0, DM=2
;*****
TS303:
MOV #052525,B#0 ;SET UP LOC 0
MOV #050505,R1 ;SET UP R1
CLR R0 ;CLEAR R0
SUB R1,(R0)+ ;SUBTRACTION, SM=0,DM=2
CMP #2020,B#0 ;CHECK DIFFERENCE AT LOC 0
BEQ TS304
EMT ;WRONG RESULT FROM SUBTRACTION

;*****
;TEST 304 TEST MFPD WITH R0, IN MODE 2
;*****
TS304:
MOV #052525,B#0 ;SET UP LOC 0
CLR R0 ;CLEAR R0
MOV #170000,PS ;SET USER MODE ON, CURRENT & PREVIOUS
MOV #USESTK,R6 ;SET USER STACK POINTER
MFPD (R0)+ ;MODE 2, MFPD
CLR PS ;SET KERNEL MODE
CMP #052525,USESTK-2 ;CHECK DATA ON STACK
BEQ TS305
EMT ;INCORRECT DATA FROM MFPD

;*****
;TEST 305 TEST MTPD WITH R0, IN MODE 2
;*****
TS305:
MOV #170000,PS ;SET USER MODE ON, CURRENT & PREVIOUS
MOV #USESTK,R6 ;SET USER STACK POINTER
MOV #125252,-(R6) ;PUSH DATA IN USER STACK
MOV #0,B#0 ;CLEAR LOC 0
CLR R0 ;CLEAR R0
```

10444 021412 106620
 10445 021414 005067 156356
 10446 021420 022737 125252 000000
 10447 021426 001464
 (2) 021430 104000
 10448
 10449 021432 000402
 10450 021434 001002
 10451 021436 001402
 10452 021440 002002
 10453 021442 002402
 10454 021444 003002
 10455 021446 003402
 10456 021450 100002
 10457 021452 100402
 10458 021454 101002
 10459 021456 101402
 10460 021460 102002
 10461 021462 102402
 10462 021464 103002
 10463 021466 103402
 10464
 10465 000002
 10466 021470 177777
 10467 021472 170360
 10468 021474 007417
 10469 021476 146063
 10470 021500 031714
 10471 021502 140060
 10472 021504 037717
 10473
 10474 021506 177400
 10475 021510 000377
 10476 021512 120240
 10477 021514 057537
 10478 021516 146314
 10479 021520 031463
 10480 021522 125252
 10481 021524 052525
 10482 000010
 10483
 10484
 10485
 10486
 10487
 10488
 10489
 10490 021526
 (2) 021526 104000
 10491 021530
 (2) 021530 104000
 10492 021532
 (2) 021532 104000
 10493 021534
 (2) 021534 104000
 10494 021536

MTPD (RO).
 CLR PS
 CMP #125252, @#0
 BEQ TESTN1
 EMT

BRTAB: BR .+6
 BNE .+6
 BEQ .+6
 BGE .+6
 BLT .+6
 BGT .+6
 BLE .+6
 BPL .+6
 BMI .+6
 BHI .+6
 BLOS .+6
 BVC .+6
 BVS .+6
 BCC .+6
 BCS .+6

.RADIX 2
 YNTAB: 1111111111111111
 1111000011110000
 0000111100001111
 1100110000110011
 0011001111001100
 1100000000110000
 0011111111001111
 1111111100000000
 0000000011111111
 1010000010100000
 0101111101011111
 1100110011001100
 0011001100110011
 1010101010101010
 0101010101010101

.RADIX 8

;MODE 2, MTPD
 ;SET KERNEL MODE
 ;CHECK DATA ON LOC 0
 ;INCORRECT DATA FROM MTPD

;SAME AS BHIS
 ;SAME AS BLO

;BR
 ;BNE: Z=0
 ;BEQ: Z=1
 ;BGE: N XOR V =0
 ;BLT: N XOR V =1
 ;BGT: Z+(N XOR V) =0
 ;BLE: Z+(N XOR V) =1
 ;BPL: N=0
 ;BMI: N=1
 ;BHI: C+Z=0
 ;BLOS: C+Z=1
 ;BVC: V=0
 ;BVS: V=1
 ;BCC: C=0
 ;BCS: C=1

 ; THE FOLLOWING ARE SPECIAL CPU TRAP
 ; HANDLERS TO TRAP AND REPORT SPECIAL TRAPS.
 ;

T04: EMT ;TRAPPED THRU LOC. 4
 T010: EMT ;TRAPPED THRU LOC. 10
 T014: EMT ;TRAPPED THRU LOC. 14
 T020: EMT ;TRAPPED THRU LOC. 20
 T030:

(2) 021536 104000
 10495 021540
 (2) 021540 104000
 10496 021542
 (2) 021542 104000
 10497 021544
 (2) 021544 104000
 10498 021546
 (2) 021546 104000
 10499 021550
 (2) 021550 104000
 10503
 10504
 10505
 10506 000000
 10507
 10508 021552 000000
 10509 021554 000000
 10510 021556 000000
 10511 021560 000000
 10512 021562 000000
 10513 021564 000000
 10514 021566 052525
 10515 021570 052400
 10516 021572 000000
 10517 021574 000000
 10518 021576 000176
 10519
 10520 021600 032737 000001 001020
 10521 021606 001403
 10522 021610 012767 001022 177760
 10523 021616
 (3)
 (2)
 (3)
 (2) 021616
 10524 021616 005006
 10525 021620 112667 156154
 10526 021624 020627 000002
 10527 021630 001401
 (2) 021632 104000
 10528
 10529 021634 012706 001000
 10530 021640 114627 000000
 10531 021644 020627 000776
 10532 021650 001401
 (2) 021652 104000
 10533
 10534 021654 005006
 10535 021656 112626
 10536 021660 020627 000004
 10537 021664 001401
 (2) 021666 104000
 10538
 10539 021670 005006
 10540 021672 005004

EMT ; TRAPPED THRU LOC. 30
 T034:
 EMT ; TRAPPED THRU LOC. 34
 T040:
 EMT ; TRAPPED THRU LOC. 40
 T0114:
 EMT ; TRAPPED THRU LOC. 114
 T0244:
 EMT ; TRAPPED THRU LOC. 244
 T0250:
 EMT ; TRAPPED THRU LOC. 250

.SBTTL ** STARTING OF TRAP TEST **
 ;SPECIAL CASE OF ODD;.EVEN .BYTE AND REGISTER 6
 HERE=0

K1: 0
 K2: 0
 K3: 0
 K4: 0
 K5: 0
 K6: 0
 K7: 052525
 K10: 052400
 K11: 0
 K12: 0
 SWR: 176

TESTN1: BIT #1, @ENV
 BEQ 14
 MOV @SWREG, SWR

14:
 ;*****
 ;TEST 306 TEST AUTO INCREMENT AND DECREMENT OF R6 FOR WORD AND BYTES
 ;*****
 TS306:

CLR #6
 MOVB (6)+, HERE ;SIX SHOULD INCREMENT BY TWO
 CMP #6, #2
 BEQ BR1
 EMT ;R6 DID NOT AUTO INCREMENT BY TWO
 BR1: MOV #1000, #6
 MOVB -(6), @HERE ;SHOULD DECREMENT BY TWO
 CMP #6, #776
 BEQ BR2
 EMT ;R6 DID NOT AUTO DECREMENT BY 2
 BR2: CLR #6
 MOVB (6)+, (6)+ ;DOUBLES AUTO INCREMENT OF R6
 CMP #6, #4
 BEQ BR3
 EMT ;WRONG AUTO INCREMENT OF R6
 BR3: CLR #6
 CLR #4

Address	OpCode	Operand1	Operand2	Operand3	Comment	Label	OpCode	Operand1	Operand2	Operand3	Comment
10541	021674	122624					CMPB	(6)+,(4)+			;TEST INCREMENT OF R6
10542	021676	020627	000002				CMP	#6,#2			
10543	021702	001401					BEQ	BR4			
(2)	021704	104000					EMT				;WRONG INCREMENT OF R6
10544											
10545	021706	005006				BR4:	CLR	#6			
10546	021710	005004					CLR	#4			
10547	021712	122426					CMPB	(4)+,(6)+			;TEST INCREMENT OF R6
10548	021714	020627	000002				CMP	#6,#2			
10549	021720	001401					BEQ	BR5			
(2)	021722	104000					EMT				;WRONG INCREMENT OF R6
10550											
10551	021724	005006				BR5:	CLR	#6			
10552	021726	005004					CLR	#4			
10553	021730	122624					CMPB	(6)+,(4)+			;TEST INCREMENT OF R4
10554	021732	020427	000001				CMP	#4,#1			
10555	021736	001401					BEQ	BR6			
(2)	021740	104000					EMT				;WRONG INCREMENT OF R4
10556	021742	005006				BR6:	CLR	#6			
10557	021744	005004					CLR	#4			
10558	021746	122426					CMPB	(4)+,(6)+			;TEST INCREMENT OF R6
10559	021750	020627	000002				CMP	#6,#2			
10560	021754	001401					BEQ	BR7			
(2)	021756	104000					EMT				;WRONG INCREMENT OF R6
10561											
10562	021760	005006				BR7:	CLR	#6			
10563	021762	005004					CLR	#4			
10564	021764	122426					CMPB	(4)+,(6)+			;TEST INCREMENT OF R4
10565	021766	020427	000001				CMP	#4,#1			
10566	021772	001401					BEQ	BR10			
(2)	021774	104000					EMT				;WRONG INCREMENT OF R4
10567											
10568	021776	012706	001000			BR10:	MOV	#1000,#6			
10569	022002	124627	000000				CMPB	-(6),#HERE			;TEST DECREMENT OF R6
10570	022006	022706	000776				CMP	#776,#6			
10571	022012	001401					BEQ	TS307			
(3)	022014	104000					EMT				;WRONG DECREMENT OF R6,OR WRONG \$TSTNM
10572											
(2)											
(3)											
(2)	022016										
10573	022016	012767	123456	177536			MOV	#123456,K5			
10574	022024	012767	050505	177520			MOV	#050505,K1			
10575	022032	012705	021552				MOV	#K1,#5			;#5=(050505)K1
10576	022036	012706	021562				MOV	#K5,#6			;#6=(123456)K5
10577	022042	112625					MOVB	(6)+,(5)+			;LOW .BYTE OF R6 TO R5
10578	022044	022767	050456	177500			CMP	#050456,K1			
10579	022052	001401					BEQ	BR11			
(2)	022054	104000					EMT				;FALSE TRANSFER OF .BYTE
10580											
10581	022056	012767	123456	177476		BR11:	MOV	#123456,K5			
10582	022064	012767	050505	177460			MOV	#050505,K1			
10583	022072	012705	021552				MOV	#K1,#5			;#5(050505)K1
10584	022076	012706	021564				MOV	#K6,#6			;#6(123456)K5
10585	022102	114625					MOVB	-(6),(5)+			;LOW .BYTE OF R6 TO R5 (DECREMENT)
10586	022104	026727	177442	050456			CMP	K1,#050456			


```

10587 022112 001401          BEQ      BR12
(2) 022114 104000          EMT                ;FALSE R6 .BYTE TRANSFER
10588
10589 022116 012767 123456 177426 BR12:  MOV     #123456,K1
10590 022124 012767 050505 177430  MOV     #050505,K5
10591 022132 012705 021552          MOV     #K1,#5      ;(123456)
10592 022136 012706 021562          MOV     #K5,#6      ;(050505)
10593 022142 112526          MOVVB   (5), (6)    ;LOW OF R5 TO LOW OF R6
10594 022144 022767 050456 177410  CMP     #050456,K5
10595 022152 001401          BEQ     BR13
(2) 022154 104000          EMT                ;FALSE R6 .BYTE TRANSFER
10596
10597 022156 012767 123456 177366 BR13:  MOV     #123456,K1
10598 022164 012767 050505 177370  MOV     #050505,K5
10599 022172 012705 021553          MOV     #K1+1,#5    ;123456
10600 022176 012706 021562          MOV     #K5,#6      ;050505
10601 022202 112526          MOVVB   (5), (6)    ;HIGH OF R5 TO LOW OF R6
10602 022204 026727 177352 050647  CMP     K5,#050647
10603 022212 001401          BEQ     BR14
(2) 022214 104000          EMT                ;FALSE R6 .BYTE TRANSFER
10604
10605 022216 012767 123456 177326 BR14:  MOV     #123456,K1
10606 022224 012767 050505 177330  MOV     #050505,K5
10607 022232 012705 021553          MOV     #K1+1,#5    ;R5-123456-ODD ADDRESS
10608 022236 012706 021562          MOV     #K5,#6      ;R6-050505-- .EVEN ADDRESS
10609 022242 112625          MOVVB   (6), (5)    ;LOW OF R6 TO HIGH OF R5
10610 022244 022767 042456 177300  CMP     #042456,K1
10611 022252 001401          BEQ     TS310
(3) 022254 104000          EMT                ;FAILED LOW OF 6 TO HIGH OF 5,OR WRONG #STN#
10612
(2)
(3)
(2) 022256
10613 022256 126767 177304 177303  CMPB   K7,K7+1      ;SAME .WORD LOW TO HIGH
10614 022264 001401          BEQ     BR15
(2) 022266 104000          EMT                ;SHOULD COMPARE LOW TO HIGH
10615
10616 022270 126767 177273 171270 BR15:  CMPB   K7+1,K7      ;COMPARE ODD TO .EVEN SAME .WORD
10617 022276 001401          BEQ     BR16
(2) 022300 104000          EMT                ;ODD TO .EVEN .BYTE FAILURE
10618
10619 022302 126767 177263 177256 BR16:  CMPB   K10+1,K7     ;SEQUENTIAL .BYTES
10620 022310 001401          BEQ     BR17
(2) 022312 104000          EMT                ;ODD TO .EVEN FAILED
10621
10622 022314 126767 177250 177242 BR17:  CMPB   K10,K6
10623 022322 001401          BEQ     BR20
(2) 022324 104000          EMT                ;.EVEN TO EVEN FAILED
10624 022326 126767 177235 177235 BR20:  CMPB   K7+1,K10+1
10625 022334 001401          BEQ     BR21
(2) 022336 104000          EMT                ;ODD TO ODD FAILED
10626
10627 022340 126767 177224 177223 BR21:  CMPB   K10,K10+1
10628 022346 001001          BNE     BR22
(2) 022350 104000          EMT                ;LOW TO HIGH IN SAME .WORD FAILED
10629

```

10630 022352 126767 177213 177211 BR22: CMPB K10.1,K10.1
 10631 022360 001401 BEQ BR23
 (2) 022362 104000 EMT ;HIGH TO LOW IN SAME .WORD FAILED
 10632
 10633 022364 126767 177200 177175 BR23: CMPB K10,K7.1
 10634 022372 001001 BNE TS311
 (3) 022374 104000 EMT ;.EVEN TO ODD FAILED,OR WRONG \$STNM
 10635
 10636
 10637
 (2)
 (3)
 (2) 022376
 10638 022376 012706 000150
 10639 022402 012767 022414 155374
 10640 022410 005746
 10641 022412 104000
 10642 022414
 10643
 10644
 (2)
 (3)
 (2) 022414
 10645 022414 012706 000150
 10646 022420 012767 022430 155356
 10647 022426 005746
 10648 022430 020627 000142
 10649 022434 001401
 (3) 022436 104000
 10650
 10651
 (2)
 (3)
 (2) 022440
 10652 022440 012706 000150
 10653 022444 005067 155476
 10654 022450 012767 022460 155326
 10655 022456 005246
 10656 022460 005767 155462
 10657 022464 001001
 (2) 022466 104000
 10658 022470 012705 001000
 10659 022474 012706 000400
 10660 022500 012767 022512 155276
 10661 022506 124645
 10662 022510 104000
 10663 022512 012706 000400
 10664 022516 012767 022530 155260
 10665 022524 134546
 10666 022526
 (2) 022526 104000
 10667 022530
 10677
 10678
 (3)
 (4)

```

;*****
;TEST 311 TEST THAT DECREMENT R6 TO A VALUE LESS THAN 400 TRAPS
;*****
TS311:
      MOV    #150,#6      ;R6 = 150
      MOV    @TDEC1,4     ;STACK OVERFLOW TRAP POINTER
      TST    -(6)         ;WITH R6 = 150 SHOULD TRAP
      EMT                    ;SHOULD HAVE TRAPPED,OR WRONG $STNM

TDEC1:

;*****
;TEST 312 TEST FOR DECREMENT OF R6 ON OVERFLOW TRAP
;*****
TS312:
      MOV    #150,#6      ;R6 = 150
      MOV    @TDEC2,4     ;TRAP POINTER
      TST    -(6)         ;WITH R6 = 150 SHOULD TRAP
      CMP    #6,@142     ;DID R6 DECREMENT
      BEQ    TS313
      EMT                    ;R6 NOT = 142,OR WRONG $STNM

;*****
;TEST 313 TEST DIFFERENT TYPES OF OVERFLOW
;*****
TS313:
      MOV    #150,#6
      CLR    146          ;STATUS WORD OF LOC 10
      MOV    @TDEC3,4     ;RETURN TO LOC 4
      INC    -(6)
      TST    146
      BNE    1#
      EMT                    ;INCREMENT OPERATION NOT INHIBITED

1#:
      MOV    #1000,#5
      MOV    #400,#6
      MOV    @TDEC4,4
      CMPB  -(6),-(5)
      EMT                    ;STACK = 400 AND DECREMENTED, SHOULD TRAP

TDEC4:
      MOV    #400,#6
      MOV    @TDEC7,4
      BITB  -(5),-(6)

TDEC6:
      EMT                    ;NO STACK OVERFLOW,OR WRONG $STNM

TDEC7:

;*****
;TEST 314 TEST THAT AN 77 CAUSES AN OVERFLOW TRAP
;*****

```

```

3) 022530
(1) 022530 012706 000400
(1) 022534 012767 022552 155246
(1) 022542 012767 022556 155234
(1) 022550 000077
(1) 022552 000167 157530
(1) 022556 012767 021530 155224
10679
(3)
(4)
(3) 022564
(1) 022564 012706 000400
(1) 022570 012767 022606 155222
(1) 022576 012767 022612 155200
(1) 022604 000004
(1) 022606 000167 157474
(1) 022612 012767 021534 155200
10680
10681
(2)
(3)
(2) 022620
10682 022620 012706 000400
10683 022624 012767 022642 155176
10684 022632 012767 022646 155144
10685 022640 104000
10686 022642 000167 157440
10687 022646 012767 002306 155154
10688
10689
(3)
(4)
(3) 022654
(1) 022654 012706 000400
(1) 022660 012767 022676 155146
(1) 022666 012767 022702 155110
(1) 022674 104400
(1) 022676 000167 157404
(1) 022702 012767 021540 155124
10690
(3)
(4)
(3) 022710
(1) 022710 012706 000400
(1) 022714 012767 022732 155072
(1) 022722 012767 022736 155054
(1) 022730 000003
(1) 022732 000167 157350
(1) 022736 012767 021532 155050
10691
(3)
(4)
(3) 022744
(1) 022744 012706 000400
(1) 022750 012767 022766 155026
(1) 022756 012767 022772 155020

```

TS314
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC2,10 ;SET UP 77 VECTOR
MOV #VDEC,4 ;SET UP OVERFLOW VECTOR
77 ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC2: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC: MOV #T010,10 ;RESTORE VECTOR
;.....
;TEST 315 TEST THAT AN IOT CAUSES AN OVERFLOW TRAP
;.....
TS315:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC4,20 ;SET UP IOT VECTOR
MOV #VDEC3,4 ;SET UP OVERFLOW VECTOR
IOT ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC4: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC3: MOV #T020,20 ;RESTORE VECTOR
;.....
;TEST 316 TEST THAT AN EMT CAUSES AN OVERFLOW TRAP (CHECK OF YELLOW ZONE)
;.....
TS316:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC6,30 ;SET UP INST VECTOR
MOV #VDEC5,4 ;SET UP OVERFLOW VECTOR
EMT ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC6: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC5: MOV #ERROR1,30 ;RESTORE VECTOR
;.....
;TEST 317 TEST THAT AN TRAP CAUSES AN OVERFLOW TRAP
;.....
TS317:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC8,34 ;SET UP TRAP VECTOR
MOV #VDEC7,4 ;SET UP OVERFLOW VECTOR
TRAP ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC8: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC7: MOV #T034,34 ;RESTORE VECTOR
;.....
;TEST 320 TEST THAT AN TRT CAUSES AN OVERFLOW TRAP
;.....
TS320:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC10,14 ;SET UP TRT VECTOR
MOV #VDEC9,4 ;SET UP OVERFLOW VECTOR
TRT ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC10: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC9: MOV #T014,14 ;RESTORE VECTOR
;.....
;TEST 321 TEST THAT AN ILLA CAUSES AN OVERFLOW TRAP
;.....
TS321:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC11,4 ;SET UP ILLA VECTOR
MOV #VDEC12,4 ;SET UP OVERFLOW VECTOR

E10

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58 P11 07 JAN 85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 10-22
TS2: TEST THAT AN ILLA CAUSES AN OVERFLOW TRAP

SEQ 0121

(1) 022764 004700
(1) 022766 000167 157314
(1) 022772 012767 021526
10692 023000 020627 000370
10693 023004 001401
(3) 023006 104000

ILLA ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC11: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
155004 VDEC12: MOV #T04,4 ;RESTORE VECTOR
CMP #6,#370 ;STACK PUSHED FOUR WORDS?
BEQ TS322
EMT ;TRAP OVERFLOW DID NOT OCCUR

10694
(3)
(4)

;*****
;TEST 322 TEST THAT AN ILLB CAUSES AN OVERFLOW TRAP
;*****

(3) 023010
(1) 023010 012706 000400
(1) 023014 012767 023032 154762
(1) 023022 012767 023036 154754
(1) 023030 000100
(1) 023032 000167 157250
(1) 023036 012767 021526 154740

TS322:
MOV #400,#6 ;SET UP STACK TO OVERFLOW
MOV #VDEC13,4 ;SET UP ILLB VECTOR
MOV #VDEC14,4 ;SET UP OVERFLOW VECTOR
ILLB ;THIS TRAP SHOULD CAUSE OVERFLOW
VDEC13: JMP ERROR1 ;USE JUMP TO GET TO ERROR BECAUSE UNSURE WHAT EMT WILL D
VDEC14: MOV #T04,4 ;RESTORE VECTOR

10695
10696
(2)
(3)
(2) 023044

;*****
;TEST 323 TEST FOR FALSE OVERFLOW TRAP
;*****
TS323:

10697
10698 023044 012767 023112 154732
10699 023052 012706 001002
10700 023056 005746
10701 023060 012706 002002
10702 023064 005746
10703 023066 012706 004002
10704 023072 005746
10705 023074 012706 010002
10706 023100 005746
10707 023102 012706 020000
10708 023106 005746
10709 023110 000401

MOV #FOVER,4 ;SET UP OVERFLOW POINTER
MOV #1002,#6
TST -(6) ;SHOULD NOT OVERFLOW
MOV #2002,#6
TST -(6) ;SHOULD NOT OVERFLOW
MOV #4002,#6
TST -(6) ;SHOULD NOT OVERFLOW
MOV #10002,#6
TST -(6)
MOV #20000,#6 ;SHOULD NOT OVERFLOW
TST -(6)
BR STP

(1) 023112
(2) 023112 104000
10710 023114 012767 021526 154662
10711 023122 005067 154660

FOVER:
EMT ;IT OVERFLOWED,OR WRONG \$TSTNM
STP: MOV #T04,4
CLR 6

10712
(2)
(3)
(2) 023126

;*****
;TEST 324 TEST THAT BIT 4 PSW WILL CAUSE A TRAP TO 14
;*****
TS324:

10713 023126 012706 001000
10714 023132 012767 023156 154654
10715 023140 012746 000020
10716 023144 012746 023152
10717 023150 000002
10718 023152 000240
10719 023154 104000
10720 023156

MOV #STBOT,SP ;SET UP TO TRAP TO 14
MOV #RETAT,RTRAP4 ;PUSH T BIT
MOV #20,-(SP) ;PUSH PC
MOV #.+6,-(SP) ;SET T BIT
RTI ;TRAP HERE
NOP ;TRACE BIT DID NOT TRAP!,OR WRONG \$TESTN
EMT

10721
(2)
(3)
(2) 023156

RETAT:
;*****
;TEST 325 TEST STACK POINTER DECREMENTS
;*****
TS325:

10722 023156 012706 001000
10723 023162 012767 023206 154624

MOV #STBOT,SP
MOV #RETBT,RTRAP4

```

10724 023170 012746 000020          MOV    #20,(SP)      ;PUSH T BIT
10725 023174 012746 023202          MOV    #.6,(SP)    ;PUSH PC
10726 023200 000002                    RTI                    ;SET T BIT
10727 023202 000240                    NOP                    ;TRAP HERE
10728 023204 104000                    EMT                    ;TRACE BIT DID NOT TRAP!
10729 023206 020627 000774          RETBT: CMP    SP,#STBOT 4
10730 023212 001401                    BEQ    TS326
      (3) 023214 104000                    EMT                    ;STACK POINTER WAS NOT PUSHED BY TRAP,OR WRONG #TESTN
10731                                     ;*****
      (2)                                     ;TEST 326      TEST FOR PROPER PC ON STACK
      (3)                                     ;*****
      (2) 023216                                     TS326:
10732 023216 012706 001000          MOV    #STBOT,SP
10733 023222 012767 023242 154564          MOV    #RETCT,RTRAP4
10734 023230 012746 000020          MOV    #20,-(SP)    ;PUSH T BIT
10735 023234 012746 023242          MOV    #.6,-(SP)    ;PUSH PC
10736 023240 000002                    RTI                    ;SET T BIT
10737                                     ;TRAP HERE
10738 023242 022767 023242 155524          RETCT: CMP    #. ,STBOT-4
10739 023250 001401                    BEQ    TS327
      (3) 023252 104000                    EMT                    ;CORRECT PC WAS NOT SAVED ON STACK,OR WRONG #TESTN
10740                                     ;*****
10741                                     ;TEST 327      TEST THAT RTT POPS T. BIT
10742                                     ;*****
      (2) 023254                                     TS327:
10743                                     ;*****
10744 023254 012706 001000          MOV    #STBOT,SP
10745 023260 005001                    CLR    R1              ;CLEAR R1
10746 023262 012746 000020          MOV    #20,-(SP)
10747 023266 012746 023302          MOV    #RTT1,-(SP)
10748 023272 012767 023310 154514          MOV    #RTT2,14
10749 023300 000006                    RTT
10750 023302 000240          RTT1: NOP
10751 023304 001401                    BEQ    TS330
      (3) 023306 104000                    EMT                    ;T-BIT DID NOT TRAP,OR WRONG #TESTN
10752                                     ;*****
10753 023310                                     RTT2:
10754                                     ;*****
      (2)                                     ;TEST 330      TEST THAT RTT ALLOWS ONE INST. BEFORE TRAP
      (3)                                     ;*****
      (2) 023310                                     TS330:
10755 023310 012705 177777          RTT5: MOV    #177777,#5
10756 023314 012706 001000          MOV    #STBOT,SP
10757 023320 012746 000020          MOV    #20,-(SP)
10758 023324 012746 023342          MOV    #RTT3,-(SP)
10759 023330 012767 023352 154456          MOV    #RTT4,14
10760 023336 005001                    CLR    R1              ;CLEAR R0
10761 023340 000006                    RTT                    ;SET T-BIT
10762 023342 005201          RTT3: INC    R1
10763 023344 005205                    INC    #5
10764 023346 001762                    BEQ    RTT5            ;DO THIS TEST NO MORE THAN 2 TIMES
10765 023350 104000                    EMT                    ;DID NOT TRAP
10766 023352 005301          RTT4: DEC    R1
10767 023354 001403                    BEQ    RTT6            ;SEE IF RTT ALLOWS 1 INST.

```

10768 023356 005205
 10769 023360 001755
 (2) 023362 104000
 10770 023364
 10771
 (2)
 (3)
 (2) 023364
 10772 023364 012706 001000
 10773 023370 012746 000020
 10774 023374 012746 023412
 10775 023400 012767 023416 154406
 10776 023406 005001
 10777 023410 000002
 10778 023412 005201
 10779 023414 104000
 10780 023416 005701
 10781
 10782 023420 001401
 (3) 023422 104000
 10783
 10784
 (2)
 (3)
 (2) 023424
 10785
 10786 023424 012706 001000
 10787 023430 012767 023470 154356
 10788 023436 005027 000016
 10789 023442 005027 000022
 10790 023446 012767 023474 154344
 10791 023454 012746 000020
 10792 023460 012746 023466
 10793 023464 000006
 10794 023466 000004
 10795 023470
 (2) 023470 104000
 10796 023472
 (2) 023472 104000
 10797 023474 012767 000016 154312
 10798 023502 012767 000022 154310

```

      INC      #5           ;DO THIS TEST NO MORE THAN TWO TIMES
      BEQ      RTT5
      EMT
      ;RTT DID NOT ALLOW 1 INST.,OR WRONG #TESTN

RTT6:
;*****
;TEST 331      TEST THAT RTI DOES NOT ALLOW 1 INST.
;*****
TS331:
      MOV      #STBOT,SP
      MOV      #20,(SP)
      MOV      #RTI1,-(SP)
      MOV      #RTI2,14
      CLR      R1
      RTI
      ;SET T-BIT
      INC      R1           ;RTI SHOULD NOT ALLOW THIS
      EIT
      TST      R1           ;T- BIT DID NOT CAUSE TRAP
      ;RTI SHOULD NOT ALLOW 1 INST. BEFORE TRAP
      BEQ      TS332
      EMT
      ;RTI DID ALLOW 1 INST. BEFORE TRAP,OR WRONG #TESTN

;*****
;TEST 332      TEST TRAP ON TRAP THAT TRACE BIT TRAPS ARE INHIBITED ON TRAP INST
;*****
TS332:
      MOV      #STBOT,#6
      MOV      #TRACE,14   ;TRACE TRAP
      CLR      #16
      CLR      #22
      MOV      #TONT1,20   ;IOT TRAP
      MOV      #20,-(SP)   ;PUSH T BIT
      MOV      #.6,-(SP)  ;PUSH PC
      RTT
      IOT
      ;TRAP, NEW CC HAVE TRACE RESET
      EMT
      ;TRACE TRAP WAS NOT INHIBITED
BR70:
      EMT
      ;WRONG TSTNM,OR WRONG #TSTNM
TONT1:
      MOV      #16,14
      MOV      #22,20

```

```

10800
  (2)
  (3)
  (2) 023510
10801 023510 012706 001000
10802 023514 012767 023540 154272
10803 023522 005067 154270
10804 023526 012746 000020
10805 023532 012746 023540
10806 023536 000002
10807 023540 036727 155232 000020 TRC1:
10808 023546 001001
  (1) 023550
  (2) 023550 104000
10809 023552 012767 021532 154234 STP3D:
10810

```

```

;*****
;TEST 333 TEST THAT THE TRACE BIT IS SAVED IN THE STACK
;*****
TS333:
      MOV     @STBOT,#6      ;SET UP STACK POINTER
      MOV     @TRC1,14      ;TRACE TRAP RETURN
      CLR     16
      MOV     @20,-(SP)     ;SET THE T BIT
      MOV     @TRC1,-(SP)
      RTI
      BIT     STBOT-2,@20   ;CHECK FOR T BIT ON STACK
      BNE     STP3D
      STP3:
      EMT
      MOV     @T014,14     ;T BIT NOT SAVED ON THE STACK,OR WRONG #TSTNM

```

```

10812
10813
10814
10815
10816
10817
10818
10819
10820
10821
10822
  (2)
  (3)
  (2) 023560
10823 023560 005000
10824 023562 005067 154220
10825 023566 012767 023652 154210
10826 023574 012706 001000
10827 023600 105720
10828 023602 020027
10829 023604 160000
10830 023606 103774
10831 023610 012737 023624 000004
10832 023616 105737 177700
10833 023622
  (2) 023622 104000
10834
10835 023624 106767 154146
10836 023630 005767 154142
10837 023634 001401
  (2) 023636 104000
10838 023640 026727 155130 023622
10839 023646 001437
  (2) 023650 104000
10840
10841 023652 005300
10842 023654 010067 000032
10843
10844 023660 013700 023604
10845 023664 005300
10846 023666 000402
10847 023670 162700 001000
10848
10849 023674 012767 023720 154102
10850 023702 012706 001000
10851 023706 005710
10852
10853 023710 020027
10854
10855 023712 000000
10856 023714 101414
  (2) 023716 104000
10857
10858
10859
10860 023720 106767 154052

```

```

;*****
;THIS ROUTINE TESTS THAT NO LEGAL ADDRESS TRAPS AND THAT AN ILLEGAL
;ADDRESS TRAPS TO LOCATION 4. THIS WILL RUN ON 30K SYSTEM. BUT IF
;SWITCH REGISTER BIT 1=0, THEN THE MEMORY FROM 28K-30K IS NOT LOOKED
;AT, SINCE IT MAY HAVE I/O DEVICES. IF SWR BIT 1=1, THEN THAT AREA IS
;CHECKED. (IT SHOULD EITHER ALL TRAP OR ALL NOT TRAP). LOC 160000
;IS NO LONGER GUARANTEED TO TRAP, SINCE IT MAY CONTAIN MEMORY. LOCATION
;177700 (THE UNIBUS ADDRESS FOR RO ON OLDER SYSTEMS) IS USED FOR FORCING
;A TIMEOUT IN THE EVENT THAT THERE WAS NO TIMEOUT FROM 0K-28K OR 30K.
;THIS ROUTINE TESTS MEMORY UNTIL IT DOES A NXM STOP
;*****
;TEST 334 TEST NON-EXISTENT ADDRESS TRAPS
;*****
TS334:
14: CLR R0 ;
CLR 6 ;
MOV #ATRAP,4 ;SET UP ADDRESS TRAP ENTRANCE
MOV #STBOT,SP ;SET STACK POINTER
NOR: TSTB (0)+ ;IF OUTSIDE OF CORE, TRAP TO 4
CMP R0,(PC)+ ;IS POINTER INSIDE 28K (30K) CORE
HICORE: .WORD 160000 ;MAY BE CHANGED TO 170000 IF 30K
BLO NOR ;TEST THE REST OF CORE
MOV #ROTRAP,#4 ;SET UP NEW VECTOR POINTER
TSTB #177700 ;SHOULD CAUSE A TRAP

TRPADR:
EMT ;SHOULD HAVE TRAPED
;TRAP TO HERE IF FORCING TRAP BY TESTING 177700
ROTRAP: MFPS STATUS
TST STATUS ;TEST PSW
BEQ 14 ;NEW PSW SHOULD HAVE BEEN ZERO
EMT ;TEST OLD PC AT STACK
14: CMP STBOT-4,#TRPADR ;OLD PC WAS NOT SAVED
BEQ TRAPB ;RETURN HERE ON AN ADDRESS TRAP FROM MEMORY BELOW 28K (OR 30K)
EMT
ATRAP: DEC R0 ;
MOV R0,CORH ;MOVE THE FIRST NXM LOCATION IN CORH
;THIS ROUTINE DOES NXM TRAPS UNTIL IT FINDS AN EXISTENT MEMORY LOCATION
MOV #HICORE,R0 ;SET UP THE HIGHEST MEM LOCATION
DEC R0 ;MAKE 1 LESS THAN THE HIGHEST CORE BOUNDARY
BR NOSUB ;DON'T SUBTRACT 1K FIRST TIME
CTRAP: SUB #1000,R0 ;SUBTRACT 1K OCTAL BYTE FROM ADDRESS
;TO SPEED UP TESTING
NOSUB: MOV #BTRAP,4 ;SET UP THE VECTOR
MOV #STBOT,SP
TST (R0) ;DOES THIS MEMORY EXIST?
;IF NXM, TRAP TO BTRAP
DTRAP1: CMP R0,(PC)+ ;IF EXISTS, IS THIS THE SAME TRAP THAT CAUSED
;TRAP TO ATRAP

CORH: .WORD 0
BLOS TRAPB ;CONTENTS OF R0 SHOULD BE LESS THAN OR EQUAL TO CORH
EMT ;IF THIS COMPARISON FAILS IT MEANS
;THAT SOME LEGAL ADDRESS TRAPPED, OR
;THAT AN ILLEGAL ADDRESS DID NOT TRAP

BTRAP: MFPS STATUS

```



```
10861 023724 005767 154046 TST STATUS
10862 023730 001401 BEQ 1#
(2) 023732 104000 EMT ;NEW PSW SHOULD HAVE BEEN ZERO
10863 023734 026727 155034 023710 1# : CMP STBOT-4,#DTRAP1 ;CHECK IF TRAP PC IS OK
10864 023742 001752 BEQ CTRAP
(1) 023744 AUTO1:
(2) 023744 104000 EMT ;OLD PC WAS NOT SAVED OR WRONG #TESTM
10865 023746 012767 021526 154030 TRAPB: MOV #T04,4 ;RESET TRAP CATCHER
10866 023754 005067 154026 CLR 6 ;RESET TRAP CATCHER
10867
10868
10869 ;THIS ROUTINE WILL FIGURE OUT IF YOU HAVE A DL11W
10870 023760 012706 001000 MOV #STBOT,SP ;SET UP THE STACK POINTER
10871 023764 012767 024000 154012 MOV #NODL,4 ;SET UP THE TRAP VECTOR
10872 023772 005767 153566 TST TTCSR ;TEST THE PUNCH STATUS REGISTER
10873 023776 000405 BR DL11W
10874 024000 012767 021526 153776 NODL: MOV #T04,4
10875 024006 000167 100624 JMP SLU1ST ;IF NO SLU FIND OUT WHY IN SLU TEST
10876 024012 012767 021526 153764 DL11W: MOV #T04,4
10877
10878 ;*****
(2) ;TEST 335 TEST THAT A TTY INTERRUPT CAUSES AN OVERFLOW TRAP
(3) ;*****
(2) 024020 TS335:
10879 024020 012767 000340 153750 MOV #340,STATUS ;LOCK OUT INTERRUPT
10880 024026 012706 000400 MOV #400,#6 ;SET UP STACK TO OVERFLOW
10881 024032 012767 024074 153744 MOV #TDEC77,4 ;SET UP OVERFLOW TRAP
10882 024040 016767 154020 001562 MOV 64,TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
10883 024046 012767 024072 154010 MOV #TDEC8,64 ;SET UP INTERRUPT VECTOR
10884 024054 012767 000100 153502 MOV #100,TTCSR ;SET INTERRUPT ENABLE
10885 024062 005067 153710 CLR STATUS ;ALLOW INTERRUPT TO OCCUR
10886 024066 000167 100544 JMP SLU1ST ;NO INTERRUPT OCCURRED SO GO TO SLU TEST
10887 ;TO FIND OUT WHY ADD REPORT PROPER ERROR
10888 024072 TDEC8:
(2) 024072 104000 EMT ;OVERFLOW TRAP DID NOT OCCUR
10889 024074 005067 153464 TDEC77: CLR TTCSR ;CLEAR INTERRUPT ENABLE
10890 024100 012767 021526 153676 MOV #T04,4
10891 024106 005067 153674 CLR 6
10892 024112 016767 001512 153744 MOV TEMP1,64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
10893 ;*****
(2) ;TEST 336 TEST THAT A PENDING INTERRUPT OCCURS BEFORE TRAP
(3) ;*****
(2) 024120 TS336:
10894 024120 012706 001000 MOV #STBOT,#6
10895 024124 012767 000340 153644 MOV #340,STATUS ;SET TO A HIGH PRIORITY LEVEL
10896 024132 016767 153726 001470 MOV 64,TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
10897 024140 012767 024204 153716 MOV #TRO,64
10898 024146 012767 000100 153410 MOV #100,TTCSR ;INTERRUPT FOR TTY PUNCH/PRINTER
10899 024154 012767 024206 153652 MOV #BR71,34 ;TRAP VECTOR
10900 024162 012767 024210 153674 MOV #TR2,64 ;TTY VECTOR
10901 024170 012767 000340 153640 MOV #340,36 ;IF TRAP TRAPS, MOVE 340 TO PRIORITY
10902 024176 005067 153574 CLR STATUS ;SHOULD INTERRUPT AT END OF CLR INST
10903 024202 104400 TRAP ;TTY INTERRUPT SHOULD OVERRIDE TRAP
10904 024204 TRO:
(2) 024204 104000 EMT ;TTY SHOULDN'T HAVE INTERRUPTED
10905 024206 BR71:
```

```

(2) 024206 104000
10906 024210 005067 153622
10907 024214 016767 001410 153642
10908 024222 042767 000100 153334
10909
(2)
(3)
(2) 024230
10910 024230 012706 001000
10911 024234 012767 000340 153534
10912 024242 012767 000100 153314
10913 024250 012767 024316 153556
10914 024256 016767 153602 001344
10915 024264 012767 024322 153572
10916 024272 012767 000340 153566
10917 024300 012767 024320 153512
10918 024306 012767 000340 153506
10919 024314 104400
10920 024316 000004
10921 024320
(2) 0 320 104000
10922 02 22 005067 153474
10923 024 26 005067 153534
10924 024332 012767 021540 153474
10925 024340 016767 001264 153516
10926 024346 012767 000022 153444
10927 024354 042767 000100 153202
10928
10929
(2)
(3)
(2) 024362
10930 024362 032737 000001 001020
(1) 024370 001403
(1) 024372 005737 001006
(2) 024376 001013
(1) 024400
10931 024400 016700 153156
10932 024404 012767 000100 153146
10933 024412 000005
10934 024414 032767 000100 153136
10935 024422 001401
(3) 024424 104000
10936 024426
10937
(2)
(3)
(2) 024426
10938 024426 032737 000001 001020
(1) 024434 001403
(1) 024436 005737 001006
(2) 024442 001024
(1) 024444
10939 024444 012706 001000
10940 024450 012767 024476 153336
10941 024456 012746 000020

```

```

EMT ;TRAP OCCURRED FIRST
TR2: CLR 36
MOV TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
BIC #100,TTCSR
;*****
;TEST 337 TEST THAT A PENDING INTERRUPT, INTERRUPTS BETWEEN TRAPS
;*****
TS337:
MOV #STBOT,#6
MOV #340,STATUS
MOV #100,TTCSR
MOV #TR3,34 ;TRAP
MOV 64, TEMP1 ;SAVE CONTENTS OF INTERRUPT VECTOR
MOV #TR4,64 ;TTY OUTPUT
MOV #340,66 ;TTY OUTPUT PRIORITY
MOV #TR5,20 ;IOT
MOV #340,22 ;IOT PRIORITY
TRAP ;THE ACT OF TRAPPING LOWER PRIORITY
TR3: IOT ;INTERRUPT SHOULD OCCUR IN PLACE OF IOT TRAP
TR5:
EMT ;NO INTERRUPT BETWEEN TRAPS,OR WRONG $TSTNM
TR4: CLR 22 ;CLR IOT PRIORITY
CLR 66
MOV #T034,34
MOV TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
MOV #22,20
BIC #100,TTCSR ;CLEAR IE BIT IN SLU1 XMIT CSR
;*****
;TEST 340 TEST THAT "RESET" GOES TO OUTSIDE WORLD
;*****
TS340:
BIT #1, #ENV ;ARE WE RUNNING UNDER APT
BEQ 704 ;IF NO THEN DO TEST
TST #PASS ;IS THIS FIRST PASS
BNE TS341 ;IF NO THEN SHIP TO NEXT TEST
704:
MOV TKB,RO ;MAKE SURE RECEIVER DONE IS CLEAR
MOV #100,TRCSR ;SET INTERRUPT ENABLE
RESET ;SHOULD CLEAR INTERRUPT ENABLE
BIT #100,TRCSR ;TEST FOR CLEAR
BEQ TS341
EMT ;RESET FAILED TO CLEAR TRCSR,OR WRONG $TSTNM
NODL2:
;*****
;TEST 341 TEST THAT RESET HAS NO EFFECT ON THE TRACE TRAP
;*****
TS341:
BIT #1, #ENV ;ARE WE RUNNING UNDER APT
BEQ 704 ;IF NO THEN DO TEST
TST #PASS ;IS THIS FIRST PASS
BNE TS342 ;IF NO THEN SHIP TO NEXT TEST
704:
MOV #STBOT,#6 ;SET STACK
MOV #RESET2,14 ;SET UP TRACE VECTOR
MOV #20,-(R6) ;SET THE T-BIT ON STACK

```

10942 024462 012746 024470
10943 024466 000006
10944 024470 000005
10945 024472 000005
10946 024474
(2) 024474 104000
10947 024476 005067 153274
10948 024502 005067 153310
10949 024506 012767 021532 153300
10950 024514
10951
10952
(2)
(3)
(2) 024514

MOV #16, (R6) ;MOVE NEW PC ON STACK
RTT
14: RESET ;SHOULD HAVE NO EFFECT
RESET ;NO EFFECT
RESET3: EMT ;TRACE TRAP FAILED,OR WRONG #TSTNM
RESET2: CLR STATUS ;CLEAR TRACK
CLR 16 ;TRACE STATUS
MOV #T014,14
SKTST2:

;TEST 342 TEST THE 'WAIT' INSTRUCTION

TS342:

10953 024514 032737 000040 000052
10954 024522 001402
10955 024524 000167 000154
10956 024530 122767 000001 154262 54:
10957 024536 001003
10958 024540 005767 154242
10959 024544 001051
10960 024546
10961 024546 042767 000100 153010 14:
10962 024554 012706 001000
10963 024560 016767 153300 001042
10964 024566 012767 024646 153270
10965 024574 005067 153266
10966 024600 105767 152760 WATE1:
10967 024604 100375 BPL WATE1 ;TO BE UP
10968 024606 012767 000015 152752 WATE2:
10969 024614 105767 152744 TSTB TTCSR ;WAIT FOR READY
10970 024620 100375 BPL WATE2 ;DO ANOTHER CARRIAGE RETURN
10971 024622 012767 000015 152736 MOV #15,TPB ;DO ANOTHER CARRIAGE RETURN
10972 024630 052767 000100 152726 BIS #100,TTCSR ;SET THE INTERRUPT ENABLE
10973 024636 005067 153134 CLR STATUS ;CLEAR THE PSW
10974 024642 000001 WATE3: WAIT ;WAIT FOR THE INTERRUPT
10975 024644 104000 EMT ;WAIT INSTRUCTION DID NOT LOOP
10976 024646 005767 153124 WATE: TST STATUS ;IS THE PSW CORRECT?
10977 024652 001401 BEQ 14 ;NEW PSW SHOULD HAVE BEEN ZERO
(2) 024654 104000 EMT ;IS THE OLD PC SAVED
10978 024656 026727 154112 024644 14: CMP STBOT-4, #WATE3+2
10979 024664 001401 BEQ STP4E
(1) 024666 STP4:
(2) 024666 104000 EMT ;OLD PC WAS NOT SAVED OR WRONG #TESTN
10980 024670 016767 000734 153166 STP4E: MOV TEMP1, 64 ;RESTORE CONTENTS OF INTERRUPT VECTOR
10981 024676 042767 000100 152660 BIC #100,TTCSR ;CLEAR IE BIT IN SLU1 XMIT CSR

;TEST 343 TEST THAT USING REGISTER ADDR (177700) CAUSES TIME OUT.

TS343:

10982
(2)
(3)
(2) 024704
10983
10984
10985
10986
10987 024704 012706 001000

;
;REGISTER ADDRESS (177700-177717) CAUSE TIME OUT WHEN USED
;AS PROGRAM ADDRESS BY THE CPU.
;
NXTST: MOV #STBOT,SP ;SET STACK POINTER

10988 024710 012767 024724 153066
10989 024716 005237 177700
10990 024722 104000
10991 024724 022767 024722 154042
10992 024732 001401
(3) 024734 104000
10993
10994
10995
10996
10997
(2)
(3)
(2) 024736
10998
10999 024736 012737 024762 000004
11000 024744 005037 000000
11001 024750 005337 000001
11002 024754 022737 177777 000000
11003 024762
(2) 024762 001401
(3) 024764 104000
11004
11005
11006
11007
11008
11009
11010
(2)
(3)
(2) 024766
11011 024766 012737 025006 000004
11012 024774 012700 177700
11013 025000 012720 001234
11014 025004 104000
11015 025006 022700 177702
11016 025012 001401
(3) 025014 104000
11017
11018
11019
11020
11021
11022
11023
11024
11025
11026
(2)
(3)
(2) 025016
11027 025016 012767 025060 152760
11028 025024 012737 000340 000006
11029 025032 012767 025056 152750
11030 025040 012737 000340 000012

PCN1: MOV #RETR1,RTRAP5 ;SET TRAP RETURN ADDR
INC #0177700 ;BAD ADDR REFERENCE, TRAP TO 4
EMT ;REFERENCING 177700 DID NOT CAUSE TIME OUT
RETR1: CMP #PCN1+4,STBOT-4 ;PROPER PC STORED ON STACK?
BEQ TS344
EMT ;OLD PC WAS NOT SAVED IN STACK
;*****
;ODD ADDRESS USED BY A "WORD" INSTRUCTION SHOULD NOT
;CAUSE A TRAP, BUT THE LOW ORDER ADDRESS BIT WOULD BE IGNORED.
;*****
;TEST 344 TEST ODD ADDRESS TRAP IS NOT IMPLEMENTED.
;*****
TS344:
;MOV #RETR2,#RTRAP5 ;SET TRAP RETURN ADDR
CLR #0 ;PUT ALL 0 IN LOC 0
DEC #1 ;DECREMENT ODD ADDRESS, SHOULD NOT TRAP
CMP #-1,#00 ;WORD LOC 0 HAS ALL ONES?
RETR2: BEQ TS345
EMT ;LOC 0 DID NOT STORE -1,OR ODD ADDR REFERENCE CAUSE TRAP
;*****
;USING ADDRESS 177700 IN MODE 2, CAUSES BUS ERROR, BUT
;THE REGISTER IN USE WILL BE INCREMENTED.
;*****
;TEST 345 TEST THAT IN MODE 2, BAD ADDRESS REFERENCE CAUSES BUS ERROR.
;*****
TS345:
MOV #RETR3,#RTRAP5 ;SET TRAP RETURN ADDR
MOV #177700,R0 ;STORES BAD MEMORY REFERENCE
MOV #1234,(R0) ;BAD ADDR REFERENCE, TRAP TO LOC 4
EMT ;ADDRESSING 177700 DID NOT CAUSE TRAP
RETR3: CMP #177702,R0 ;WAS R0 INCREMENTED?
BEQ TS346
EMT ;R0 WAS NOT INCREMENTED
;*****
;AFTER THE FIRST BUS ERROR WAS ENCOUNTERED, AN ATTEMPT WAS MADE
;TO PUSH PC AND PS INTO THE STACK. HOWEVER, IF THE STACK POINTER
;WAS BAD, A DOUBLE BUS ERROR OCCURED. THE STACK POINTER WOULD
;THEN BE SET TO LOCATION 4, OLD PC AND PS WERE PUSHED INTO
;LOCATIONS 0 AND 2. THE PROCESSOR WOULD TRAP TO 4 AND CONTINUE
;EXECUTION.
;*****
;TEST 346 TEST FOR DOUBLE BUS ERROR.
;*****
TS346:
MOV #DBE1,RTRAP5 ;SET TRAP RETURN ADDR
MOV #340,#6 ;SET UP PS
MOV #DBE2,RTRAP ;SET TRAP RETURN ADDR
MOV #340,#12 ;SET UP PS

11

11

11031	025046	012706	177700			MOV	#177700,SP	;SET ILLEGAL SP	116
11032	025052	000077				DBE:	TRAPA	;ILLEGAL INSTRUCTION	116
11033	025054	104000				EMT		;DOUBLE BUS ERROR DID NOT CAUSE TRAP	116
11034	025056					DBE2:			116
(2)	025056	104000				EMT		;TRAP TO WRONG LOCATION	116
11035	025060	022737	025054	000000	DBE1:	CMF	#DBE+2,0#0	;OLD PC GOT SAVED?	116
11036	025066	001401				BEQ	DBE3		116
(2)	025070	104000				EMT		;OLD PC DID NOT GET SAVEDD	116
11037	025072	022737	000340	000002	DBE3:	CMF	#340,0#2	;CORRECT PS SAVED?	116
11038	025100	001401				BEQ	DBE4		116
(2)	025102	104000				EMT		;CORRECT PS DID NOT GET SAVE	116
11039	025104	022706	000000		DBE4:	CMF	#0,SP	;SP POINTS TO LOC 0?	116
11040	025110	001401				BEQ	DBE5		116
(2)	025112	104000				EMT		;SP IS NOT POINTING TO LOC 0	116
11041	025114	012706	001000		DBE5:	MOV	#STBOT,SP	;RESET SP	116
11042	025120	012767	021526	152656		MOV	#T04,4	;RESET VECTOR 4	116
11043	025126	012767	021530	152654		MOV	#T010,10	;RESET VECTOR 10	116
11044									116
11045									116
11046									116
11047									116
11048									116
11049									116
11050									116
11051									116
11052									116
(2)									116
(3)									116
(2)	025134								116
11053									116
11054	025134	012706	001000			MOV	#STBOT,R6	;INIT STACK POINTER	116
11055	025140	012737	025164	000010		MOV	#1\$,0#10	;SET UP RETURN ADDR FROM TRAP	116
11056	025146	012737	000340	000012		MOV	#340,0#12	;SET TRAP PRIORITY=7	116
11057	025154	075006				FADD	R6	;EXECUTE FIS INSTR..SHOULD CAUSE CTLERR	116
11058	025156	004767	106246			JSR	PC,ABORT	;SEE IF WE ARE UNDER UFD	116
11059	025162	000000				HALT		;DID NOT TRAP..CHECK CSEL LINE	116
11060	025164	012706	001000		1\$:	MOV	#STBOT,R6	;RE-INIT STACK POINTER	116
11061	025170	012767	021530	152612		MOV	#T010,10	;RESET VECTOR 10	116
11062									116
11063									116
(2)									116
(3)									116
(2)	025176								116
11064	025176	042767	000100	152360		BIC	#100,TTCSR		116
11065								; SET UP TO SEE IF	116
11066	025204	013767	000010	000042		MOV	#010,TENSAVE	; THIS PROCESSOR HAS THE	116
11067	025212	012737	025256	000010		MOV	#TRAP10,0#10	; FLOATING POINT OPTION	116
11068	025220	170127	000000			LDFPS	#0	;DO A FPP INSTRUCTION	116
11069								;IF NO TRAP FPP INSTALLED	116
11070	025224	013767	025600	000356		MOV	#FPP,FINISH	;SO RESET END OF TABLE POINTER	116
11071	025232	000411				BR	AROUND	; THE FOLLOWING	116
11072									116
11073									116
11074	025234	042777	000040	174334		* IF NO CIS OPTION TRAP TO HERE			116
11075	025242	012716	025310		CISTRP:	BIC	#40,0SWR	;CLEAR CIS OPTION IN SWR	116
11076	025246	000002				MOV	#CONCIS,(SP)	;CHANGE RETURN ADDRESS TO CONCIS LOCATION	116
						RTI		;RETURN	116

```

;*****
;THIS TEST WILL CHECK THE SERVICE ROUTINE FOR A CONTROL CHIP ERROR.
;THIS IS DONE BY EXECUTING INSTRUCTIONS WHICH JUMP TO NON-EXISTENT
;CONTROL-CHIP. THE TEST EXECUTES AN FIS INSTRUCTION WHICH
;IS ILLEGAL ON ALL PROCESSORS USING THE DCF11-A CHIP SET.
;A CTLERR TRAPS TO LOCATION 10.
;THE RESET LINE IS ALSO ASSERTED FOR 1 CYCLE.
;*****
;TEST 347 TEST CTLERR SERVICE ROUTINE
;*****
TS347:

```

```

;*****
;TEST 350 TEST THAT ALL RESERVED INSTRUCTIONS TRAP
;*****
TS350:

```

11077	025250	000000				CISADR: WORD 0		; DATA FOR CIS
11078	025252	000000				WORD 0		; INSTRUCTION
11079	025254	000000				TENSAVE: WORD 0		; A PLACE TO STORE CONTENTS OF 10
11080	025256					TRAP10:		; LEAVE THE TABLE ALONE
11081								
11082	025256					AROUND:		; CONTINUATION POINT
11083	025256	012737	000246	000244		MOV #246,#244		; RESTORE THE TRAP VECTOR
11084	025264	012737	025234	000010		MOV #CISTRP,#10		; SET UP TO SEE IF THIS HAS THE CIS OPTION
11085	025272	076144				WORD 076144		; EXECUTE A CMPCI INSTRUCTION
11086	025274	025250				WORD CISADR		; OPERANDS
11087	025276	025250				WORD CISADR		; FOR CIS
11088	025300	000000				WORD 0		; INSTRUCTION
11089	025302	052777	000040	174266		BIS #40,BSWR		; SET CIS PRESENT BIT
11090	025310	016737	177740	000010		CONCIS: MOV TENSAVE,#10		; RESTORE THE ILLEGAL INST. VECTOR
11091	025316	012703	025470			MOV #TABLE,TAB		; TABLE POINTER
11092	025322	012305				GIN1: MOV (TAB)+,FIRST		; FIRST OR CURRENT INSTRUCTION
11093	025324	012301				MOV (TAB)+,LAST		; LAST INSTRUCTION OR GROUP
11094	025326	020537	025544			CMP FIRST,#CIS		
11095	025332	001007				BNE 18		
11096	025334	032777	000040	174234		BIT #40,BSWR		
11097	025342	001403				BEQ 18		
11098	025344	012703	025600			MOV #FPP,TAB		
11099	025350	000764				BR GIN1		
11100	025352	020567	000232			18: CMP FIRST,FINISH		; TESTED ALL
11101	025356	001415				BEQ GIN3		; YES BRANCH
11102	025360	010567	000226			MOV FIRST,INST		; SET UP INST
11103	025364	005267	000222			GIN2: INC INST		
11104	025370	012767	025424	152412		MOV #RET,10		; SET UP RETURN FROM TRAP
11105	025376	012706	001000			MOV #STBOT,SP		; SET UP STACK POINTER
11106	025402	005067	152370			CLR CC		; CLEAR PRIORITY
11107	025406	000167	000200			JMP INST		; EXECUTE RESERVED INSTRUCTION
11108	025412	012767	021530	152370		GIN3: MOV #T010,10		; RESET VECTOR 10
11109	025420	000167	000252			JMP THRPRT		; JUMP TO EIS TEST
11110								
11111								; TRAPPING SHOULD SEND YOU HERE
11112	025424	020627	000774			RET: CMP SP,#STBOT-4		; TEST DECREMENT OF SP
11113	025430	001401				BEQ RET1		
11114	025432	104000				EMT		; WRONG DECREMENT
11115	025434	026727	153334	025614		RET1: CMP STBOT-4,#INST+2		; LOC OF INST UNINCREMENTED
11116	025442	001401				BEQ RET2		
11117	025444	104000				EMT		; INST INC ON TRAP
11118	025446	005767	153324			RET2: TST STBOT-2		
11119	025452	001401				BEQ RET3		
(1)	025454					RET4:		
(2)	025454	104000				EMT		; CONDITION CODES SET ON TRAP OR WRONG #STNPM
11120	025456	026701	000130			RET3: CMP INST,LAST		
11121	025462	001717				BEQ GIN1		; SET UP NEW GROUP
11122	025464	000167	177674			JMP GIN2		; FINISH OLD GROUP
11123								; END OF INSTRUCTION GROUP
11124	025470	000007				TABLE: 7		; END OF OPERATE
11125	025472	000077				77		
11126	025474	000207				207		; RTS,R:1,JMP
11127	025476	000227				227		
11128	025500	006777				6777		
11129	025502	007777				7777		
11130	025504	075037				075037		

11131 025506 076017
 11132 025510 076032
 11133 025512 076037
 11134 025514 076045
 11135 025516 076047
 11136 025520 076077
 11137 025522 076127
 11138 025524 076132
 11139 025526 076137
 11140 025530 076145
 11141 025532 076147
 11142 025534 076157
 11143 025536 076167
 11144 025540 076177
 11145 025542 076777
 11146 025544 076017
 11147 025546 076032
 11148 025550 076037
 11149 025552 076045
 11150 025554 076047
 11151 025556 076077
 11152 025560 076127
 11153 025562 076132
 11154 025564 076137
 11155 025566 076145
 11156 025570 076147
 11157 025572 076157
 11158 025574 076167
 11159 025576 076177
 11160 025600 167777
 11161 025602 177700
 11162 025604 177716
 11163 025606 177777
 11164 025610 025610
 11165 025612 000000
 11166 025614 000000
 11167 025616 000000
 11168 025620 000000
 11169 025622 000000
 11170
 11171
 11172
 11173 000000
 11174 000051
 11175 000176
 11176
 11177 025624
 11178 025626
 11179 025626
 11180 025630
 11181 025630
 11182 025632
 11183 025632
 11184 025634
 11185 025634
 11186 025636

76017
 76032
 76037
 76045
 76047
 76077
 76127
 76132
 76137
 76145
 76147
 76157
 76167
 76177
 76777
 CIS: 76017
 76032
 76037
 76045
 76047
 76077
 76127
 76132
 76137
 76145
 76147
 76157
 76167
 76177
 FPP: 167777
 177700
 177716
 177777
 FINISH:
 INST: HALT
 HALT
 HALT
 HALT
 HALT

; START OF THE FPP INSTRUCTIONS

;END FLAG
 ;WILL CONTINUE RESERVED INST
 ;SHOULD TRAP TO LOC 10
 ;LOC 10 SHOULD SEND YOU TO
 ;RET

.SBTTL ** STARTING OF EIS TEST **

DUMMY= 0
 F= 51
 N= 176
 COUNT:
 . =COUNT*2
 PWORD:
 . =PWORD*2
 TEMP1:
 . =TEMP1*2
 TEMP2:
 . =TEMP2*2
 TEMP3:
 . =TEMP3*2

11187 025636
11188 025640 025640
11189 025640 000000
11190 025642 000000
11191 025644 177771
11192 025646 025644
11193 025650 177772
11194 025652 177777
11195 025654 040000
11196 025656 025654
11197 025660 040000
11198 025662 177776
11199 025664 000002
11200 025666 025664
11201 025670 000002
11202 025672 177566
11203 025674 177564
11204
11205
11206
11207
11208
11209
11210

TEMP4: .-TEMP4.2
TEMP5: .WORD
TEMP6: .WORD
S1: 7
S2: S1
S3: -6
S4: -1
S5: 40000
S6: S5
S7: 40000
S8: 2
S9: 2
S10: S9
S11: 2
\$TPB: 177566
\$TPS: 177564


```
11321 025676          THRPT:
11322
11323 025676 012705 001004          MOV    @TESTN,R5      ;MAKE R5 POINT TO WHERE TEST # IS SAVED
11324 025702 005037 025624          CLR    @COUNT       ;CLEAR THE COUNTER
11325 025706 012715 000001          MOV    @1,(R5)       ;INITIALIZE TEST NUMBER
11326 025712 012706 001000          MOV    @STBOT,SP     ;** STACK AT STBOT **
11327 025716 012737 000001 025630 28:  MOV    @1,@TEMP1     ;TEMP1=1
11328 025724 005037 025632          CLR    @TEMP2        ;TEMP2=0
11329 025730 012737 000001 025634          MOV    @1,@TEMP3     ;TEMP3=1
11330 025736 005037 025636          CLR    @TEMP4        ;TEMP4=0
11331 025742 106427 000000          MTPS   @0
11332
11333
```

11338
 11339
 11340
 11341
 11342
 11343
 11344
 11345
 11346
 11347
 11348
 11349
 11350
 11351
 11352
 11353
 11354
 11355
 11356
 11357
 11358
 11362
 11366
 11367
 (2)
 11368
 11369
 11370
 (1)
 (2)
 11371
 11372
 11373
 11374
 11375
 11376
 11377
 11378
 11379
 11380
 11381
 11382
 11383
 11384
 11385
 11386
 11387
 11388
 11389
 11390
 11394
 11398
 11399
 (2)
 11400
 11401

025746 013700 025630
 025752 032737 000001 001006
 025760 001004
 025762 013701 025632
 025766 072001
 025770 000402
 025772 072067 177634
 025776 106737 025626
 026002 123737 025636 025626
 026010 001401
 026012 104000
 026014 005237 025624
 026020 023700 025634
 026024 001401
 026026
 026026 104000
 026030 021537 025624
 026034 001374
 026036 005215
 026040 021527 000037
 026044 002011
 026046 005237 025632
 026052 006367 177556
 026056 021527 000020
 026062 001004
 026064 000167 000670
 026070 004767 000712
 026074 013701 025630
 026100 032737 000001 001006
 026106 001004
 026110 013702 025632
 026114 072102
 026116 000402
 026120 072167 177506
 026124 106737 025626
 026130 123737 025636 025626
 026136 001401
 026140 104000
 026142 005237 025624
 026146 023701 025634

ASTART:
 24:
 44:
 114:
 64:
 124:
 84:
 REGR1:
 24:
 44:
 114:

MOV @TEMP1,#0
 BIT #1,@#PASS
 BNE 24
 MOV @TEMP2,R1
 ASH R1,R0
 BR 44
 ASH TEMP2,#0
 MFPS @#PSWORD
 CMPB @TEMP4,@#PSWORD
 BEQ 114
 EMT
 INC @#COUNT
 CMP @TEMP3,#0
 BEQ 124
 EMT
 CMP (R5),@#COUNT
 BNE 64
 INC (R5)
 CMP (R5),#37
 BGE 84
 INC @TEMP2
 ASL TEMP3
 CMP (R5),#20
 BNE REGR1
 JMP NEGAT
 JSR PC,TST37
 REGR1: MOV @TEMP1,#1
 BIT #1,@#PASS
 BNE 24
 MOV @TEMP2,R2
 ASH R2,R1
 BR 44
 ASH TEMP2,#1
 MFPS @#PSWORD
 CMPB @TEMP4,@#PSWORD
 BEQ 114
 EMT
 INC @#COUNT
 CMP @TEMP3,#1

;LOAD R0 WITH THE CONTENTS OF TEMP1
 ;IS IT AN EVEN PASS ?
 ;IF NOT THEN GO TO 24
 ;OTHERWISE EXECUTE THE INSTRUCTION
 ;IN MODE 0 USING R1
 ;SHIFT R0 BY THE NUMBER SPECIFIED BY TEMP2
 ;SAVE PS
 ;IS THE PS = TEMP4 ?
 ;THE PS IS NOT EQUAL TO 0
 ;INCREMENT THE COUNTER
 ;IS THE RESULT IN R0 EQUAL TO TEMP3?
 ;EITHER INCORRECT R0 OR INCORRECT SEQUENCE
 ;IS THE TEST NUMBER EQUAL TO THE
 ;COUNTER?
 ;IF NOT GO TO THE HLT ABOVE
 ;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT
 ;BY 14. AND RIGHT BY 14.?
 ;SHIFT TEMP3 LEFT.
 ;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
 ;IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
 ;IF SO GO AND CONTINUE THE REST OF THE PROGRAM
 ;LOAD R1 WITH THE CONTENTS OF TEMP1
 ;IS IT AN EVEN PASS ?
 ;IF NOT THEN GO TO 24
 ;OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
 ;USING R1
 ;SHIFT R1 BY THE NUMBER SPECIFIED BY TEMP2
 ;SAVE PS
 ;IS THE PS = TEMP4 ?
 ;THE PS IS NOT EQUAL TO 0
 ;INCREMENT THE COUNTER
 ;IS THE RESULT IN R1 EQUAL TO TEMP3?

 ; ASH INSTRUCTION TESTS

 ; TESTS 1-36

Address	OpCode	Register	Value	Label	Instruction	Comment
11464	026426	001401			BEG 124	
(1)	026430			64:		
(2)	026430	104000			EMT	; EITHER INCORRECT R3 OR INCORRECT SEQUENCE
11465	026432	021537	025624	124:	CMP (R5),#COUNT	; IS THE TEST NUMBER EQUAL TO THE COUNTER?
11466	026436	001374			BNE 64	; IF NOT GO TO THE HLT ABOVE
11467	026440	005215			INC (R5)	
11468	026442	021527	000037		CMP (R5),#37	; HAS THE CONTENTS OF REGISTERS BEEN SHIFTED
11469						; LEFT BY 14, AND RIGHT BY 14.?
11470	026446	002010			BGE 84	
11471	026450	005237	025632		INC #TEMP2	
11472	026454	006367	177154		ASL TEMP3	; SHIFT TEMP3 LEFT?
11473	026460	021527	000020		CMP (R5),#20	; HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
11474	026464	001003			BNE REGR4	
11475	026466	000534			BR NEGAT	; IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
11476	026470	004767	000312	84:	JSR PC,TST37	; IF SO GO AND CONTINUE THE REST OF THE PROGRAM
11477	026474	013704	025630	REGR4:	MOV #TEMP1,#4	; LOAD R4 WITH THE CONTENTS OF TEMP1
11478	026500	010501			MOV R5,R1	; SAVE R5
11479	026502	032737	000001	001006	BIT #1,#PASS	; IS IT AN EVEN PASS ?
11480	026510	001004			BNE 24	; IF NOT THEN GO TO 24
11481	026512	013705	025632		MOV #TEMP2,R5	; OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
11482	026516	072405			ASH R5,R4	; USING R4
11483	026520	000402			BR -4	
11484	026522	072467	177104	24:	ASH TEMP2,#4	; SHIFT R4 BY THE NUMBER SPECIFIED BY TEMP2
11488	026526	106737	025626	44:	MFPS #PSWORD	; SAVE PS
11492	026532	123737	025636	025626	CMPB #TEMP4,#PSWORD	; IS PS = TEMP4 ?
11493	026540	001401			BEQ 114	
(2)	026542	104000			EMT	; THE PS IS NOT EQUAL TO 0
11494	026544	005237	025624	114:	INC #COUNT	
11495	026550	023704	025634		CMP #TEMP3,#4	; IS THE RESULT IN R4 EQUAL TO TEMP3?
11496	026554	001401			BEQ 124	
(1)	026556			64:		
(2)	026556	104000			EMT	; EITHER INCORRECT R4 OR INCORRECT SEQUENCE
11497	026560	010105		124:	MOV R1,R5	; RESTORE R5
11498	026562	021537	025624		CMP (R5),#COUNT	; IS THE TEST NUMBER EQUAL TO THE COUNTER?
11499	026566	001373			BNE 64	; IF NOT GO TO THE HLT ABOVE
11500	026570	005215			INC (R5)	
11501	026572	021527	000037		CMP (R5),#37	; HAS THE CONTENTS OF REGISTERS BEEN
11502						; SHIFTED LEFT BY 14, AND RIGHT BY 14.?
11503	026576	002010			BGE 84	
11504	026600	005237	025632		INC #TEMP2	
11505	026604	006367	177024		ASL TEMP3	; SHIFT TEMP3 LEFT
11506	026610	021527	000020		CMP (R5),#20	; HAS THE CONTENTS OF REGISTER BEEN SHIFTED BY 14.?
11507	026614	001003			BNE REGR5	
11508	026616	000460			BR NEGAT	; IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
11509	026620	004767	000162	84:	JSR PC,TST37	; IF SO GO AND CONTINUE THE REST OF THE PROGRAM
11510	026624	010501		REGR5:	MOV R5,R1	; SAVE R5
11511	026626	013705	025630		MOV #TEMP1,#5	; LOAD R5 WITH THE CONTENTS OF TEMP1
11512	026632	032737	000001	001006	BIT #1,#PASS	; IS IT AN EVEN PASS ?
11513	026640	001004			BNE 24	; IF NOT THEN GO TO 24
11514	026642	013700	025632		MOV #TEMP2,R0	; OTHERWISE EXECUTE ASH INSTRUCTION IN MODE 0
11515	026646	072500			ASH R0,R5	; USING R5
11516	026650	000402			BR 44	
11517	026652	072567	176754	24:	ASH TEMP2,#5	; SHIFT R5 BY THE NUMBER SPECIFIED BY TEMP2
11521	026656	106737	025626	44:	MFPS #PSWORD	; SAVE PS
11525	026662	123737	025636	025626	CMPB #TEMP4,#PSWORD	; IS PS = TEMP4 ?
11526	026670	001401			BEQ 114	

(2)	026672	104000				EMT			;THE PS IS NOT EQUAL TO 0.
11527	026674	005237	025624		114:	INC	#COUNT		
11528	026700	023705	025634			CMP	#TEMP3,#5		;IS THE RESULT IN R5 EQUAL TO TEMP3?
11529	026704	001401				BEG	124		
(1)	026706				64:				
(2)	026706	104000				EMT			;EITHER INCORRECT R5 OR INCORRECT SEQUENCE
11530	026710	021137	025624		124:	CMP	(R1),#COUNT		;IS THE TEST NUMBER EQUAL TO THE COUNTER?
11531	026714	001374				BNE	64		;IF NOT GO TO THE HLT ABOVE
11532	026716	010105				MOV	R1,R5		;RESTORE R5
11533	026720	005215				INC	(R5)		
11534	026722	021527	000037			CMP	(R5),#37		;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED
11535									;LEFT BY 14. AND RIGHT BY 14.?
11536	026726	002010				BGE	84		;IF SO GO AND CONTINUE THE REST OF THE PROGRAM
11537	026730	005237	025632			INC	#TEMP2		
11538	026734	006367	176674			ASL	TEMP3		;SHIFT TEMP3 LEFT
11539	026740	021527	000020			CMP	(R5),#20		;HAS THE CONTENTS OF REGISTERS BEEN SHIFTED LEFT BY 14.?
11540	026744	001405				BEG	NEGAT		;IF SO GO TO NEGAT AND INITIATE RIGHT SHIFT
11541	026746	000402				BR	104		
11542	026750	004767	000032		84:	JSR	PC,TST37		
11543	026754	000167	176766		104:	JMP	ASTART		;GO BACK TO START
11544	026760	012737	040000	025630	NEGAT:	MOV	#40000,#TEMP1		;TEMP1=40000
11545	026766	012737	177762	025632		MOV	#177762,#TEMP2		;TEMP2=177762
11546	026774	012737	000001	025634		MOV	#1,#TEMP3		;TEMP3=1
11547	027002	000167	176740			JMP	ASTART		
11548	027006	021527	000037		TST37:	CMP	(R5),#37		;IS IT TEST 37?
11549	027012	001013				BNE	TST40		;IF NOT THEN TRY TEST 40
11550	027014	005037	025630			CLR	#TEMP1		;0
11551	027020	012737	000020	025632		MOV	#16,#TEMP2		;SHIFTED BY 16
11552	027026	005037	025634			CLR	#TEMP3		;IS=0
11553	027032	012737	000004	025636		MOV	#4,#TEMP4		;AND PS=4
11554	027040	000207				RTS	PC		
11555	027042	021527	000040		TST40:	CMP	(R5),#40		;IS IT TEST 40?
11556	027046	001003				BNE	TST41		;IF NOT THEN TRY TEST 41
11557	027050	005037	025632			CLR	#TEMP2		;0 SHIFTED BY 0=0 AND PS=4
11558	027054	000207				RTS	PC		
11559	027056	021527	000041		TST41:	CMP	(R5),#41		;IS IT TEST 41?
11560	027062	001004				BNE	TST42		;IF NOT THEN TRY TEST 42
11561	027064	012737	177760	025632		MOV	#-16,#TEMP2		;0 SHIFTED BY -16.=0 AND PS=4
11562	027072	000207				RTS	PC		
11563	027074	021527	000042		TST42:	CMP	(R5),#42		;IS IT TEST 42?
11564	027100	001013				BNE	TST43		;IF NOT THEN TRY TEST 43
11565	027102	012737	100000	025630		MOV	#100000,#TEMP1		;100000
11566	027110	005237	025632			INC	#TEMP2		;SHIFTED BY -15
11567	027114	005337	025634			DEC	#TEMP3		;IS=-1
11568	027120	012737	000010	025636		MOV	#10,#TEMP4		;AND PS=10
11569	027126	000207				RTS	PC		
11570	027130	021527	000043		TST43:	CMP	(R5),#43		;IS IT TEST 43?
11571	027134	001012				BNE	TST44		;IF NOT THEN IF NOT THEN TRY TEST 44
11572	027136	012737	125252	025630		MOV	#125252,#TEMP1		;125252
11573	027144	012737	177777	025632		MOV	#-1,#TEMP2		;SHIFTED BY 1
11574	027152	012737	152525	025634		MOV	#152525,#TEMP3		;IS=152525 AND PS=10
11575	027160	000207				RTS	PC		
11576	027162	021527	000044		TST44:	CMP	(R5),#44		;IS IT TEST 44?
11577	027166	001012				BNE	TST45		;IF NOT THEN TRY TEST 45
11578	027170	012737	000001	025632		MOV	#1,#TEMP2		;125252 SHIFTED BY 1
11579	027176	012737	052524	025634		MOV	#52524,#TEMP3		;IS=52524

```
11580 027204 012737 000003 025636      MOV      #3,#TEMP4      ;AND PS=3
11581 027212 000207                RTS      PC
11582 027214 021527 000045      TST45:  CMP      (R5),#45      ;IS IT TEST 45?
11583 027220 001012                BNE      TST46      ;IF NOT THEN TRY TEST 46
11584 027222 012737 177776 025632      MOV      #-2,#TEMP2      ;125252 SHIFTED BY 2
11585 027230 012737 165252 025634      MOV      #165252,#TEMP3    ;IS=165252
11586 027236 012737 000011 025636      MOV      #11,#TEMP4      ;AND PS=11
11587 027244 000207                RTS      PC
11588 027246 021527 000046      TST46:  CMP      (R5),#46      ;IS IT TEST 46?
11589 027252 001014                BNE      TST47      ;IF NOT THEN TRY TEST 47
11590 027254 012737 177777 025630      MOV      #-1,#TEMP1      ;-1
11591 027262 012737 000020 025632      MOV      #16,#TEMP2      ;SHIFTED BY 15.
11592 027270 005037 025634      CLR      #TEMP3          ;IS=0
11593 027274 012737 000007 025636      MOV      #7,#TEMP4      ;AND PS=7
11594 027302 000207                RTS      PC
11595 027304 021527 000047      TST47:  CMP      (R5),#47      ;IS IT TEST 47?
11596 027310 001011                BNE      TST50      ;IF NOT THEN TRY TEST 50
11597 027312 005337 025632      DEC      #TEMP2          ;-1 SHIFTED BY 15
11598 027316 012737 100000 025634      MOV      #100000,#TEMP3    ;IS=100000
11599 027324 012737 000011 025636      MOV      #11,#TEMP4      ;AND PS=11
11600 027332 000207                RTS      PC
11601 027334 021527 000050      TST50:  CMP      (R5),#50      ;IS IT TEST 50
11602 027340 001007                BNE      ENT51      ;IF NOT THEN TRY TEST 51
11603 027342 012737 137777 025630      MOV      #137777,#TEMP1    ;137777 SHIFTED BY 15. IS=100000
11604 027350 012737 000013 025636      MOV      #13,#TEMP4      ;AND PS=13
11605 027356 000207                RTS      PC
11606 027360 021527 000051      ENT51:  CMP      (R5),#51      ;IS IT ENTERING TEST 51?
11607 027364 001401                BEQ      1#           ;TEST NUMBER GOOFED
(2) 027366 104000                EMT
11608
11609 027370 005726                1#:     TST      (SP),#      ;RESTORE STACK POINTER
11610 027372 012704 177771      MOV      #-7,#4
11611 027376 012702 025644      MOV      #51,#2
11612 027402 012703 025646      MOV      #52,#3
11613
(1)
(1)
(1)
(1) 027406 012701 125252      TST51:  MOV      #125252,#1    ;LOAD R1 WITH 125252
(1) 027412 072127 000005      ASH      #5,#1          ;SHIFT R1 BY #5
(1) 027416 106737 025626      MFPS     #PSWORD        ;SAVE PS
(1) 027422 122737 000003 025626      CMPB     #3,#PSWORD     ;IS THE PS 3?
(2) 027430 001401                BEQ      11#           ;THE PS IS NOT EQUAL TO 3
(3) 027432 104000                EMT
(1) 027434 022701 052500      11#:    CMP      #52500,#1      ;IS THE RESULT 52500?
(2) 027440 001401                BEQ      12#
(3) 027442 104000                EMT
(1) 027444 005215      12#:    INC      (R5)        ;R1 IS NOT EQUAL TO 52500 OR INCORRECT SEQUENCE
(1)
(1)
(1)
11614
(1)
(1)
(1)
(1) 027446 012700 125252      TST52:  MOV      #125252,#0    ;LOAD R0 WITH 125252
```


(1) 027630 022700 177525
(2) 027634 001401
(3) 027636 104000
(1) 027640 005215
(1)
(1)
(1)

114: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 124
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
124: INC (R5)

11618

(1)
(1)
(1)
(1)
(1) 027642 012700 125252
(1) 027646 072042
(1) 027650 106737 025626
(1) 027654 122737 000010 025626
(2) 027662 001401
(3) 027664 104000
(1) 027666 022700 177525
(2) 027672 001401
(3) 027674 104000
(1) 027676 005215
(1)
(1)
(1)

;TEST:56 11/34 ASH 125252 SHIFTED BY -(2) = 177525 PS = 10

TST56: MOV #125252,#0 ;LOAD RO WITH 125252
ASH -(2),#0 ;SHIFT RO BY -(2)
MFPS @PPSWORD ;SAVE PS
CMPB #10,@PPSWORD ;IS THE PS 10?
BEQ 114
EMT ;THE PS IS NOT EQUAL TO 10
114: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 124
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
124: INC (R5)

11619

(1)
(1)
(1)
(1) 027700 012700 125252
(1) 027704 072063 000002
(1) 027710 106737 025626
(1) 027714 122737 000011 025626
(2) 027722 001401
(3) 027724 104000
(1) 027726 022700 177252
(2) 027732 001401
(3) 027734 104000
(1) 027736 005215
(1)
(1)
(1)

;TEST:57 11/34 ASH 125252 SHIFTED BY 2(3) = 177252 PS = 11

TST57: MOV #125252,#0 ;LOAD RO WITH 125252
ASH 2(3),#0 ;SHIFT RO BY 2(3)
MFPS @PPSWORD ;SAVE PS
CMPB #11,@PPSWORD ;IS THE PS 11?
BEQ 114
EMT ;THE PS IS NOT EQUAL TO 11
114: CMP #177252,#0 ;IS THE RESULT 177252?
BEQ 124
EMT ;RO IS NOT EQUAL TO 177252 OR INCORRECT SEQUENCE
124: INC (R5)

11620

(1)
(1)
(1)
(1) 027740 012700 125252
(1) 027744 072073 000000
(1) 027750 106737 025626
(1) 027754 122737 000010 025626
(2) 027762 001401
(3) 027764 104000
(1) 027766 022700 177525
(2) 027772 001401
(3) 027774 104000
(1) 027776 005215
(1)

;TEST:60 11/34 ASH 125252 SHIFTED BY 8(3) = 177525 PS = 10

TST60: MOV #125252,#0 ;LOAD RO WITH 125252
ASH 8(3),#0 ;SHIFT RO BY 8(3)
MFPS @PPSWORD ;SAVE PS
CMPB #10,@PPSWORD ;IS THE PS 10?
BEQ 114
EMT ;THE PS IS NOT EQUAL TO 10
114: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 124
EMT ;RO IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
124: INC (R5)

(1)
(1)
11621
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
11622
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

;TEST:61 11/34 ASH 125252 SHIFTED BY @ (3) = 177525 PS = 10

030000 012700 125252
030004 072033
030006 106737 025626
030012 122737 000010 025626
030020 001401
030022 104000
030024 022700 177525
030030 001401
030032 104000
030034 005215

TST61: MOV #125252,#0 ;LOAD R0 WITH 125252
ASH @ (3),#0 ;SHIFT R0 BY @ (3)
MFPS @PSWORD ;SAVE PS
CMPB #10,@PSWORD ;IS THE PS 10?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 10
11#: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 12#
EMT ;R0 IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
12#: INC (R5)

;TEST:62 11/34 ASH 125252 SHIFTED BY @-(3) = 177525 PS = 10

030036 012700 125252
030042 072033
030044 106737 025626
030050 122737 000010 025626
030056 001401
030060 104000
030062 022700 177525
030066 001401
030070 104000
030072 005215

TST62: MOV #125252,#0 ;LOAD R0 WITH 125252
ASH @-(3),#0 ;SHIFT R0 BY @-(3)
MFPS @PSWORD ;SAVE PS
CMPB #10,@PSWORD ;IS THE PS 10?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 10
11#: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 12#
EMT ;R0 IS NOT EQUAL TO 177525 OR INCORRECT SEQUENCE
12#: INC (R5)

N 1 1

1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1660
1664
1665
(2)
1666
1667
1668
(2)
1669
1670
1671
1672
(2)
1673
1674
1675
(2)
1676
1677
1678
1679
1680
1681
1682
1683
1684

: ASHC INSTRUCTION TESTS

: TESTS 63-157

```

030074 012737 000062 025624      MOV    #62,#COUNT
030102 005037 025630      CLR    #TEMP1           ;TEMP1=0
030106 012737 000001 025632      MOV    #1,#TEMP2       ;TEMP2=1
030114 005037 025634      CLR    #TEMP3           ;TEMP3=0
030120 005037 025636      CLR    #TEMP4           ;TEMP4=0
030124 012737 000001 025640      MOV    #1,#TEMP5       ;TEMP5=1
030132 005037 025642      CLR    #TEMP6           ;0 1 SHIFTED BY 0=0 1, PS=0

REG01: MOV    R5,R2           ;SAVE R5
        MOV    #TEMP1,#0   ;PLACE THE CONTENTS OF TEMP1 IN REGISTER 0
        MOV    #TEMP2,#0:1 ;PLACE THE CONTENTS OF TEMP2 IN REGISTER 1
        CLC
        BIT    #1,#PASS    ;IS IT AN EVEN PASS ?
        BNE    2#         ;IF NOT THEN GO TO 2#
        MOV    #TEMP3,R5   ;OTHERWISE EXECUTE ASHC INSTRUCTION IN MODE 0
        ASHC  R5,R0       ;USING R0
        BR    4#
        ASHC  TEMP3,#0    ;ASHC REGISTER 0 BY THE CONTENTS OF TEMP3
        MFPS  #PSWORD     ;SAVE PS
        CMPB  #TEMP6,#PSWORD ;COMPARE PS WITH THE CONTENTS OF TEMP6
        BEQ   11#
        EMT                    ;WRONG PS
        INC   #COUNT     ;IS THE RESULT IN R0 SAME AS TEMP4?
        CMP   #TEMP4,#0
        BEQ   12#
        EMT                    ;WRONG RESULT IN R0
        CMP   #TEMP5,#1   ;IS THE RESULT IN R1 SAME AS TEMP5?
        ;TEMP1 TEMP2 SHIFTED BY TEMP3-TEMP4 TEMPS
        ;AND PS=TEMP6
        BEQ   13#
        EMT                    ;WRONG RESULT IN R1
        MOV   R2,R5       ;RESTORE R5
        CMP   (R5),#COUNT ;IS TEST NUMBER=COUNTER?
        BEQ   14#
        EMT                    ;NO
        INC   (R5)
        CMP   (R5),#160   ;HAVE THE FIRST 159 TEST BEEN EXECUTED?
        BGE   6#         ;YES
        INC   #TEMP3
        CLC
        ROL   #TEMP5     ;ROTATE TEMPS LEFT BY 1 PLACE
        ROL   #TEMP4     ;INTRODUCE CARRY FROM TEMP4 IN TEMP5
        CMP   (R5),#121  ;IS IT TEST 121?
        BNE   REGR23

```

11

11

11685	030306	004467	000344			JSR	R4,RITSH	IF SO THEN GO AND INITIATE RIGHT SHIFT
11686	030312	004767	000374		68:	JSR	#7,TST160	
11687	030316	013702	025630		REGR23:	MOV	#0TEMP1,#2	PLACE THE CONTENTS OF TEMP1 IN REGISTER 2
11688	030322	013703	025632			MOV	#0TEMP2,#2!1	PLACE THE CONTENTS OF TEMP2 IN REGISTER 3
11689	030326	000241				CLC		
11690	030330	032737	000001	001006		BIT	#1,#0!PASS	IS IT AN EVEN PASS ?
11691	030336	001004				BNE	2#	IF NOT THEN GO TO 2#
11692	030340	013704	025634			MOV	#0TEMP3,R4	OTHERWISE EXECUTE ASMC INSTRUCTION IN MODE 0
11693	030344	073204				ASMC	R4,R2	USING R2
11694	030346	000402				BR	4#	
11695	030350	073267	175260		2#:	ASMC	TEMP3,#2	ASMC REGISTER 2 BY THE CONTENTS OF TEMP3
11699	030354	106737	025626		4#:	MFPS	#0PSWORD	SAVE PS
11703	030360	123737	025642	025626		CMPS	#0TEMP6,#0PSWORD	COMPARE PS WITH THE CONTENTS OF TEMP6
11704	030366	001401				BEQ	11#	
(2)	030370	104000				EMT		WRONG PS
11705	030372	005237	025624		11#:	INC	#0COUNT	
11706	030376	023702	025636			CMPS	#0TEMP4,#2	IS THE RESULT IN R2 SAME AS TEMP4?
11707	030402	001401				BEQ	12#	
(2)	030404	104000				EMT		WRONG RESULT IN R2
11708	030406	023703	025640		12#:	CMPS	#0TEMP5,#3	IS THE RESULT IN R3 SAME AS TEMP5? TEMP1 TEMP2 SHIFTED BY TEMP3-TEMP4 AND PS=TEMP6
11709								
11710								
11711	030412	001401				BEQ	13#	
(2)	030414	104000				EMT		WRONG RESULT IN R1
11712	030416	021537	025624		13#:	CMPS	(R5),#0COUNT	IS TEST NUMBER=COUNTER?
11713	030422	001401				BEQ	14#	
(2)	030424	104000				EMT		NO
11714	030426	005215			14#:	INC	(R5)	
11715	030430	021527	000160			CMPS	(R5),#0160	HAVE THE FIRST 159 TEST BEEN EXECUTED?
11716	030434	002014				BGE	6#	YES
11717	030436	005237	025634			INC	#0TEMP3	
11718	030442	000241				CLC		
11719	030444	006137	025640			ROL	#0TEMP5	ROTATE TEMP5 LEFT BY 1 PLACE
11720	030450	006137	025636			ROL	#0TEMP4	INTRODUCE CARRY FROM TEMP5 IN TEMP4
11721	030454	021527	000121			CMPS	(R5),#0121	IS IT TEST 121?
11722	030460	001004				BNE	REG45	
11723	030462	004467	000170			JSR	R4,RITSH	IF SO THEN GO AND INITIATE RIGHT SHIFT
11724	030466	004767	000220		68:	JSR	#7,TST160	
11725	030472	010501			REG45:	MOV	R5,R1	SAVE R5
11726	030474	013704	025630			MOV	#0TEMP1,#4	PLACE THE CONTENTS OF TEMP1 IN REGISTER 4
11727	030500	013705	025632			MOV	#0TEMP2,#4!1	PLACE THE CONTENTS OF TEMP2 IN REGISTER 5
11728	030504	000241				CLC		
11729	030506	032737	000001	001006		BIT	#1,#0!PASS	IS IT AN EVEN PASS ?
11730	030514	001004				BNE	2#	IF NOT THEN GO TO 2#
11731	030516	013700	025634			MOV	#0TEMP3,R0	OTHERWISE EXECUTE ASMC INSTRUCTION IN MODE 0
11732	030522	073400				ASMC	R0,R4	USING R4
11733	030524	000402				BR	4#	
11734	030526	073467	175102		2#:	ASMC	TEMP3,#4	ASMC REGISTER 4 BY THE CONTENTS OF TEMP3
11738	030532	106737	025626		4#:	MFPS	#0PSWORD	SAVE PS
11742	030536	123737	025642	025626		CMPS	#0TEMP6,#0PSWORD	COMPARE PS WITH THE CONTENTS OF TEMP6
11743	030544	001401				BEQ	11#	
(2)	030546	104000				EMT		WRONG PS
11744	030550	005237	025624		11#:	INC	#0COUNT	
11745	030554	023704	025636			CMPS	#0TEMP4,#4	IS THE RESULT IN R4 SAME AS TEMP4?
11746	030560	001401				BEQ	12#	
(2)	030562	104000				EMT		WRONG RESULT IN R4

11747	030564	023705	025640	128:	CMP	@TEMP5,#5	IS THE RESULT IN R5 SAME AS TEMP5? TEMP1 TEMP2 SHIFTED BY TEMP3-TEMP4 TEMP5 AND PS=TEMP6
11748							
11749							
11750	030570	001401			BEQ	138	
(2)	030572	104000			EMT		WRONG RESULT IN R5
11751	030574	021137	025624	138:	CMP	(R1),@COUNT	IS TEST NUMBER=COUNTER?
11752	030600	001401			BEQ	148	
(2)	030602	104000			EMT		NO
11753	030604	010105		148:	MOV	R1,R5	RESTORE R5
11754	030606	005215			INC	(R5)	
11755	030610	021527	000160		CMP	(R5),#160	HAVE THE FIRST 159 TEST BEEN EXECUTED?
11756	030614	002014			BGE	#1	YES
11757	030616	005237	025634		INC	@TEMP3	
11758	030622	000241			CLC		
11759	030624	006137	025640		ROL	@TEMP5	ROTATE TEMP5 LEFT BY 1 PLACE
11760	030630	006137	025636		ROL	@TEMP4	INTRODUCE CARRY FROM TEMP5 IN TEMP4
11761	030634	021527	000121		CMP	(R5),#121	IS IT TEST 121?
11762	030640	001004			BNE	#1	
11763	030642	004467	000010		JSR	R4,RITSH	IF SO THEN GO AND INITIATE RIGHT SHIFT
11764	030646	004767	000040	68:	JSR	#7,TST160	
11765	030652	000167	177260	88:	JMP	REG01	
11766	030656	022424		RITSH:	CMP	(R4),@(R4)	MAKE R4 POINT TO THE NEXT REG TAG
11767	030660	012737	040000	025630	MOV	#4000,@TEMP1	TEMP1=4000
11768	030666	005037	025632		CLR	@TEMP2	TEMP2=0
11769	030672	012737	177742	025634	MOV	#-30,@TEMP3	TEMP3=-30
11770	030700	005037	025636		CLR	@TEMP4	TEMP4=0
11771	030704	005237	025640		INC	@TEMP5	TEMP5=1
11772	030710	000204			RTS	R4	
11773	030712	021527	000160	TST160:	CMP	(R5),#160	IS IT TEST 160
11774	030716	001010			BNE	TST161	IF NOT THEN TRY TEST 161
11775	030720	005037	025630		CLR	@TEMP1	0 0 SHIFTED BY 0
11776	030724	005037	025636		CLR	@TEMP4	IS EQUAL TO 0 0
11777	030730	012737	000004	025642	MOV	#4,@TEMP6	AND PS=4
11778	030736	000207			RTS	#7	
11779	030740	021527	000161	TST161:	CMP	(R5),#161	IS IT TEST 161
11780	030744	001004			BNE	TST162	
11781	030746	012737	177746	025634	MOV	#-32,@TEMP3	0 0 SHIFTED BY -32=0 0, PS=4
11782	030754	000207			RTS	#7	
11783	030756	021527	000162	TST162:	CMP	(R5),#162	IS IT TEST 162
11784	030762	001004			BNE	TST163	IF NOT THEN TRY TEST 163
11785	030764	012737	000032	025634	MOV	#32,@TEMP3	0 0 SHIFTED BY 32=0 0, PS=4
11786	030772	000207			RTS	#7	
11787	030774	021527	000163	TST163:	CMP	(R5),#163	IS IT TEST 163?
11788	031000	001016			BNE	TST164	IF NOT THEN TRY TEST 164
11789	031002	012737	052525	025630	MOV	#52525,@TEMP1	52525 0
11790	031010	012737	177760	025634	MOV	#-16,@TEMP3	SHIFTED BY -16.
11791	031016	005037	025636		CLR	@TEMP4	
11792	031022	012737	052525	025640	MOV	#52525,@TEMP5	IS EQUAL TO 0 52525
11793	031030	005037	025642		CLR	@TEMP6	AND PS = 0
11794	031034	000207			RTS	#7	
11795	031036	021527	000164	TST164:	CMP	(R5),#164	IS IT TEST 164?
11796	031042	001014			BNE	TST165	IF NOT THEN TRY TEST 165
11797	031044	012737	125252	025630	MOV	#125252,@TEMP1	125252 0 SHIFTED BY -16.
11798	031052	005337	025636		DEC	@TEMP4	
11799	031056	012737	125252	025640	MOV	#125252,@TEMP5	IS EQUAL TO -1 125252
11800	031064	012737	000010	025642	MOV	#10,@TEMP6	AND PS=10

11801	031072	000207			RTS	#7	
11802	031074	021527	000165		TST165: CMP	(R5),#165	:IS IT TEST 165?
11803	031100	001007			BNE	TST166	:IF NOT THEN TRY TEST 166
11804	031102	012737	177777	025630	MOV	#-1,#TEMP1	: 1 0 SHIFTED BY 16
11805	031110	012737	177777	025640	MOV	#-1,#TEMP5	:IS EQUAL TO -1 1. AND PS=10
11806	031116	000207			RTS	#7	
11807	031120	021527	000166		TST166: CMP	(R5),#166	:IS IT TEST 166?
11808	031124	001011			BNE	TST167	:IF NOT THEN TRY TEST 167
11809	031126	012737	100000	025630	MOV	#100000,#TEMP1	:100000 0
11810	031134	012737	177740	025634	MOV	#-32.,#TEMP3	:SHIFTED BY -32 IS EQUAL TO -1 -1
11811	031142	005237	025642		INC	#TEMP6	:AND PS=11
11812	031146	000207			RTS	#7	
11813	031150	021527	000167		TST167: CMP	(R5),#167	:IS IT TEST 167?
11814	031154	001014			BNE	TST170	:IF NOT THEN TRY TEST 170

E12

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07 JAN-85 09:05

MAC:11 30(1046) 07 JAN-85 09:28 PAGE 13
ASMC INSTRUCTION TESTS

SEQ 0147

11816 031156 005037 025630
11817 031162 005337 025632

CLR @TEMP1
DEC @TEMP2 :0 1

11819 031166 012737 000020 025634 MOV #16,@TEMP3 ;SHIFTED BY 16.
11820 031174 005037 025640 CLR @TEMP5 ;IS EQUAL TO 10
11821 031200 005237 025642 INC @TEMP6 ;AND PS-12
11822 031204 000207 RTS #7
11823 031206 021527 000170 TST170: CMP (R5),#170 ;IS IT TEST 170?

G12

CJKL580 LCP-5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 15
ASMC INSTRUCTION TESTS

SEQ 0149

11825 031212 001007

BNE TST171

;IF NOT THEN TRY TEST 171

11827	031214	012737	125252	025632	MOV	#125252,@TEMP2	;0 125252 SHIFTED BY 16
11828	031222	012737	125252	025636	MOV	#125252,@TEMP4	;IS EQUAL TO 125252 0, AND PS=12
11829	031230	000207			RTS	#7	
11830	031232	021527	000171		TST171: CMP	(R5),#171	;IS IT TEST 171?
11831	031236	001010			BNE	TST172	;IF NOT THEN TRY TEST 172
11832	031240	005337	025634		DEC	@TEMP3	;0 125252 SHIFTED BY 15
11833	031244	012737	052525	025636	MOV	#52525,@TEMP4	;IS EQUAL TO 52525 0
11834	031252	005037	025642		CLR	@TEMP6	;AND PS=0
11835	031256	000207			RTS	#7	
11836	031260	021527	000172		TST172: CMP	(R5),#172	;IS IT TEST 172?
11837	031264	001006			BNE	TST173	;IF NOT THEN TRY TEST 173
11838	031266	012737	052525	025632	MOV	#52525,@TEMP2	;0 52525
11839	031274	005237	025634		INC	@TEMP3	;SHIFTED BY 16. IS EQUAL TO 52525 0, AND PS=0
11840	031300	000207			RTS	#7	
11841	031302	021527	000173		TST173: CMP	(R5),#173	;IS IT TEST 173?
11842	031306	001014			BNE	TST174	;IF NOT THEN TRY TEST 174
11843	031310	012737	177777	025632	MOV	#-1,@TEMP2	;0 -1
11844	031316	005337	025634		DEC	@TEMP3	;SHIFTED BY 15.
11845	031322	012737	077777	025636	MOV	#77777,@TEMP4	
11846	031330	012737	100000	025640	MOV	#100000,@TEMP5	;IS EQUAL TO 77777 100000, AND PS=0
11847	031336	000207			RTS	#7	
11848	031340	021527	000174		TST174: CMP	(R5),#174	;IS IT TEST 174?
11849	031344	001013			BNE	TST175	;IF NOT THEN TRY TEST 175
11850	031346	012737	100000	025630	MOV	#100000,@TEMP1	
11851	031354	005337	025632		DEC	@TEMP2	;100000 -2 SHIFTED BY 15.
11852	031360	005037	025640		CLR	@TEMP5	;IS EQUAL TO 77777 0
11853	031364	012737	000002	025642	MOV	#2,@TEMP6	;AND PS=2
11854	031372	000207			RTS	#7	
11855	031374	021527	000175		TST175: CMP	(R5),#175	;IS IT TEST 175?
11856	031400	001015			BNE	ENT176	;IF NOT THEN TRY TEST 176
11857	031402	012737	177777	025630	MOV	#-1,@TEMP1	
11858	031410	005037	025632		CLR	@TEMP2	; -1 0
11859	031414	005237	025634		INC	@TEMP3	;SHIFTED BY 16.
11860	031420	005037	025636		CLR	@TEMP4	;IS EQUAL TO 0 0
11861	031424	012737	000007	025642	MOV	#7,@TEMP6	;AND PS=7

11863 031432 000207
 11864 031434 021527 000176
 11865 031440 001401
 (2) 031442 104000
 11866
 11867 031444 005726
 11868
 11869
 (1)
 (1)
 (1)
 (1) 031446
 (1) 031446 012701 000000
 (1) 031452 012701 000001
 (1) 031456 000241
 (1) 031460 073127 000010
 (1) 031464 106737 025626
 (1) 031470 122737 000000 025626
 (2) 031476 001401
 (3) 031500 104000
 (1) 031502 022701 000400
 (2) 031506 001401
 (3) 031510 104000
 (1) 031512
 (1) 031512 005215
 (1)
 (1)
 11870
 (1)
 (1)
 (1)
 (1) 031514
 (1) 031514 012703 000000
 (1) 031520 012703 177777
 (1) 031524 000241
 (1) 031526 073327 000017
 (1) 031532 106737 025626
 (1) 031536 122737 000011 025626
 (2) 031544 001401
 (3) 031546 104000
 (1) 031550 022703 100000
 (2) 031554 001401
 (3) 031556 104000
 (1) 031560
 (1) 031560 005215
 (1)
 (1)
 11871
 (1)
 (1)
 (1)
 (1) 031562
 (1) 031562 010501
 (1) 031564 012705 000000
 (1) 031570 012705 052525
 (1) 031574 000241

```

RTS      #7
ENT176:  CMP      (R5),#176      ;IS THE PROGRAM ENTERING TEST 176?
          BEQ      1#           ;
          EMT
          ;TEST NUMBER GOOFED
1# :      TST      (SP)          ;RESTORE STACK POINTER
;*****
;TEST:176      1 SHIFTED BY 8. = 400 PS = 0
;*****
TST176:
MOV      #DUMMY,#1      ;LOAD R1 WITH DUMMY
MOV      #1,#1!1        ;LOAD R1!1 WITH 1
CLC
ASMC     #8.,#1          ;SHIFT R1,R1!1 BY 8.
MFPS     @#PSWORD        ;SAVE PS
CMPB     #0,@#PSWORD     ;IS THE PS 0?
BEQ      11#
EMT
          ;THE PS IS NOT EQUAL TO 0
11# :     CMP      #400,#1        ;IS THE RESULT 400?
          BEQ      13#
          EMT
          ;R1 IS NOT EQUAL TO 400
13# :     INC      (R5)

;*****
;TEST:177      -1 SHIFTED BY 15. = 100000 PS = 11
;*****
TST177:
MOV      #DUMMY,#3      ;LOAD R3 WITH DUMMY
MOV      #-1,#3!1       ;LOAD R3!1 WITH 1
CLC
ASMC     #15.,#3         ;SHIFT R3,R3!1 BY 15.
MFPS     @#PSWORD        ;SAVE PS
CMPB     #11,@#PSWORD    ;IS THE PS 11?
BEQ      11#
EMT
          ;THE PS IS NOT EQUAL TO 11
11# :     CMP      #100000,#3     ;IS THE RESULT 100000?
          BEQ      13#
          EMT
          ;R3 IS NOT EQUAL TO 100000
13# :     INC      (R5)

;*****
;TEST:200      52525 SHIFTED BY 0 = 52525 PS = 0
;*****
TST200:
MOV      R5,R1          ;SAVE R5
MOV      #DUMMY,#5      ;LOAD R5 WITH DUMMY
MOV      #52525,#5!1    ;LOAD R5!1 WITH 52525
CLC

```

J12

(1) 031576 073527 000000
(1) 031602 106737 025626
(1) 031606 122737 000000 025626
(2) 031614 001401
(3) 031616 104000
(1) 031620 022705 052525
(2) 031624 001401
(3) 031626 104000
(1) 031630
(1) 031630 010105
(1) 031632 005215
(1)
(1)

ASMC #0,#5 ;SHIFT R5,R5!1 BY 0
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS THE PS 0?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 0
11#: CMP #52525,#5 ;IS THE RESULT 52525?
BEQ 13#
EMT ;R5 IS NOT EQUAL TO 52525
13#: MOV R1,R5 ;RESTORE R5
INC (R5)

11872

(1)
(1)
(1)
(1) 031634
(1) 031634 012701 000000
(1) 031640 012701 020010
(1) 031644 000241
(1) 031646 073127 177763
(1) 031652 106737 025626
(1) 031656 122737 000000 025626
(2) 031664 001401
(3) 031666 104000
(1) 031670 022701 000101
(2) 031674 001401
(3) 031676 104000
(1) 031700
(1) 031700 005215
(1)
(1)

;TEST:201 20010 SHIFTED BY -13. = 101 PS = 0

TST201:
MOV #DUMMY,#1 ;LOAD R1 WITH DUMMY
MOV #20010,#1!1 ;LOAD R1!1 WITH 20010
CLC
ASMC #-13.,#1 ;SHIFT R1,R1!1 BY -13.
MFPS @#PSWORD ;SAVE PS
CMPB #0,@#PSWORD ;IS THE PS 0?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 0
11#: CMP #101,#1 ;IS THE RESULT 101?
BEQ 13#
EMT ;R1 IS NOT EQUAL TO 101
13#: INC (R5)

11873

(1)
(1)
(1)
(1) 031702
(1) 031702 012703 000000
(1) 031706 012703 177777
(1) 031712 000241
(1) 031714 073327 000020
(1) 031720 106737 025626
(1) 031724 122737 000011 025626
(2) 031732 001401
(3) 031734 104000
(1) 031736 022703 000000
(2) 031742 001401
(3) 031744 104000
(1) 031746
(1) 031746 005215
(1)
(1)

;TEST:202 -1 SHIFTED BY 16. = 0 PS = 11

TST202:
MOV #DUMMY,#3 ;LOAD R3 WITH DUMMY
MOV #-1,#3!1 ;LOAD R3!1 WITH -1
CLC
ASMC #16.,#3 ;SHIFT R3,R3!1 BY 16.
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS THE PS 11?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 11
11#: CMP #0,#3 ;IS THE RESULT 0?
BEQ 13#
EMT ;R3 IS NOT EQUAL TO 0
13#: INC (R5)

11874

(1)
(1)

;TEST:203 1 SHIFTED BY -1 = 100000 PS = 1

(1)
(1) 031750
(1) 031750 010501
(1) 031752 012705 000000
(1) 031756 012705 000001
(1) 031762 000241
(1) 031764 073527 177777
(1) 031770 106737 025626
(1) 031774 122737 000001 025626
(2) 032002 001401
(3) 032004 104000
(1) 032006 022705 100000
(2) 032012 001401
(3) 032014 104000
(1) 032016
(1) 032016 010105
(1) 032020 005215
(1)
(1)

TST203:
MOV R5,R1 ;SAVE R5
MOV #DUMMY,#5 ;LOAD R5 WITH DUMMY
MOV #1,#5!1 ;LOAD R5!1 WITH 1
CLC
ASMC #-1,#5 ;SHIFT R5,R5!1 BY 1
MFPS #PASSWORD ;SAVE PS
CMPB #1,#PASSWORD ;IS THE PS 1?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 1
11#: CMP #100000,#5 ;IS THE RESULT 100000?
BEQ 13#
EMT ;R5 IS NOT EQUAL TO 100000
13#: MOV R1,R5 ;RESTORE R5
INC (R5)

;TEST:204 125252 SHIFTED BY -16. = 125252 PS = 11

11875
(1)
(1)
(1)
(1) 032022
(1) 032022 012701 000000
(1) 032026 012701 125252
(1) 032032 000241
(1) 032034 073127 177760
(1) 032040 106737 025626
(1) 032044 122737 000011 025626
(2) 032052 001401
(3) 032054 104000
(1) 032056 022701 125252
(2) 032062 001401
(3) 032064 104000
(1) 032066
(1) 032066 005215
(1)
(1)

TST204:
MOV #DUMMY,#1 ;LOAD R1 WITH DUMMY
MOV #125252,#1!1 ;LOAD R1!1 WITH 125252
CLC
ASMC #-16.,#1 ;SHIFT R1,R1!1 BY -16.
MFPS #PASSWORD ;SAVE PS
CMPB #11,#PASSWORD ;IS THE PS 11?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 11
11#: CMP #125252,#1 ;IS THE RESULT 125252?
BEQ 13#
EMT ;R1 IS NOT EQUAL TO 125252
13#: INC (R5)

;TEST:205 125252 125252 SHIFTED BY 21. = 52500 000000 PS = 3

11876
(1)
(1)
(1)
(1) 032070
(1) 032070 012702 125252
(1) 032074 012703 125252
(1) 032100 000241
(1) 032102 073227 000025
(1) 032106 106737 025626
(1) 032112 122737 000003 025626
(2) 032120 001401
(3) 032122 104000
(1) 032124 022702 052500
(2) 032130 001401
(3) 032132 104000
(1) 032134 022703 000000

TST205:
MOV #125252,#2 ;LOAD R2 WITH 125252
MOV #125252,#2!1 ;LOAD R2!1 WITH 125252
CLC
ASMC #21.,#2 ;SHIFT R2,R2!1 BY 21.
MFPS #PASSWORD ;SAVE PS
CMPB #3,#PASSWORD ;IS THE PS 3?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 3
11#: CMP #52500,#2 ;IS THE RESULT 52500?
BEQ 12#
EMT ;R2 IS NOT EQUAL TO 52500
12#: CMP #000000,#2!1 ;IS THE RESULT 000000?

(2) 032140 001401
(3) 032142 104000
(1) 032144
(1) 032144 005215
(1)
(1)
11877
11878 032146 012702 177771
11879 032152 012703 025644
11880 032156 012704 025646
11881
11882

BEQ 13#
EMT ;R2:1 IS NOT EQUAL TO 000000
13#:
INC (P5)

MOV #-7,#2
MOV #S1,#3
MOV #S2,#4

;TEST:206 125252 125252 SHIFTED BY S1 = 177525 52525 PS = 10

TST206:

(1) 032162
(1) 032162 012700 125252
(1) 032166 012701 125252
(1) 032172 000241
(1) 032174 073067 173444
(1) 032200 106737 025626
(1) 032204 122737 000010 025626
(2) 032212 001401
(3) 032214 104000
(1) 032216 022700 177525
(2) 032222 001401
(3) 032224 104000
(1) 032226 022701 052525
(2) 032232 001401
(3) 032234 104000
(1) 032236
(1) 032236 005215
(1)
(1)

MOV #125252,#0 ;LOAD R0 WITH 125252
MOV #125252,#0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC S1,#0 ;SHIFT R0,R0!1 BY S1
MFPS @#PSW:ORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 10
11#:
CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 12#
EMT ;R0 IS NOT EQUAL TO 177525
12#:
CMP #52525,#0!1 ;IS THE RESULT 52525?
BEQ 13#
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13#:
INC (R5)

;TEST:207 125252 125252 SHIFTED BY #S2 = 177525 52525 PS = 10

TST207:

(1) 032240
(1) 032240 012700 125252
(1) 032244 012701 125252
(1) 032250 000241
(1) 032252 073077 173370
(1) 032256 106737 025626
(1) 032262 122737 000010 025626
(2) 032270 001401
(3) 032272 104000
(1) 032274 022700 177525
(2) 032300 001401
(3) 032302 104000
(1) 032304 022701 052525
(2) 032310 001401
(3) 032312 104000
(1) 032314
(1) 032314 005215
(1)

MOV #125252,#0 ;LOAD R0 WITH 125252
MOV #125252,#0!1 ;LOAD R0!1 WITH 125252
CLC
ASHC #S2,#0 ;SHIFT R0,R0!1 BY #S2
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS THE PS 10?
BEQ 11#
EMT ;THE PS IS NOT EQUAL TO 10
11#:
CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 12#
EMT ;R0 IS NOT EQUAL TO 177525
12#:
CMP #52525,#0!1 ;IS THE RESULT 52525?
BEQ 13#
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
13#:
INC (R5)

```

(1)
11884
(1)
(1)
(1)
(1) 032316
(1) 032316 012700 125252
(1) 032322 012701 125252
(1) 032326 000241
(1) 032330 073037 025644
(1) 032334 106737 025626
(1) 032340 122737 000010 025626
(2) 032346 001401
(3) 032350 104000
(1) 032352 022700 177525
(2) 032356 001401
(3) 032360 104000
(1) 032362 022701 052525
(2) 032366 001401
(3) 032370 104000
(1) 032372
(1) 032372 005215
(1)
(1)

```

```

;*****
;TEST:210      125252 125252 SHIFTED BY @S1 = 177525 52525 PS = 10
;*****
TST210:
MOV      #125252,#0          ;LOAD R0 WITH 125252
MOV      #125252,#0!1       ;LOAD R0!1 WITH 125252
CLC
ASMC     @S1,#0              ;SHIFT R0,R0!1 BY @S1
MFPS     @PPSWORD           ;SAVE PS
CMPB     #10,@PPSWORD       ;IS THE PS 10?
BEQ      11$
EMT
11$:    CMP      #177525,#0   ;THE PS IS NOT EQUAL TO 10
BEQ      12$                ;IS THE RESULT 177525?
EMT
12$:    CMP      #52525,#0!1  ;R0 IS NOT EQUAL TO 177525
BEQ      13$                ;IS THE RESULT 52525?
EMT
13$:    EMT                    ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
INC      (R5)

```

```

11885
(1)
(1)
(1)
(1) 032374
(1) 032374 012700 125252
(1) 032400 012701 125252
(1) 032404 000241
(1) 032406 073013
(1) 032410 106737 025626
(1) 032414 122737 000010 025626
(2) 032422 001401
(3) 032424 104000
(1) 032426 022700 177525
(2) 032432 001401
(3) 032434 104000
(1) 032436 022701 052525
(2) 032442 001401
(3) 032444 104000
(1) 032446
(1) 032446 005215
(1)
(1)

```

```

;*****
;TEST:211      125252 125252 SHIFTED BY (3) = 177525 52525 PS = 10
;*****
TST211:
MOV      #125252,#0          ;LOAD R0 WITH 125252
MOV      #125252,#0!1       ;LOAD R0!1 WITH 125252
CLC
ASMC     (3),#0              ;SHIFT R0,R0!1 BY (3)
MFPS     @PPSWORD           ;SAVE PS
CMPB     #10,@PPSWORD       ;IS THE PS 10?
BEQ      11$
EMT
11$:    CMP      #177525,#0   ;THE PS IS NOT EQUAL TO 10
BEQ      12$                ;IS THE RESULT 177525?
EMT
12$:    CMP      #52525,#0!1  ;R0 IS NOT EQUAL TO 177525
BEQ      13$                ;IS THE RESULT 52525?
EMT
13$:    EMT                    ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
INC      (R5)

```

```

11886
(1)
(1)
(1)
(1) 032450
(1) 032450 012700 125252
(1) 032454 012701 125252
(1) 032460 000241
(1) 032462 073023

```

```

;*****
;TEST:212      125252 125252 SHIFTED BY (3) = 177525 52525 PS = 10
;*****
TST212:
MOV      #125252,#0          ;LOAD R0 WITH 125252
MOV      #125252,#0!1       ;LOAD R0!1 WITH 125252
CLC
ASMC     (3),#0              ;SHIFT R0,R0!1 BY (3)

```

NIC

```
(1) 032464 106737 025626 MFPS @PASSWORD ;SAVE PS
(1) 032470 122737 000010 025626 CMPB @10,@PASSWORD ;IS THE PS 10?
(2) 032476 001401 BEQ 11#
(3) 032500 104000 EMT ;THE PS IS NOT EQUAL TO 10
(1) 032502 022700 177525 11# : CMP @177525,#0 ;IS THE RESULT 177525?
(2) 032506 001401 BEQ 12#
(3) 032510 104000 EMT ;RO IS NOT EQUAL TO 177525
(1) 032512 022701 052525 12# : CMP @52525,#0!1 ;IS THE RESULT 52525?
(2) 032516 001401 BEQ 13#
(3) 032520 104000 EMT ;RO!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
(1) 032522 13# :
(1) 032522 005215 INC (R5)
```

```
1887
(1) ;*****
;TEST:213 125252 125252 SHIFTED BY -(3) = 177525 52525 PS = 10
(1) ;*****
```

```
(1) 032524 TST213:
(1) 032524 012700 125252 MOV @125252,#0 ;LOAD R0 WITH 125252
(1) 032530 012701 125252 MOV @125252,#0!1 ;LOAD R0!1 WITH 125252
(1) 032534 000241 CLC
(1) 032536 073043 ASHC -(3),#0 ;SHIFT R0,R0!1 BY -(3)
(1) 032540 106737 025626 MFPS @PASSWORD ;SAVE PS
(1) 032544 122737 000010 025626 CMPB @10,@PASSWORD ;IS THE PS 10?
(2) 032552 001401 BEQ 11#
(3) 032554 104000 EMT ;THE PS IS NOT EQUAL TO 10
(1) 032556 022700 177525 11# : CMP @177525,#0 ;IS THE RESULT 177525?
(2) 032562 001401 BEQ 12#
(3) 032564 104000 EMT ;RO IS NOT EQUAL TO 177525
(1) 032566 022701 052525 12# : CMP @52525,#0!1 ;IS THE RESULT 52525?
(2) 032572 001401 BEQ 13#
(3) 032574 104000 EMT ;RO!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
(1) 032576 13# :
(1) 032576 005215 INC (R5)
```

```
1888
(1) ;*****
;TEST:214 125252 125252 SHIFTED BY 2(4) = 177252 125252 PS = 11
(1) ;*****
```

```
(1) 032600 TST214:
(1) 032600 012700 125252 MOV @125252,#0 ;LOAD R0 WITH 125252
(1) 032604 012701 125252 MOV @125252,#0!1 ;LOAD R0!1 WITH 125252
(1) 032610 000241 CLC
(1) 032612 073064 000002 ASHC 2(4),#0 ;SHIFT R0,R0!1 BY 2(4)
(1) 032616 106737 025626 MFPS @PASSWORD ;SAVE PS
(1) 032622 122737 000011 025626 CMPB @11,@PASSWORD ;IS THE PS 11?
(2) 032630 001401 BEQ 11#
(3) 032632 104000 EMT ;THE PS IS NOT EQUAL TO 11
(1) 032634 022700 177252 11# : CMP @177252,#0 ;IS THE RESULT 177252?
(2) 032640 001401 BEQ 12#
(3) 032642 104000 EMT ;RO IS NOT EQUAL TO 177252
(1) 032644 022701 125252 12# : CMP @125252,#0!1 ;IS THE RESULT 125252?
(2) 032650 001401 BEQ 13#
(3) 032652 104000 EMT ;RO!1 IS NOT EQUAL TO 125252 OR INCORRECT SEQUENCE
```

R17

```

: 032654
: 032654 005215
(1)
11889
(1)
(1)
(1)
(1) 032656
(1) 032656 012700 125252
(1) 032662 012701 125252
(1) 032664 000241
(1) 032670 073074 000000
(1) 032674 106737 025626
(1) 032700 122737 000010 025626
(2) 032706 001401
(3) 032710 104000
(1) 032712 022760 177525
(2) 032716 001401
(3) 032720 104000
(1) 032722 022701 052525
(2) 032726 001401
(3) 032730 104000
(1) 032732
(1) 032732 005215
(1)
(1)

```

```

138: INC (R5)

;.....
;TEST:215 125252 125252 SHIFTED BY 8(4) = 177525 52525 PS = 10
;.....

TST215:
MOV #125252,#0 ;LOAD R0 WITH 125252
MOV #125252,#0!1 ;LOAD R0!1 WITH 125252
CLC
ASMC 8(4),#0 ;SHIFT R0,R0!1 BY 8(4)
MFPS @PPSWORD ;SAVE PS
CMPB #10,@PPSWORD ;IS THE PS 10?
BEQ 118
EMT ;THE PS IS NOT EQUAL TO 10
118: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 128
EMT ;R0 IS NOT EQUAL TO 177525
128: CMP #52525,#0!1 ;IS THE RESULT 52525?
BEQ 138
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
138: INC (R5)

```

```

11890
(1)
(1)
(1)
(1) 032734
(1) 032734 012700 125252
(1) 032740 012701 125252
(1) 032744 000241
(1) 032746 073034
(1) 032750 106737 025626
(1) 032754 122737 000010 025626
(2) 032762 001401
(3) 032764 104000
(1) 032766 022700 177525
(2) 032772 001401
(3) 032774 104000
(1) 032776 022701 052525
(2) 033002 001401
(3) 033004 104000
(1) 033006
(1) 033006 005215
(1)
(1)

```

```

;.....
;TEST:216 125252 125252 SHIFTED BY 8(4) = 177525 52525 PS = 10
;.....

TST216:
MOV #125252,#0 ;LOAD R0 WITH 125252
MOV #125252,#0!1 ;LOAD R0!1 WITH 125252
CLC
ASMC 8(4),#0 ;SHIFT R0,R0!1 BY 8(4)
MFPS @PPSWORD ;SAVE PS
CMPB #10,@PPSWORD ;IS THE PS 10?
BEQ 118
EMT ;THE PS IS NOT EQUAL TO 10
118: CMP #177525,#0 ;IS THE RESULT 177525?
BEQ 128
EMT ;R0 IS NOT EQUAL TO 177525
128: CMP #52525,#0!1 ;IS THE RESULT 52525?
BEQ 138
EMT ;R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
138: INC (R5)

```

```

11891
(1)
(1)
(1)
(1) 033010
(1) 033010 012700 125252

```

```

;.....
;TEST:217 125252 125252 SHIFTED BY 8(4) = 177525 52525 PS = 10
;.....

TST217:
MOV #125252,#0 ;LOAD R0 WITH 125252

```


(1)	033014	012701	125252		MOV	#125252,#0!1	!LOAD R0!1 WITH 125252
(1)	033020	000241			CLC		
(1)	033022	073054			ASMC	B-(4),#0	!SHIFT #0,R0!1 BY 2(4)
(1)	033024	106737	025626		MFPS	#PPSWORD	!SAVE PS
(1)	033030	122737	000010	025626	CMPB	#10,#PPSWORD	!IS THE PS 10?
(2)	033036	001401			BEQ	11#	
(3)	033040	104000			EMT		!THE PS IS NOT EQUAL TO 10
(1)	033042	022700	177525	11#:	CMP	#177525,#0	!IS THE RESULT 177525?
(2)	033046	001401			BEQ	12#	
(3)	033050	104000			EMT		!RO IS NOT EQUAL TO 177525
(1)	033052	022701	052525	12#:	CMP	#52525,#0!1	!IS THE RESULT 52525?
(2)	033056	001401			BEQ	13#	
(3)	033060	104000			EMT		!R0!1 IS NOT EQUAL TO 52525 OR INCORRECT SEQUENCE
(1)	033062			13#:			
(1)	033062	005215			INC	(R5)	
(1)							
(1)							
11892							
11893							
11894							
11895							
11896							
11897							
11898							

11981
11982
11983
11984
11985
11986
11987
11988
(1)
(1)
(1)
(1) 033064
(1) 033064 012700 000001
(1) 033070 070027 000000
(1) 033074 106737 025626
(1) 033100 122737 000004 025626
(2) 033106 001401
(3) 033110 104000
(1) 033112 022700 000000
(2) 033116 001401
(3) 033120 104000
(1) 033122 022701 000000
(2) 033126 001401
(3) 033130 104000
(1) 033132
(1) 033132 005215
(1)
(1)

```
*****  
; MUL INSTRUCTION TESTS  
*****  
  
;*****  
;TEST:220 MUL 1 * #0 = 0 0 PS = 4  
;*****  
  
TST220:  
MOV #1,#0 ;LOAD MULTIPLICAND WITH 1  
MUL #0,#0 ;MULTIPLY 1 * #0  
MFPS @#PSWORD ;SAVE PS  
CMPB #4,@#PSWORD ;IS PS = 4  
BEQ 11#  
EHT ;PS IS WRONG  
11# : CMP #0,#0 ;IS HIGH ORDER = 0  
BEQ 12#  
EHT ;HIGH ORDER IS WRONG  
12# : CMP #0,#0!1 ;IS LOW ORDER = 0  
BEQ 13#  
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE  
13# : INC (R5)
```

11989
(1)
(1)
(1)
(1) 033134
(1) 033134 012700 177777
(1) 033140 070027 000001
(1) 033144 106737 025626
(1) 033150 122737 000010 025626
(2) 033156 001401
(3) 033160 104000
(1) 033162 022700 177777
(2) 033166 001401
(3) 033170 104000
(1) 033172 022701 177777
(2) 033176 001401
(3) 033200 104000
(1) 033202
(1) 033202 005215
(1)
(1)

```
;*****  
;TEST:221 MUL -1 * #1 = -1 -1 PS = 10  
;*****  
  
TST221:  
MOV #-1,#0 ;LOAD MULTIPLICAND WITH -1  
MUL #1,#0 ;MULTIPLY -1 * #1  
MFPS @#PSWORD ;SAVE PS  
CMPB #10,@#PSWORD ;IS PS = 10  
BEQ 11#  
EHT ;PS IS WRONG  
11# : CMP #-1,#0 ;IS HIGH ORDER = -1  
BEQ 12#  
EHT ;HIGH ORDER IS WRONG  
12# : CMP #-1,#0!1 ;IS LOW ORDER = -1  
BEQ 13#  
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE  
13# : INC (R5)
```

11990
(1)
(1)
(1)
(1) 033204
(1) 033204 012702 000002
(1) 033210 070227 000002

```
;*****  
;TEST:222 MUL 2 * #2 = 0 4 PS = 0  
;*****  
  
TST222:  
MOV #2,#2 ;LOAD MULTIPLICAND WITH 2  
MUL #2,#2 ;MULTIPLY 2 * #2
```

```

(1) 033214 106737 025626      MFPS      @@PSWORD      ;SAVE PS
(1) 033220 122737 000000 025626  CMPB      @0,@@PSWORD ;IS PS = 0
(2) 033226 001401          BEQ      11$          ;PS IS WRONG
(3) 033230 104000          EMT
(1) 033232 022702 000000 11$:    CMP      @0,@2      ;IS HIGH ORDER = 0
(2) 033236 001401          BEQ      12$          ;HIGH ORDER IS WRONG
(3) 033240 104000          EMT
(1) 033242 022703 000004 12$:    CMP      @4,@2!1    ;IS LOW ORDER = 4
(2) 033246 001401          BEQ      13$          ;LOW ORDER IS WRONG OR WRONG SEQUENCE
(3) 033250 104000          EMT
(1) 033252          13$:    INC      (R5)
(1)
(1)
11991
(1)
(1)
(1)
(1) 033254          TST223:
(1) 033254 010501          MOV      R5,R1      ;SAVE R5
(1) 033256 012704 001000          MOV      @1000,@4   ;LOAD MULTIPLICAND WITH 1000
(1) 033262 070427 000200          MUL      @200,@4    ;MULTIPLY 1000 * @200
(1) 033266 106737 025626      MFPS      @@PSWORD   ;SAVE PS
(1) 033272 122737 000001 025626  CMPB      @1,@@PSWORD ;IS PS = 1
(2) 033300 001401          BEQ      11$          ;PS IS WRONG
(3) 033302 104000          EMT
(1) 033304 022704 000001 11$:    CMP      @1,@4      ;IS HIGH ORDER = 1
(2) 033310 001401          BEQ      12$          ;HIGH ORDER IS WRONG
(3) 033312 104000          EMT
(1) 033314 022705 000000 12$:    CMP      @0,@4!1    ;IS LOW ORDER = 0
(2) 033320 001401          BEQ      13$          ;LOW ORDER IS WRONG OR WRONG SEQUENCE
(3) 033322 104000          EMT
(1) 033324          13$:    MOV      R1,R5      ;RESTORE R5
(1) 033324 010105          INC      (R5)
(1) 033326 005215
(1)
(1)
11992
(1)
(1)
(1)
(1) 033330          TST224:
(1) 033330 012700 000002          MOV      @2,@0      ;LOAD MULTIPLICAND WITH 2
(1) 033334 070027 077777          MUL      @77777,@0  ;MULTIPLY 2 * @77777
(1) 033340 106737 025626      MFPS      @@PSWORD   ;SAVE PS
(1) 033344 122737 000001 025626  CMPB      @1,@@PSWORD ;IS PS = 1
(2) 033352 001401          BEQ      11$          ;PS IS WRONG
(3) 033354 104000          EMT
(1) 033356 022700 000000 11$:    CMP      @0,@0      ;IS HIGH ORDER = 0
(2) 033362 001401          BEQ      12$          ;HIGH ORDER IS WRONG
(3) 033364 104000          EMT
(1) 033366 022701 177776 12$:    CMP      @177776,@0!1 ;IS LOW ORDER = 177776
(2) 033372 001401          BEQ      13$          ;LOW ORDER IS WRONG OR WRONG SEQUENCE
(3) 033374 104000          EMT
(1) 033376          13$:    INC      (R5)
(1) 033376 005215

```

```
(1)
(1)
11993
(1)
(1)
(1)
(1)
(1) 033400
(1) 033400 012702 007777
(1) 033404 070227 000010
(1) 033410 106737 025626
(1) 033414 122737 000000 025626
(2) 033422 001401
(3) 033424 104000
(1) 033426 022702 000000 11:
(2) 033432 001401
(3) 033434 104000
(1) 033436 022703 077770 12:
(2) 033442 001401
(3) 033444 104000
(1) 033446
(1) 033446 005215
(1)
(1)
11994
(1)
(1)
(1)
(1) 033450
(1) 033450 010501
(1) 033452 012704 077777
(1) 033456 070427 077777
(1) 033462 106737 025626
(1) 033466 122737 000001 025626
(2) 033474 001401
(3) 033476 104000
(1) 033500 022704 037777 11:
(2) 033504 001401
(3) 033506 104000
(1) 033510 022705 000001 12:
(2) 033514 001401
(3) 033516 104000
(1) 033520
(1) 033520 010105
(1) 033522 005215
(1)
(1)
11995
(1)
(1)
(1)
(1) 033524
(1) 033524 012702 177777
(1) 033530 070227 077777
(1) 033534 106737 025626
(1) 033540 122737 000010 025626
(2) 033546 001401
```

```

;*****
;TEST:225      MUL      7777 * #10 = 0 77770      PS = 0
;*****

TST225:
MOV      #7777,#2      ;LOAD MULTIPLICAND WITH 7777
MUL      #10,#2        ;MULTIPLY 7777 * #10
MFPS     @#PSWORD      ;SAVE PS
CMPB     #0,@#PSWORD   ;IS PS = 0
BEQ      11:
EMT
;PS IS WRONG
11:      CMP      #0,#2      ;IS HIGH ORDER = 0
BEQ      12:
EMT
;HIGH ORDER IS WRONG
12:      CMP      #77770,#2!1 ;IS LOW ORDER = 77770
BEQ      13:
EMT
;LOW ORDER IS WRONG OR WRONG SEQUENCE
13:      INC      (R5)

;*****
;TEST:226      MUL      77777 * #77777 = 37777 1      PS = 1
;*****

TST226:
MOV      R5,R1         ;SAVE R5
MOV      #77777,#4     ;LOAD MULTIPLICAND WITH 77777
MUL      #77777,#4     ;MULTIPLY 77777 * #77777
MFPS     @#PSWORD      ;SAVE PS
CMPB     #1,@#PSWORD   ;IS PS = 1
BEQ      11:
EMT
;PS IS WRONG
11:      CMP      #37777,#4   ;IS HIGH ORDER = 37777
BEQ      12:
EMT
;HIGH ORDER IS WRONG
12:      CMP      #1,#4!1    ;IS LOW ORDER = 1
BEQ      13:
EMT
;LOW ORDER IS WRONG OR WRONG SEQUENCE
13:      MOV      R1,R5     ;RESTORE R5
INC      (R5)

;*****
;TEST:227      MUL      -1 * #77777 = -1 100001      PS = 10
;*****

TST227:
MOV      #-1,#2        ;LOAD MULTIPLICAND WITH 1
MUL      #77777,#2     ;MULTIPLY -1 * #77777
MFPS     @#PSWORD      ;SAVE PS
CMPB     #10,@#PSWORD  ;IS PS = 10
BEQ      11:
```

(3) 033550 104000
(1) 033552 022702 177777
(2) 033556 001401
(3) 033560 104000
(1) 033562 022703 100001
(2) 033566 001401
(3) 033570 104000
(1) 033572
(1) 033572 005215
(1)
(1)

114: EMT ;PS IS WRONG
CMP # 1,#2 ;IS HIGH ORDER = 1
BEQ 124
124: EMT ;HIGH ORDER IS WRONG
CMP #100001,#2!1 ;IS LOW ORDER = 100001
BEQ 134
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
134: INC (R5)

11996

(1)
(1)
(1)
(1) 033574
(1) 033574 012700 177776
(1) 033600 070027 077777
(1) 033604 106737 025626
(1) 033610 122737 000011 025626
(2) 033616 001401
(3) 033620 104000
(1) 033622 022700 177777
(2) 033626 001401
(3) 033630 104000
(1) 033632 022701 000002
(2) 033636 001401
(3) 033640 104000
(1) 033642
(1) 033642 005215
(1)
(1)

;TEST:230 MUL -2 * #77777 = -1 2 PS = 11

TST230:
MOV #-2,#0 ;LOAD MULTIPLICAND WITH 2
MUL #77777,#0 ;MULTIPLY -2 * #77777
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 114
EMT ;PS IS WRONG
114: CMP #-1,#0 ;IS HIGH ORDER = 1
BEQ 124
EMT ;HIGH ORDER IS WRONG
124: CMP #2,#0!1 ;IS LOW ORDER = 2
BEQ 134
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
134: INC (R5)

11997

(1)
(1)
(1)
(1) 033644
(1) 033644 012702 125252
(1) 033650 070227 000002
(1) 033654 106737 025626
(1) 033660 122737 000011 025626
(2) 033666 001401
(3) 033670 104000
(1) 033672 022702 177777
(2) 033676 001401
(3) 033700 104000
(1) 033702 022703 052524
(2) 033706 001401
(3) 033710 104000
(1) 033712
(1) 033712 005215
(1)
(1)

;TEST:231 MUL 125252 * #2 = -1 52524 PS = 11

TST231:
MOV #125252,#2 ;LOAD MULTIPLICAND WITH 125252
MUL #2,#2 ;MULTIPLY 125252 * #2
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 114
EMT ;PS IS WRONG
114: CMP #-1,#2 ;IS HIGH ORDER = -1
BEQ 124
EMT ;HIGH ORDER IS WRONG
124: CMP #52524,#2!1 ;IS LOW ORDER = 52524
BEQ 134
EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
134: INC (R5)

11998

(1)
(1)

;TEST:232 MUL 125252 * #40000 = 165252 100000 PS = 11

```
(1)
(1) 033714
(1) 033714 010501
(1) 033716 012704 125252
(1) 033722 070427 040000
(1) 033726 106737 025626
(1) 033732 122737 000011 025626
(2) 033740 001401
(3) 033742 104000
(1) 033744 022704 165252 11#:
(2) 033750 001401
(3) 033752 104000
(1) 033754 022705 100000 12#:
(2) 033760 001401
(3) 033762 104000
(1) 033764 13#:
(1) 033764 010105
(1) 033766 005215
(1)
(1)
11999
(1)
(1)
(1)
(1) 033770
(1) 033770 012700 107070
(1) 033774 070027 107070
(1) 034000 106737 025626
(1) 034004 122737 000001 025626
(2) 034012 001401
(3) 034014 104000
(1) 034016 022700 031222 11#:
(2) 034022 001401
(3) 034024 104000
(1) 034026 022701 026100 12#:
(2) 034032 001401
(3) 034034 104000
(1) 034036 13#:
(1) 034036 005215
(1)
(1)
12000
(1)
(1)
(1)
(1) 034040
(1) 034040 012701 177777
(1) 034044 070127 000001
(1) 034050 106737 025626
(1) 034054 122737 000010 025626
(2) 034062 001401
(3) 034064 104000
(1) 034066 022701 177777 11#:
(2) 034072 001401
(3) 034074 104000
(1) 034076 022701 177777 12#:

TST232:
MOV R5,R1 ;SAVE R5
MOV #125252,#4 ;LOAD MULTIPLICAND WITH 125252
MUL #40000,#4 ;MULTIPLY 125252 * #40000
MFPS @#PSWORD ;SAVE PS
CMPB #11,@#PSWORD ;IS PS = 11
BEQ 11#
EHT ;PS IS WRONG
11#: CMP #165252,#4 ;IS HIGH ORDER = 165252
BEQ 12#
EHT ;HIGH ORDER IS WRONG
12#: CMP #100000,#4!1 ;IS LOW ORDER = 100000
BEQ 13#
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13#:
MOV R1,R5 ;RESTORE R5
INC (R5)

;*****
;TEST:233 MUL 107070 * #107070 = 31222 26100 PS = 1
;*****

TST233:
MOV #107070,#0 ;LOAD MULTIPLICAND WITH 107070
MUL #107070,#0 ;MULTIPLY 107070 * #107070
MFPS @#PSWORD ;SAVE PS
CMPB #1,@#PSWORD ;IS PS = 1
BEQ 11#
EHT ;PS IS WRONG
11#: CMP #31222,#0 ;IS HIGH ORDER = 31222
BEQ 12#
EHT ;HIGH ORDER IS WRONG
12#: CMP #26100,#0!1 ;IS LOW ORDER = 26100
BEQ 13#
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13#:
INC (R5)

;*****
;TEST:234 MUL -1 * #1 = -1 -1 PS = 10
;*****

TST234:
MOV #-1,#1 ;LOAD MULTIPLICAND WITH -1
MUL #1,#1 ;MULTIPLY -1 * #1
MFPS @#PSWORD ;SAVE PS
CMPB #10,@#PSWORD ;IS PS = 10
BEQ 11#
EHT ;PS IS WRONG
11#: CMP #-1,#1 ;IS HIGH ORDER = -1
BEQ 12#
EHT ;HIGH ORDER IS WRONG
12#: CMP #-1,#1!1 ;IS LOW ORDER = -1
```

(2) 034102 001401
 (3) 034104 104000
 (1) 034106
 (1) 034106 005215
 (1)
 (1)
 12001
 (1)
 (1)
 (1)
 (1) 034110
 (1) 034110 012703 177777
 (1) 034114 070327 000000
 (1) 034120 106737 025626
 (1) 034124 122737 000004 025626
 (2) 034132 001401
 (3) 034134 104000
 (1) 034136 022703 000000
 (2) 034142 001401
 (3) 034144 104000
 (1) 034146 022703 000000
 (2) 034152 001401
 (3) 034154 104000
 (1) 034156
 (1) 034156 005215
 (1)
 (1)

```

      BEQ      13$
      EMT
13$:
      INC      (R5)

;*****
;TEST:235      MUL      -1 * #0 = 0 0      PS = 4
;*****

TST235:
      MOV      #-1,#3      ;LOAD MULTIPLICAND WITH 1
      MUL      #0,#3      ;MULTIPLY -1 * #0
      MFPS     @#PSWORD    ;SAVE PS
      CMPB    #4,@#PSWORD ;IS PS = 4
      BEQ      11$
      EMT
11$:      CMP      #0,#3      ;PS IS WRONG
      BEQ      12$      ;IS HIGH ORDER = 0
      EMT
12$:      CMP      #0,#3!1    ;HIGH ORDER IS WRONG
      BEQ      13$      ;IS LOW ORDER = 0
      EMT
13$:      INC      (R5)      ;LOW ORDER IS WRONG OR WRONG SEQUENCE

```

12002
 (1)
 (1)
 (1)
 (1) 034160
 (1) 034160 010501
 (1) 034162 012705 077777
 (1) 034166 070527 100000
 (1) 034172 106737 025626
 (1) 034176 122737 000011 025626
 (2) 034204 001401
 (3) 034206 104000
 (1) 034210 022705 100000
 (2) 034214 001401
 (3) 034216 104000
 (1) 034220 022705 100000
 (2) 034224 001401
 (3) 034226 104000
 (1) 034230
 (1) 034230 010105
 (1) 034232 005215
 (1)
 (1)

```

;*****
;TEST:236      MUL      77777 * #100000 = 100000 100000      PS = 11
;*****

TST236:
      MOV      R5,R1      ;SAVE R5
      MOV      #77777,#5  ;LOAD MULTIPLICAND WITH 77777
      MUL      #100000,#5 ;MULTIPLY 77777 * #100000
      MFPS     @#PSWORD    ;SAVE PS
      CMPB    #11,@#PSWORD ;IS PS = 11
      BEQ      11$
      EMT
11$:      CMP      #100000,#5 ;PS IS WRONG
      BEQ      12$      ;IS HIGH ORDER = 100000
      EMT
12$:      CMP      #100000,#5!1 ;HIGH ORDER IS WRONG
      BEQ      13$      ;IS LOW ORDER = 100000
      EMT
13$:      MOV      R1,R5      ;LOW ORDER IS WRONG OR WRONG SEQUENCE
      INC      (R5)      ;RESTORE R5

```

12003
 (1)
 (1)
 (1)
 (1) 034234
 (1) 034234 012701 177777

```

;*****
;TEST:237      MUL      -1 * #77777 = 100001 100001      PS = 10
;*****

TST237:
      MOV      #-1,#1      ;LOAD MULTIPLICAND WITH -1

```

(1) 034240 070127 077777
 (1) 034244 106737 025626
 (1) 034250 122737 000010 025626
 (2) 034256 001401
 (3) 034260 104000
 (1) 034262 022701 100001 11:
 (2) 034266 001401
 (3) 034270 104000
 (1) 034272 022701 100001 12:
 (2) 034276 001401
 (3) 034300 104000
 (1) 034302 13:
 (1) 034302 005215
 (1)
 (1)

MUL #77777,#1 ;MULTIPLY 1 * #77777
 MFPS @#PSWORD ;SAVE PS
 CMPB #10,@#PSWORD ;IS PS = 10
 BEQ 11: ;PS IS WRONG
 EMT ;IS HIGH ORDER = 100001
 CMP #100001,#1
 BEQ 12: ;HIGH ORDER IS WRONG
 EMT ;IS LOW ORDER = 100001
 CMP #100001,#1!1
 BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 EMT
 INC (R5)

12004

(1)
 (1)
 (1)
 (1) 034304
 (1) 034304 012703 077777
 (1) 034310 070327 077777
 (1) 034314 106737 025626
 (1) 034320 122737 000001 025626
 (2) 034326 001401
 (3) 034330 104000
 (1) 034332 022703 000001 11:
 (2) 034336 001401
 (3) 034340 104000
 (1) 034342 022703 000001 12:
 (2) 034346 001401
 (3) 034350 104000
 (1) 034352 13:
 (1) 034352 005215
 (1)
 (1)

 ;TEST:240 MUL 77777 * #77777 = 1 1 PS = 1

 TST240:
 MOV #77777,#3 ;LOAD MULTIPLICAND WITH 77777
 MUL #77777,#3 ;MULTIPLY 77777 * #77777
 MFPS @#PSWORD ;SAVE PS
 CMPB #1,@#PSWORD ;IS PS = 1
 BEQ 11: ;PS IS WRONG
 EMT ;IS HIGH ORDER = 1
 CMP #1,#3
 BEQ 12: ;HIGH ORDER IS WRONG
 EMT ;IS LOW ORDER = 1
 CMP #1,#3!1
 BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 EMT
 INC (R5)

12005

(1)
 (1)
 (1)
 (1) 034354
 (1) 034354 010501
 (1) 034356 012705 000002
 (1) 034362 070527 000002
 (1) 034366 106737 025626
 (1) 034372 122737 000000 025626
 (2) 034400 001401
 (3) 034402 104000
 (1) 034404 022705 000004 11:
 (2) 034410 001401
 (3) 034412 104000
 (1) 034414 022705 000004 12:
 (2) 034420 001401
 (3) 034422 104000
 (1) 034424 13:
 (1) 034424 010105

 ;TEST:241 MUL 2 * #2 = 4 4 PS = 0

 TST241:
 MOV R5,R1 ;SAVE R5
 MOV #2,#5 ;LOAD MULTIPLICAND WITH 2
 MUL #2,#5 ;MULTIPLY 2 * #2
 MFPS @#PSWORD ;SAVE PS
 CMPB #0,@#PSWORD ;IS PS = 0
 BEQ 11: ;PS IS WRONG
 EMT ;IS HIGH ORDER = 4
 CMP #4,#5
 BEQ 12: ;HIGH ORDER IS WRONG
 EMT ;IS LOW ORDER = 4
 CMP #4,#5!1
 BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 EMT
 MOV R1,R5 ;RESTORE R5


```

(1) 034426 005215
(1)
(1)
12006 034430 012702 040000
12007 034434 012703 025654
12008 034440 012704 025656
12009
12010
(1)
(1)
(1)
(1) 034444
(1) 034444 012700 125252
(1) 034450 070067 171200
(1) 034454 106737 025626
(1) 034460 122737 000011 025626
(2) 034466 001401
(3) 034470 104000
(1) 034472 022700 165252
(2) 034476 001401
(3) 034500 104000
(1) 034502 022701 100000
(2) 034506 001401
(3) 034510 104000
(1) 034512
(1) 034512 005215
(1)
(1)

```

```

INC (R5)
MOV #40000,#2
MOV #S5,#3
MOV #S6,#4
;*****
;TEST:242 MUL 125252 * S5 = 165252 100000 PS = 11
;*****
TST242:
MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
MUL S5,#0 ;MULTIPLY 125252 * S5
MFPS @PSWORD ;SAVE PS
CMPB #11,@PSWORD ;IS PS = 11
BEQ 11$
EHT ;PS IS WRONG
11$: CMP #165252,#0 ;IS HIGH ORDER = 165252
BEQ 12$
EHT ;HIGH ORDER IS WRONG
12$: CMP #100000,#0!1 ;IS LOW ORDER = 100000
BEQ 13$
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13$: INC (R5)

```

```

12011
(1)
(1)
(1)
(1) 034514
(1) 034514 012700 125252
(1) 034520 070077 171132
(1) 034524 106737 025626
(1) 034530 122737 000011 025626
(2) 034536 001401
(3) 034540 104000
(1) 034542 022700 165252
(2) 034546 001401
(3) 034550 104000
(1) 034552 022701 100000
(2) 034556 001401
(3) 034560 104000
(1) 034562
(1) 034562 005215
(1)
(1)

```

```

;*****
;TEST:243 MUL 125252 * S6 = 165252 100000 PS = 11
;*****
TST243:
MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
MUL S6,#0 ;MULTIPLY 125252 * S6
MFPS @PSWORD ;SAVE PS
CMPB #11,@PSWORD ;IS PS = 11
BEQ 11$
EHT ;PS IS WRONG
11$: CMP #165252,#0 ;IS HIGH ORDER = 165252
BEQ 12$
EHT ;HIGH ORDER IS WRONG
12$: CMP #100000,#0!1 ;IS LOW ORDER = 100000
BEQ 13$
EHT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
13$: INC (R5)

```

```

12012
(1)
(1)
(1)
(1) 034564
(1) 034564 012700 125252
(1) 034570 070037 025654

```

```

;*****
;TEST:244 MUL 125252 * S5 = 165252 100000 PS = 11
;*****
TST244:
MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
MUL @S5,#0 ;MULTIPLY 125252 * S5

```

(1) 034574 106737 025626
 (1) 034600 122737 000011 025626
 (2) 034606 001401
 (3) 034610 104000
 (1) 034612 022700 165252
 (2) 034616 001401
 (3) 034620 104000
 (1) 034622 022701 100000
 (2) 034626 001401
 (3) 034630 104000
 (1) 034632
 (1) 034632 005215
 (1)
 (1)

MFPS B@PSWORD ;SAVE PS
 CMPB #11,B@PSWORD ;IS PS = 11
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #165252,#0 ;IS HIGH ORDER = 165252
 BEQ 12#
 EMT ;HIGH ORDER IS WRONG
 12#: CMP #100000,#0!1 ;IS LOW ORDER = 100000
 BEQ 13#
 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 13#: INC (R5)

12013

(1)
 (1)
 (1)
 (1) 034634
 (1) 034634 012700 125252
 (1) 034640 070002
 (1) 034642 106737 025626
 (1) 034646 122737 000011 025626
 (2) 034654 001401
 (3) 034656 104000
 (1) 034660 022700 165252
 (2) 034664 001401
 (3) 034666 104000
 (1) 034670 022701 100000
 (2) 034674 001401
 (3) 034676 104000
 (1) 034700
 (1) 034700 005215
 (1)
 (1)

 ;TEST:245 MUL 125252 * #2 = 165252 100000 PS = 11

 TST245:
 MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
 MUL #2,#0 ;MULTIPLY 125252 * #2
 MFPS B@PSWORD ;SAVE PS
 CMPB #11,B@PSWORD ;IS PS = 11
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #165252,#0 ;IS HIGH ORDER = 165252
 BEQ 12#
 EMT ;HIGH ORDER IS WRONG
 12#: CMP #100000,#0!1 ;IS LOW ORDER = 100000
 BEQ 13#
 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 13#: INC (R5)

12014

(1)
 (1)
 (1)
 (1) 034702
 (1) 034702 012700 125252
 (1) 034706 070023
 (1) 034710 106737 025626
 (1) 034714 122737 000011 025626
 (2) 034722 001401
 (3) 034724 104000
 (1) 034726 022700 165252
 (2) 034732 001401
 (3) 034734 104000
 (1) 034736 022701 100000
 (2) 034742 001401
 (3) 034744 104000
 (1) 034746
 (1) 034746 005215
 (1)
 (1)

 ;TEST:246 MUL 125252 * (3) = 165252 100000 PS = 11

 TST246:
 MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
 MUL (3),#0 ;MULTIPLY 125252 * (3)
 MFPS B@PSWORD ;SAVE PS
 CMPB #11,B@PSWORD ;IS PS = 11
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #165252,#0 ;IS HIGH ORDER = 165252
 BEQ 12#
 EMT ;HIGH ORDER IS WRONG
 12#: CMP #100000,#0!1 ;IS LOW ORDER = 100000
 BEQ 13#
 EMT ;LOW ORDER IS WRONG OR WRONG SEQUENCE
 13#: INC (R5)

12015

```

(1)
(1)
(1)
(1) 034750
(1) 034750 012700 125252
(1) 034754 070043
(1) 034756 106737 025626
(1) 034762 122737 000011 025626
(2) 034770 001401
(3) 034772 104000
(1) 034774 022700 165252
(2) 035000 001401
(3) 035002 104000
(1) 035004 022701 100000
(2) 035010 001401
(3) 035012 104000
(1) 035014
(1) 035014 005215
(1)
(1)

```

```

;*****
;TEST:247      MUL      125252 * -(3) = 165252 100000      PS = 11
;*****

TST247:
      MOV      #125252,#0      ;LOAD MULTIPLICAND WITH 125252
      MUL      -(3),#0      ;MULTIPLY 125252 * -(3)
      MFPS     @#PSWORD      ;SAVE PS
      CMPB    #11,@#PSWORD    ;IS PS = 11
      BEQ     11#
      EMT
      11#:    CMP      #165252,#0      ;PS IS WRONG
      BEQ     12#      ;IS HIGH ORDER = 165252
      EMT
      12#:    CMP      #100000,#0!1    ;HIGH ORDER IS WRONG
      BEQ     13#      ;IS LOW ORDER = 100000
      EMT
      13#:    EMT
      INC     (R5)      ;LOW ORDER IS WRONG OR WRONG SEQUENCE

```

12016

```

(1)
(1)
(1)
(1) 035016
(1) 035016 012700 125252
(1) 035022 070064 000002
(1) 035026 106737 025626
(1) 035032 122737 000011 025626
(2) 035040 001401
(3) 035042 104000
(1) 035044 022700 165252
(2) 035050 001401
(3) 035052 104000
(1) 035054 022701 100000
(2) 035060 001401
(3) 035062 104000
(1) 035064
(1) 035064 005215
(1)
(1)

```

```

;*****
;TEST:250      MUL      125252 * 2(4) = 165252 100000      PS = 11
;*****

TST250:
      MOV      #125252,#0      ;LOAD MULTIPLICAND WITH 125252
      MUL      2(4),#0      ;MULTIPLY 125252 * 2(4)
      MFPS     @#PSWORD      ;SAVE PS
      CMPB    #11,@#PSWORD    ;IS PS = 11
      BEQ     11#
      EMT
      11#:    CMP      #165252,#0      ;PS IS WRONG
      BEQ     12#      ;IS HIGH ORDER = 165252
      EMT
      12#:    CMP      #100000,#0!1    ;HIGH ORDER IS WRONG
      BEQ     13#      ;IS LOW ORDER = 100000
      EMT
      13#:    EMT
      INC     (R5)      ;LOW ORDER IS WRONG OR WRONG SEQUENCE

```

12017

```

(1)
(1)
(1)
(1) 035066
(1) 035066 012700 125252
(1) 035072 070074 000000
(1) 035076 106737 025626
(1) 035102 122737 000011 025626
(2) 035110 001401
(3) 035112 104000
(1) 035114 022700 165252
(2) 035120 001401
(3) 035122 104000

```

```

;*****
;TEST:251      MUL      125252 * 8(4) = 165252 100000      PS = 11
;*****

TST251:
      MOV      #125252,#0      ;LOAD MULTIPLICAND WITH 125252
      MUL      8(4),#0      ;MULTIPLY 125252 * 8(4)
      MFPS     @#PSWORD      ;SAVE PS
      CMPB    #11,@#PSWORD    ;IS PS = 11
      BEQ     11#
      EMT
      11#:    CMP      #165252,#0      ;PS IS WRONG
      BEQ     12#      ;IS HIGH ORDER = 165252
      EMT
      EMT
      EMT      ;HIGH ORDER IS WRONG

```

(1) 035124 022701 100000
(2) 035130 001401
(3) 035132 104000
(1) 035134
(1) 035134 005215
(1)
(1)

12: CMP #100000,#0!1 ;IS LOW ORDER = 100000
BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
EMT
13: INC (R5)

2018

(1)
(1)
(1)

;TEST:252 MUL 125252 * 8(4)+ = 165252 100000 PS = 11

(1) 035136
(1) 035136 012700 125252
(1) 035142 070034
(1) 035144 106737 025626
(1) 035150 122737 000011 025626
(2) 035156 001401
(3) 035160 104000
(1) 035162 022700 165252
(2) 035166 001401
(3) 035170 104000
(1) 035172 022701 100000
(2) 035176 001401
(3) 035200 104000
(1) 035202
(1) 035202 005215
(1)
(1)

TST252:
MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
MUL 8(4)+,#0 ;MULTIPLY 125252 * 8(4)+
MFPS @PSWORD ;SAVE PS
CMPB #11,@PSWORD ;IS PS = 11
BEQ 11: ;PS IS WRONG
EMT
11: CMP #165252,#0 ;IS HIGH ORDER = 165252
BEQ 12: ;HIGH ORDER IS WRONG
EMT
12: CMP #100000,#0!1 ;IS LOW ORDER = 100000
BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
EMT
13: INC (R5)

2019

(1)
(1)
(1)

;TEST:253 MUL 125252 * 8-(4) = 165252 100000 PS = 11

(1) 035204
(1) 035204 012700 125252
(1) 035210 070054
(1) 035212 106737 025626
(1) 035216 122737 000011 025626
(2) 035224 001401
(3) 035226 104000
(1) 035230 022700 165252
(2) 035234 001401
(3) 035236 104000
(1) 035240 022701 100000
(2) 035244 001401
(3) 035246 104000
(1) 035250
(1) 035250 005215
(1)
(1)

TST253:
MOV #125252,#0 ;LOAD MULTIPLICAND WITH 125252
MUL 8-(4),#0 ;MULTIPLY 125252 * 8(4)
MFPS @PSWORD ;SAVE PS
CMPB #11,@PSWORD ;IS PS = 11
BEQ 11: ;PS IS WRONG
EMT
11: CMP #165252,#0 ;IS HIGH ORDER = 165252
BEQ 12: ;HIGH ORDER IS WRONG
EMT
12: CMP #100000,#0!1 ;IS LOW ORDER = 100000
BEQ 13: ;LOW ORDER IS WRONG OR WRONG SEQUENCE
EMT
13: INC (R5)

12024
12025
12026
12027
12028
12029
12030

.....
; DIV INSTRUCTION TESTS
.....

.....
;TEST:254 DIV 0 4 / #2 = 2 REM = 0 PS = 0
.....

TST254:
MOV #0,#0 ;LOAD HIGH ORDER WITH 0
MOV #4,#0+1 ;LOAD LOW ORDER WITH 4
DIV #2,#0 ;DIVIDE BY #2
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11#
EIT ;PS IS WRONG
11#: CMP #2,#0 ;IS QUOTIENT = 2
BEQ 12#
EIT ;QUOTIENT IS WRONG
12#: CMP #0,#0+1 ;IS REMAINDER = 0
BEQ 13#
EIT ;WRONG REMAINDER
13#: INC (R5)

.....
;TEST:255 DIV -1 -9. / #3 = -3 REM = 0 PS = 10
.....

TST255:
MOV #-1,#2 ;LOAD HIGH ORDER WITH 1
MOV #-9.,#2+1 ;LOAD LOW ORDER WITH -9.
DIV #3,#2 ;DIVIDE BY #3
MFPS @PSWORD ;SAVE PS
CMPB #10,@PSWORD ;IS PS = 10
BEQ 11#
EIT ;PS IS WRONG
11#: CMP #-3,#2 ;IS QUOTIENT = -3
BEQ 12#
EIT ;QUOTIENT IS WRONG
12#: CMP #0,#2+1 ;IS REMAINDER = 0
BEQ 13#
EIT ;WRONG REMAINDER
13#: INC (R5)

.....
;TEST:256 DIV 0 9. / #2 = 4 REM = 1 PS = 0
.....

TST256:
MOV R5,R1 ;SAVE R5
MOV #0,#4 ;LOAD HIGH ORDER WITH 0
MOV #9.,#4+1 ;LOAD LOW ORDER WITH 9.

12031
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
12032
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

035252
035252 012700 000000
035256 012701 000004
035262 071027 000002
035266 106737 025626
035272 122737 000000 025626
035300 001401
035302 104000
035304 022700 000002
035310 001401
035312 104000
035314 022701 000000
035320 001401
035322 104000
035324
035324 005215

035326
035326 012702 177777
035332 012703 177767
035336 071227 000003
035342 106737 025626
035346 122737 000010 025626
035354 001401
035356 104000
035360 022702 177775
035364 001401
035366 104000
035370 022703 000000
035374 001401
035376 104000
035400
035400 005215

035402
035402 010501
035404 012704 000000
035410 012705 000011

```
(1) 035414 071427 000002          DIV      #2,#4          ;DIVIDE BY #2
(1) 035420 106737 025626          MFPS     @PPSWORD      ;SAVE PS
(1) 035424 122737 000000 025626  CMPB     #0,@PPSWORD   ;IS PS = 0
(2) 035432 001401                    BEQ      11#
(3) 035434 104000                    EMT
(1) 035436 022704 000004          11# :  CMP      #4,#4          ;PS IS WRONG
(2) 035442 001401                    BEQ      12#          ;IS QUOTIENT = 4
(3) 035444 104000                    EMT
(1) 035446 022705 000001          12# :  CMP      #1,#4+1      ;QUOTIENT IS WRONG
(2) 035452 001401                    BEQ      13#          ;IS REMAINDER = 1
(3) 035454 104000                    EMT
(1) 035456                    13# :                    ;WRONG REMAINDER
(1) 035456 010105                    MOV      R1,R5        ;RESTORE R5
(1) 035460 005215                    INC      (R5)

;*****
;TEST:257      DIV      -1 -9. / #2 = -4      REM = 1      PS = 10
;*****

(1)
(1)
(1)
(1) 035462          TST257:
(1) 035462 012700 177777          MOV      #-1,#0        ;LOAD HIGH ORDER WITH -1
(1) 035466 012701 177767          MOV      #-9.,#0+1     ;LOAD LOW ORDER WITH -9.
(1) 035472 071027 000002          DIV      #2,#0         ;DIVIDE BY #2
(1) 035476 106737 025626          MFPS     @PPSWORD      ;SAVE PS
(1) 035502 122737 000010 025626  CMPB     #10,@PPSWORD  ;IS PS = 10
(2) 035510 001401                    BEQ      11#
(3) 035512 104000                    EMT
(1) 035514 022700 177774          11# :  CMP      #-4,#0         ;PS IS WRONG
(2) 035520 001401                    BEQ      12#          ;IS QUOTIENT = -4
(3) 035522 104000                    EMT
(1) 035524 022701 177777          12# :  CMP      #-1,#0+1     ;QUOTIENT IS WRONG
(2) 035530 001401                    BEQ      13#          ;IS REMAINDER = -1
(3) 035532 104000                    EMT
(1) 035534          13# :                    ;WRONG REMAINDER
(1) 035534 005215                    INC      (R5)

;*****
;TEST:260      DIV      0 2 / #-3 = 0      REM = 2      PS = 4
;*****

(1)
(1)
(1)
(1) 035536          TST260:
(1) 035536 012702 000000          MOV      #0,#2         ;LOAD HIGH ORDER WITH 0
(1) 035542 012703 000002          MOV      #2,#2+1       ;LOAD LOW ORDER WITH 2
(1) 035546 071227 177775          DIV      #-3,#2        ;DIVIDE BY #-3
(1) 035552 106737 025626          MFPS     @PPSWORD      ;SAVE PS
(1) 035556 122737 000004 025626  CMPB     #4,@PPSWORD   ;IS PS = 4
(2) 035564 001401                    BEQ      11#
(3) 035566 104000                    EMT
(1) 035570 022702 000000          11# :  CMP      #0,#2         ;PS IS WRONG
(2) 035574 001401                    BEQ      12#          ;IS QUOTIENT = 0
(3) 035576 104000                    EMT
(1) 035600 022703 000002          12# :  CMP      #2,#2+1     ;QUOTIENT IS WRONG
(2) 035604 001401                    BEQ      13#          ;IS REMAINDER = 2
(3) 035606 104000                    EMT
(1) 035610          13# :                    ;WRONG REMAINDER
(1) 035610 005215                    INC      (R5)
```

```

(1)
12035
(1)
(1)
(1)
(1) 035612
(1) 035612 010501
(1) 035614 012704 177777
(1) 035620 012705 177776
(1) 035624 071427 C00003
( ) 035630 10 37 025626
(1) 035634 122737 000004 025626
(2) 035642 001401
(3) 035644 104000
(1) 035646 022704 000000
(2) 035652 001401
(3) 035654 104000
(1) 035656 022705 177776
(2) 035662 001401
(3) 035664 104000
(1) 035666
(1) 035666 010105
(1) 035670 005215
(1)
  
```

```

;*****
;TEST:261      DIV      1 2 / #3 = 0      REM = 2      PS = 4
;*****

TST261:
      MOV      R5,R1      ;SAVE R5
      MOV      # -1,#4    ;LOAD HIGH ORDER WITH 1
      MOV      # -2,#4+1  ;LOAD LOW ORDER WITH 2
      DIV      #3,#4      ;DIVIDE BY #3
      MFPS     @#PSWORD   ;SAVE PS
      CMPB    #4,@#PSWORD ;IS PS = 4
      BEQ     11$
      EMT
      11$:    CMP      #0,#4      ;PS IS WRONG
      BEQ     12$              ;IS QUOTIENT = 0
      EMT
      12$:    CMP      # -2,#4+1  ;QUOTIENT IS WRONG
      BEQ     13$              ;IS REMAINDER = 2
      EMT
      13$:    MOV      R1,R5      ;WRONG REMAINDER
      INC     (R5)             ;RESTORE R5
  
```

```

12036
(1)
(1)
(1)
(1) 035672
(1) 035672 012700 177777
(1) 035676 012701 177777
(1) 035702 071027 000001
(1) 035706 106737 025626
(1) 035712 122737 000010 025626
(2) 035720 001401
(3) 035722 104000
(1) 035724 022700 177777
(2) 035730 001401
(3) 035732 104000
(1) 035734 022701 000000
(2) 035740 001401
(3) 035742 104000
(1) 035744
(1) 035744 005215
(1)
  
```

```

;*****
;TEST:262      DIV      -1 -1 / #1 = -1      REM = 0      PS = 10
;*****

TST262:
      MOV      # -1,#0     ;LOAD HIGH ORDER WITH 1
      MOV      # -1,#0+1  ;LOAD LOW ORDER WITH 1
      DIV      #1,#0      ;DIVIDE BY #1
      MFPS     @#PSWORD   ;SAVE PS
      CMPB    #10,@#PSWORD ;IS PS = 10
      BEQ     11$
      EMT
      11$:    CMP      # -1,#0     ;PS IS WRONG
      BEQ     12$              ;IS QUOTIENT = -1
      EMT
      12$:    CMP      #0,#0+1    ;QUOTIENT IS WRONG
      BEQ     13$              ;IS REMAINDER = 0
      EMT
      13$:    INC     (R5)              ;WRONG REMAINDER
  
```

```

12037
(1)
(1)
(1)
(1) 035746
(1) 035746 012700 000000
(1) 035752 012701 000000
(1) 035756 071027 000001
(1) 035762 106737 025626
(1) 035766 122737 000004 025626
(2) 035774 001401
  
```

```

;*****
;TEST:263      DIV      0 0 / #1 = 0      REM = 0      PS = 4
;*****

TST263:
      MOV      #0,#0      ;LOAD HIGH ORDER WITH 0
      MOV      #0,#0+1   ;LOAD LOW ORDER WITH 0
      DIV      #1,#0      ;DIVIDE BY #1
      MFPS     @#PSWORD   ;SAVE PS
      CMPB    #4,@#PSWORD ;IS PS = 4
      BEQ     11$
  
```

(3) 035776 104000
 (1) 036000 022700 000000
 (2) 036004 001401
 (3) 036006 104000
 (1) 036010 022701 000000
 (2) 036014 001401
 (3) 036016 104000
 (1) 036020
 (1) 036020 005215
 (1)

EMT ;PS IS WRONG
 11: CMP #0,#0 ;IS QUOTIENT = 0
 BEQ 12;
 EMT ;QUOTIENT IS WRONG
 12: CMP #0,#0+1 ;IS REMAINDER = 0
 BEQ 13;
 EMT ;WRONG REMAINDER
 13: INC (R5)

12038

(1)
 (1)
 (1)
 (1) 036022
 (1) 036022 012702 177777
 (1) 036026 012703 125252
 (1) 036032 071227 000002
 (1) 036036 106737 025626
 (1) 036042 122737 000010 025626
 (2) 036050 001401
 (3) 036052 104000
 (1) 036054 022702 152525
 (2) 036060 001401
 (3) 036062 104000
 (1) 036064 022703 000000
 (2) 036070 001401
 (3) 036072 104000
 (1) 036074
 (1) 036074 005215
 (1)

 ;TEST:264 DIV -1 125252 / #2 = 152525 REM = 0 PS = 10

 TST264:
 MOV #-1,#2 ;LOAD HIGH ORDER WITH -1
 MOV #125252,#2+1 ;LOAD LOW ORDER WITH 125252
 DIV #2,#2 ;DIVIDE BY #2
 MFPS @PSWORD ;SAVE PS
 CMPB #10,@PSWORD ;IS PS = 10
 BEQ 11;
 EMT ;PS IS WRONG
 11: CMP #152525,#2 ;IS QUOTIENT = 152525
 BEQ 12;
 EMT ;QUOTIENT IS WRONG
 12: CMP #0,#2+1 ;IS REMAINDER = 0
 BEQ 13;
 EMT ;WRONG REMAINDER
 13: INC (R5)

12039

(1)
 (1)
 (1)
 (1) 036076
 (1) 036076 010501
 (1) 036100 012704 177777
 (1) 036104 012705 177777
 (1) 036110 071427 177777
 (1) 036114 106737 025626
 (1) 036120 122737 000000 025626
 (2) 036126 001401
 (3) 036130 104000
 (1) 036132 022704 000001
 (2) 036136 001401
 (3) 036140 104000
 (1) 036142 022705 000000
 (2) 036146 001401
 (3) 036150 104000
 (1) 036152
 (1) 036152 010105
 (1) 036154 005215
 (1)

 ;TEST:265 DIV -1 -1 / #-1 = 1 REM = 0 PS = 0

 TST265:
 MOV R5,R1 ;SAVE R5
 MOV #-1,#4 ;LOAD HIGH ORDER WITH -1
 MOV #-1,#4+1 ;LOAD LOW ORDER WITH -1
 DIV #-1,#4 ;DIVIDE BY #-1
 MFPS @PSWORD ;SAVE PS
 CMPB #0,@PSWORD ;IS PS = 0
 BEQ 11;
 EMT ;PS IS WRONG
 11: CMP #1,#4 ;IS QUOTIENT = 1
 BEQ 12;
 EMT ;QUOTIENT IS WRONG
 12: CMP #0,#4+1 ;IS REMAINDER = 0
 BEQ 13;
 EMT ;WRONG REMAINDER
 13: MOV R1,R5 ;RESTORE R5
 INC (R5)

12040

(1)

 ;TEST:266 DIV 25253 1 / #125252 = 100000 REM = 1 PS = 10

```
.....
(1) 036156
(1) 036156 012700 025253
(1) 036162 012701 000001
(1) 036166 071027 125252
(1) 036172 106737 025626
(1) 036176 122737 000010 025626
(2) 036204 001401
(3) 036206 104000
(1) 036210 022700 100000 11:
(2) 036214 001401
(3) 036216 104000
(1) 036220 022701 000001 12:
(2) 036224 001401
(3) 036226 104000
(1) 036230 13:
(1) 036230 005215
(1) INC (R5)
```

```
12041
(1) ;TEST:267 DIV 37777 77777 / 77777 = 77777 REM = 77776 PS = 0
(1) ;
(1) ;
```

```
(1) 036232
(1) 036232 012702 037777
(1) 036236 012703 077777
(1) 036242 071227 077777
(1) 036246 106737 025626
(1) 036252 122737 000000 025626
(2) 036260 001401
(3) 036262 104000
(1) 036264 022702 077777 11:
(2) 036270 001401
(3) 036272 104000
(1) 036274 022703 077776 12:
(2) 036300 001401
(3) 036302 104000
(1) 036304 13:
(1) 036304 005215
(1) INC (R5)
```

```
12042
(1) ;TEST:270 DIV 0 100000 / 2 = 40000 REM = 0 PS = 0
(1) ;
(1) ;
```

```
(1) 036306
(1) 036306 010501
(1) 036310 012704 000000
(1) 036314 012705 100000
(1) 036320 071427 000002
(1) 036324 106737 025626
(1) 036330 122737 000000 025626
(2) 036336 001401
(3) 036340 104000
(1) 036342 022704 040000 11:
(2) 036346 001401
(3) 036350 104000
```

(1) 036352 022705 000000
(2) 036356 001401
(3) 036360 104000
(1) 036362
(1) 036362 010105
(1) 036364 005215
(1)
12043
(1)
(1)
(1)
(1) 036366
(1) 036366 012700 177777
(1) 036372 012701 077777
(1) 036376 071027 177776
(1) 036402 106737 025626
(1) 036406 122737 000000 025626
(2) 036414 001401
(3) 036416 104000
(1) 036420 022700 040000
(2) 036424 001401
(3) 036426 104000
(1) 036430 022701 177777
(2) 036434 001401
(3) 036436 104000
(1) 036440
(1) 036440 005215
(1)

124: CMP #0,#4+1 ;IS REMAINDER = 0
BEQ 134 ;WRONG REMAINDER
EMT
134: MOV R1,R5 ;RESTORE R5
INC (R5)
;*****
;TEST:271 DIV 177777 77777 / #177776 = 40000 REM = 177777 PS = 0
;*****
TST271:
MOV #177777,#0 ;LOAD HIGH ORDER WITH 177777
MOV #77777,#0+1 ;LOAD LOW ORDER WITH 77777
DIV #177776,#0 ;DIVIDE BY #177776
MFPS #PSWORD ;SAVE PS
CMPB #0,#PSWORD ;IS PS = 0
BEQ 114
EMT ;PS IS WRONG
114: CMP #40000,#0 ;IS QUOTIENT = 40000
BEQ 124
EMT ;QUOTIENT IS WRONG
124: CMP #177777,#0+1 ;IS REMAINDER = 177777
BEQ 134 ;WRONG REMAINDER
EMT
134: INC (R5)

12044
(1)
(1)
(1)
(1) 036442
(1) 036442 012702 000000
(1) 036446 012703 052525
(1) 036452 071227 052525
(1) 036456 106737 025626
(1) 036462 122737 000000 025626
(2) 036470 001401
(3) 036472 104000
(1) 036474 022702 000001
(2) 036500 001401
(3) 036502 104000
(1) 036504 022703 000000
(2) 036510 001401
(3) 036512 104000
(1) 036514
(1) 036514 005215
(1)

;*****
;TEST:272 DIV 0 52525 / #52525 = 1 REM = 0 PS = 0
;*****
TST272:
MOV #0,#2 ;LOAD HIGH ORDER WITH 0
MOV #52525,#2+1 ;LOAD LOW ORDER WITH 52525
DIV #52525,#2 ;DIVIDE BY #52525
MFPS #PSWORD ;SAVE PS
CMPB #0,#PSWORD ;IS PS = 0
BEQ 114
EMT ;PS IS WRONG
114: CMP #1,#2 ;IS QUOTIENT = 1
BEQ 124
EMT ;QUOTIENT IS WRONG
124: CMP #0,#2+1 ;IS REMAINDER = 0
BEQ 134 ;WRONG REMAINDER
EMT
134: INC (R5)

12045
(1)
(1)
(1)
(1) 036516
(1) 036516 010501
(1) 036520 012704 000000

;*****
;TEST:273 DIV 0 77777 / #0 = DUMMY REM = DUMMY PS = 3
;*****
TST273:
MOV R5,R1 ;SAVE R5
MOV #0,#4 ;LOAD HIGH ORDER WITH 0

(1) 036524 012705 077777
(1) 036530 071427 000000
(1) 036534 106737 025626
(1) 036540 042737 000014 025626
(1) 036546 122737 000003 025626
(2) 036554 001401
(3) 036556 104000
(1) 036560
(1) 036560 010105
(1) 036562 005215
(1)

MOV #77777,#4*1 ;LOAD LOW ORDER WITH 77777
DIV #0,#4 ;DIVIDE BY #0
MFPS @PSWORD ;SAVE PS
BIC #14,@PSWORD
CMPB #3,@PSWORD ;IS PS = 3
BEQ 13#
EHT ;PS IS WRONG
13#:
MOV R1,R5 ;RESTORE R5
INC (R5)

12046

;TEST:274 DIV 77777 177777 / #2 = DUMMY REM = DUMMY PS = 2

(1) 036564
(1) 036564 012700 077777
(1) 036570 012701 177777
(1) 036574 071027 000002
(1) 036600 106737 025626
(1) 036604 042737 000014 025626
(1) 036612 122737 000002 025626
(2) 036620 001401
(3) 036622 104000
(1) 036624
(1) 036624 005215
(1)

TST274:
MOV #77777,#0 ;LOAD HIGH ORDER WITH 77777
MOV #177777,#0*1 ;LOAD LOW ORDER WITH 177777
DIV #2,#0 ;DIVIDE BY #2
MFPS @PSWORD ;SAVE PS
BIC #14,@PSWORD
CMPB #2,@PSWORD ;IS PS = 2
BEQ 13#
EHT ;PS IS WRONG
13#:
INC (R5)

12047 036626 012702 000002
12048 036632 012703 025664
12049 036636 012704 025666

MOV #2,#2
MOV #S9,#3
MOV #S10,#4

12050

;TEST:275 DIV 0 52525 / S9 = 25252 REM = 1 PS = 0

(1) 036642
(1) 036642 012700 000000
(1) 036646 012701 052525
(1) 036652 071067 167006
(1) 036656 106737 025626
(1) 036662 122737 000000 025626
(2) 036670 001401
(3) 036672 104000
(1) 036674 022700 025252
(2) 036700 001401
(3) 036702 104000
(1) 036704 022701 000001
(2) 036710 001401
(3) 036712 104000
(1) 036714
(1) 036714 005215
(1)

TST275:
MOV #0,#0 ;LOAD HIGH ORDER WITH 0
MOV #52525,#0*1 ;LOAD LOW ORDER WITH 52525
DIV S9,#0 ;DIVIDE BY S9
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11#
EHT ;PS IS WRONG
11#:
CMP #25252,#0 ;IS QUOTIENT = 25252
BEQ 12#
EHT ;QUOTIENT IS WRONG
12#:
CMP #1,#0*1 ;IS REMAINDER = 1
BEQ 13#
EHT ;WRONG REMAINDER
13#:
INC (R5)

12052

;TEST:276 DIV 0 52525 / S10 = 25252 REM = 1 PS = 0

(1)
(1)
(1)

(1) 036716
 (1) 036716 012700 000000
 (1) 036722 012701 052525
 (1) 036726 071077 166734
 (1) 036732 106737 025626
 (1) 036736 122737 000000 025626
 (2) 036744 001401
 (3) 036746 104000
 (1) 036750 022700 025252
 (2) 036754 001401
 (3) 036756 104000
 (1) 036760 022701 000001
 (2) 036764 001401
 (3) 036766 104000
 (1) 036770
 (1) 036770 005215
 (1)

TST276:
 MOV #0,#0 ;LOAD HIGH ORDER WITH 0
 MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
 DIV #510,#0 ;DIVIDE BY #510
 MFPS @PASSWORD ;SAVE PS
 CMPB #0,@PASSWORD ;IS PS = 0
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #25252,#0 ;IS QUOTIENT = 25252
 BEQ 12#
 EMT ;QUOTIENT IS WRONG
 12#: CMP #1,#0+1 ;IS REMAINDER = 1
 BEQ 13#
 EMT ;WRONG REMAINDER
 13#: INC (R5)

12053
 (1) ;*****
 (1) ;TEST:277 DIV 0 52525 / #459 = 25252 REM = 1 PS = 0
 (1) ;*****

(1) 036772
 (1) 036772 012700 000000
 (1) 036776 012701 052525
 (1) 037002 071037 025664
 (1) 037006 106737 025626
 (1) 037012 122737 000000 025626
 (2) 037020 001401
 (3) 037022 104000
 (1) 037024 022700 025252
 (2) 037030 001401
 (3) 037032 104000
 (1) 037034 022701 000001
 (2) 037040 001401
 (3) 037042 104000
 (1) 037044
 (1) 037044 005215
 (1)

TST277:
 MOV #0,#0 ;LOAD HIGH ORDER WITH 0
 MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
 DIV #459,#0 ;DIVIDE BY #459
 MFPS @PASSWORD ;SAVE PS
 CMPB #0,@PASSWORD ;IS PS = 0
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #25252,#0 ;IS QUOTIENT = 25252
 BEQ 12#
 EMT ;QUOTIENT IS WRONG
 12#: CMP #1,#0+1 ;IS REMAINDER = 1
 BEQ 13#
 EMT ;WRONG REMAINDER
 13#: INC (R5)

12054
 (1) ;*****
 (1) ;TEST:300 DIV 0 52525 / #2 = 25252 REM = 1 PS = 0
 (1) ;*****

(1) 037046
 (1) 037046 012700 000000
 (1) 037052 012701 052525
 (1) 037056 071002
 (1) 037060 106737 025626
 (1) 037064 122737 000000 025626
 (2) 037072 001401
 (3) 037074 104000
 (1) 037076 022700 025252
 (2) 037102 001401
 (3) 037104 104000
 (1) 037106 022701 000001
 (2) 037112 001401
 (3) 037114 104000

TST300:
 MOV #0,#0 ;LOAD HIGH ORDER WITH 0
 MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
 DIV #2,#0 ;DIVIDE BY #2
 MFPS @PASSWORD ;SAVE PS
 CMPB #0,@PASSWORD ;IS PS = 0
 BEQ 11#
 EMT ;PS IS WRONG
 11#: CMP #25252,#0 ;IS QUOTIENT = 25252
 BEQ 12#
 EMT ;QUOTIENT IS WRONG
 12#: CMP #1,#0+1 ;IS REMAINDER = 1
 BEQ 13#
 EMT ;WRONG REMAINDER

```

(1) 037116
(1) 037116 005215
(1)
12055
(1)
(1)
(1)
(1) 037120
(1) 037120 012700 000000
(1) 037124 012701 052525
(1) 037130 071023
(1) 037132 106737 025626
(1) 037136 122737 000000 025626
(2) 037144 001401
(3) 037146 104000
(1) 037150 022700 025252
(2) 037154 001401
(3) 037156 104000
(1) 037160 022701 000001
(2) 037164 001401
(3) 037166 104000
(1) 037170
(1) 037170 005215
(1)

```

```

134:      INC      (R5)

;*****
;TEST:301      DIV      0 52525 / (3) = 25252      REM = 1      PS = 0
;*****

TST301:
MOV      #0,#0      ;LOAD HIGH ORDER WITH 0
MOV      #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV      (3),#0      ;DIVIDE BY (3)
MFPS     @PSWORD     ;SAVE PS
CMPB    #0,@PSWORD   ;IS PS = 0
BEQ     114
EMT
114:     CMP      #25252,#0 ;PS IS WRONG
BEQ     124 ;IS QUOTIENT = 25252
EMT
124:     CMP      #1,#0+1 ;QUOTIENT IS WRONG
BEQ     134 ;IS REMAINDER = 1
EMT ;WRONG REMAINDER

134:     INC      (R5)

```

```

12056
(1)
(1)
(1)
(1) 037172
(1) 037172 012700 000000
(1) 037176 012701 052525
(1) 037202 071043
(1) 037204 106737 025626
(1) 037210 122737 000000 025626
(2) 037216 001401
(3) 037220 104000
(1) 037222 022700 025252
(2) 037226 001401
(3) 037230 104000
(1) 037232 022701 000001
(2) 037236 001401
(3) 037240 104000
(1) 037242
(1) 037242 005215
(1)

```

```

;*****
;TEST:302      DIV      0 52525 / -(3) = 25252      REM = 1      PS = 0
;*****

TST302:
MOV      #0,#0      ;LOAD HIGH ORDER WITH 0
MOV      #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV      -(3),#0     ;DIVIDE BY -(3)
MFPS     @PSWORD     ;SAVE PS
CMPB    #0,@PSWORD   ;IS PS = 0
BEQ     114
EMT
114:     CMP      #25252,#0 ;PS IS WRONG
BEQ     124 ;IS QUOTIENT = 25252
EMT
124:     CMP      #1,#0+1 ;QUOTIENT IS WRONG
BEQ     134 ;IS REMAINDER = 1
EMT ;WRONG REMAINDER

134:     INC      (R5)

```

```

12057
(1)
(1)
(1)
(1) 037244
(1) 037244 012700 000000
(1) 037250 012701 052525
(1) 037254 071064 000002
(1) 037260 106737 025626
(1) 037264 122737 000000 025626
(2) 037272 001401

```

```

;*****
;TEST:303      DIV      0 52525 / 2(4) = 25252      REM = 1      PS = 0
;*****

TST303:
MOV      #0,#0      ;LOAD HIGH ORDER WITH 0
MOV      #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV      2(4),#0     ;DIVIDE BY 2(4)
MFPS     @PSWORD     ;SAVE PS
CMPB    #0,@PSWORD   ;IS PS = 0
BEQ     114

```

(3) 037274 104000
(1) 037276 022700 025252
(2) 037302 001401
(3) 037304 104000
(1) 037306 022701 000001
(2) 037312 001401
(3) 037314 104000
(1) 037316
(1) 037316 005215
(1)

11\$: EMT ;PS IS WRONG
CMP #25252,#0 ;IS QUOTIENT = 25252
BEQ 12\$
12\$: EMT ;QUOTIENT IS WRONG
CMP #1,#0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

12058

(1)
(1)
(1)
(1) 037320
(1) 037320 012700 000000
(1) 037324 012701 052525
(1) 037330 071074 000000
(1) 037334 106737 025626
(1) 037340 122737 000000 025626
(2) 037346 001401
(3) 037350 104000
(1) 037352 022700 025252
(2) 037356 001401
(3) 037360 104000
(1) 037362 022701 000001
(2) 037366 001401
(3) 037370 104000
(1) 037372
(1) 037372 005215
(1)

;TEST:304 DIV 0 52525 / B(4) = 25252 REM = 1 PS = 0

TST304:
MOV #0,#0 ;LOAD HIGH ORDER WITH 0
MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV B(4),#0 ;DIVIDE BY B(4)
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$
11\$: EMT ;PS IS WRONG
CMP #25252,#0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,#0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

12059

(1)
(1)
(1)
(1) 037374
(1) 037374 012700 000000
(1) 037400 012701 052525
(1) 037404 071034
(1) 037406 106737 025626
(1) 037412 122737 000000 025626
(2) 037420 001401
(3) 037422 104000
(1) 037424 022700 025252
(2) 037430 001401
(3) 037432 104000
(1) 037434 022701 000001
(2) 037440 001401
(3) 037442 104000
(1) 037444
(1) 037444 005215
(1)

;TEST:305 DIV 0 52525 / B(4) = 25252 REM = 1 PS = 0

TST305:
MOV #0,#0 ;LOAD HIGH ORDER WITH 0
MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV B(4),#0 ;DIVIDE BY B(4)
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$
11\$: EMT ;PS IS WRONG
CMP #25252,#0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,#0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

12060

(1)
(1)
(1)

;TEST:306 DIV 0 52525 / B(4) = 25252 REM = 1 PS = 0

(1) 037446
(1) 037446 012700 000000
(1) 037452 012701 052525
(1) 037456 071054
(1) 037460 106737 025626
(1) 037464 122737 000000 025626
(2) 037472 001401
(3) 037474 104000
(1) 037476 022700 025252
(2) 037502 001401
(3) 037504 104000
(1) 037506 022701 000001
(2) 037512 001401
(3) 037514 104000
(1) 037516
(1) 037516 005215
(1)

TST306:

MOV #0,#0 ;LOAD HIGH ORDER WITH 0
MOV #52525,#0+1 ;LOAD LOW ORDER WITH 52525
DIV @-(4),#0 ;DIVIDE BY @-(4)
MFPS @PSWORD ;SAVE PS
CMPB #0,@PSWORD ;IS PS = 0
BEQ 11\$
EMT ;PS IS WRONG
11\$: CMP #25252,#0 ;IS QUOTIENT = 25252
BEQ 12\$
EMT ;QUOTIENT IS WRONG
12\$: CMP #1,#0+1 ;IS REMAINDER = 1
BEQ 13\$
EMT ;WRONG REMAINDER
13\$: INC (R5)

:SPECIAL MULTIPLY DATA PATTERN TEST

TSTSPC: MOV #-1,R1 ;MAKE R1 -1 SO WE KNOW INSTR. WAS MODIFIER
MOV #77700,R0 ;SET UP TEST DATA
MUL #1,R0 ;DO MULTIPLY INSTRUCTION
CMP #77700,R1 ;CHECK LOW ORDER WORD
BEQ 1\$
EMT ;LOW ORDER PRODUCT ERROR
1\$: TST R0 ;CHECK HIGH ORDER WORD
BEQ EISEND
EMT ;HIGH ORDER PRODUCT ERROR
EISEND: JMP MMUTST ;JMP OVER GARBAGE AND GET TO MMU TEST

12061
12062
12063
12064
12065 037520 012701 177777
12066 037524 012700 077700
12067 037530 070027 000001
12068 037534 022701 077700
12069 037540 001401
(2) 037542 104000
12070 037544 005700
12071 037546 001401
(2) 037550 104000
12072 037552 000167 000026

C15

12290
 12291
 12298
 (2)
 (3)
 (2) 037670
 12299 037670 005000
 12300 037672 005001
 12301 037674 106400
 12302 037676 106701
 12303 037700 042701 177437
 12304 037704 020001
 12305 037706 001401
 (2) 037710 104000
 12306
 12307
 12308
 12309 037712 062700 000040
 12310 037716 022700 000400
 12311 037722 001363
 12312
 12317
 (2)
 (3)
 (2) 037724
 12318 037724 005000
 12319 037726 005067 140044
 12320 037732 050067 140040
 12321 037736 016701 140034
 12322 037742 042701 007777
 12323 037746 020001
 12324 037750 001401
 (2) 037752 104000
 12325
 12326
 12327
 12328 037754 062700 010000
 12329 037760 001362
 12330 037762 005067 140010
 12331
 12338
 (2)
 (3)
 (2) 037766
 12339 037766 005067 140004
 12340 037772 012700 000360
 12341 037776 110067 137775
 12342 040002 016701 137770
 12343 040006 042701 007437
 12344 040012 000300
 12345 040014 020001
 12346 040016 001401
 (2) 040020 104000
 12347
 12348
 12349

```

;*****
;TEST 351      PSW PRIORITY BIT TEST
;*****
TS351:
24:  CLR      R0      ;INITIALIZE R0 WITH PRIORITY=0 DATA
    CLR      R1      ;PREPARE R1 TO ACCEPT DATA READ
    MTPS     R0      ;WRITE PRIORITY BITS IN THE PSW
    MFPS     R1      ;READ BACK THE LOW BYTE OF PSW
    BIC      @177437,R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS
    CMP      R0,R1   ;WAS CORRECT PRIORITY SET IN THE PSW?
    BEQ      34
    EMT
                                ;PRIORITY BITS SET WRONG IN PSW
                                ;FOR TIGHTER SCOPE LOOP
                                ;REPLACE ERROR CALL WITH
                                ;"BR 24" = 000770
34:  ADD      @40,R0  ;CHANGE DATA TO NEXT PRIORITY
    CMP      @400,R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?
    BNE      24      ;BRANCH IF NO

;*****
;TEST 352      PSW MODE BIT TEST
;*****
TS352:
24:  CLR      R0      ;INITIALIZE R0 WITH MODE BITS = 0000
    CLR      PSW     ;INITIALIZE PSW
    BIS      R0,PSW  ;BIT SET THE PSW MODE BITS WITH R0
    MOV      PSW,R1  ;READ BACK THE CONTENTS OF THE PSW
    BIC      @007777,R1 ;MASK OFF EVERYTHING EXCEPT THE MODE BITS
    CMP      R0,R1   ;WERE THE MODE BITS SET CORRECTLY?
    BEQ      34
    EMT
                                ;MODE BITS SET WRONG IN PSW
                                ;FOR TIGHTER SCOPE LOOP
                                ;REPLACE ERROR CALL WITH
                                ;"BR 24" = 000763
34:  ADD      @10000,R0 ;CHANGE MODE BIT DATA
    BNE      24      ;BRANCH IF STILL MORE COMBINATIONS
    CLR      PSW     ;RESET PSW BEFORE LEAVING

;*****
;TEST 353      BYTE ADDRESSING TEST FOR PSW
;*****
TS353:
24:  CLR      PSW     ;CLEAR THE PSW
    MOV      @360,R0 ;PUT THE HIGH BYTE DATA INTO R0
    MOVB     R0,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
    MOV      PSW,R1  ;READ BACK THE ENTIRE PSW
    BIC      @007437,R1 ;MASK OFF THE T & CC BITS
    SWAB     R0      ;GET DATA WRITTEN IN HIGH BYTE OF R0
    CMP      R0,R1   ;WAS THE PSW WRITTEN TO CORRECTLY
    BEQ      44
    EMT
                                ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
                                ;FOR TIGHTER SCOPE LOOP
                                ;REPLACE ERROR CALL WITH
                                ;"BR 24" = 000760

```

12350 040022 005067 137750
12351 040026 012700 000340
12352 040032 110067 137740
12353 040036 016701 137734
12354 040042 042701 007437
12355 040046 020001
12356 040050 001401
(3) 040052 104000

41: CLR PSW ;CLEAR THE PSW
MOV #340,R0 ;PUT THE LOW BYTE DATA INTO R0
MOVB R0,PSW ;WRITE THE LOW BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T&CC BITS
CMP R0,R1 ;WAS PSW WRITTEN TO CORRECTLY
BEQ T354
EMT ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 21" = 000736

12357
12358
12359
12360
12369
(2)
(3)

;*****
;TEST 354 TEST AND SETUP OF STACK POINTERS
;*****
T354:

(2) 040054
12370 040054 005067 137716
12371 040060 012706 001000
12372 040064 012767 140000 137704
12373 040072 012706 000600
12374 040076 005067 137674
12375 040102 022706 001000
12376 040106 001401
(3) 040110 104000

CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK,KSP ;SET KERNEL STACK POINTER TO 1100
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;SET USER STACK POINTER TO 700
CLR PSW ;BACK TO KERNEL MODE
CMP #KERSTK,KSP ;IS KERNEL R6 STILL 1100?
BEQ T355
EMT ;KERNEL R6 CHANGED BY WRITING USER R6
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;000756

12377
12378
12379
12380
12381
12382
12383
12384
12385
12386
12387
12388
12389
12390
12391
12392
12393
12403

;*****
;* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE
;* MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,KERNEL & USER PAR/PDR'S).
;* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.
;* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED
;* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF "AND-ING" AND "OR-ING"
;* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE
;* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE
;* THINGS BEING CHECKED.
;*****

12404
(2)
(3)

;*****
;TEST 355 SR0,SR1,SR2,SR3 TIMEOUT TEST
;*****
T355:

(2) 040112
12405 040112 012703 177572
12406 040116 012701 000003
12407 040122 005710
12408
12409 040124 062700 000002
12410 040130 077104
12411 040132 005737 172516

MOV #SR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
MOV #3,R1 ;LOAD R1 WITH THE LOOP COUNT
21: TST (R0) ;TRY ADDRESSING A STATUS REGISTER
;IF IT TIMES OUT GO TO 51
31: ADD #2,R0 ;PUT NEXT ADDRESS IN R0
SOB R1,21 ;LOOP BACK TO 21 UNTIL ALL TESTED
TST #0172516 ;CHECK SR3 FOR RESPONSE

12412
12416
(2)

;*****
;TEST 356 KERNEL PAR'S TIMEOUT TEST
;*****

(3)
(2) 040136
12417
(1) 040136 012700 172340
(1) 040142 012701 000010
(1) 040146 005710
(1)
(1) 040150 062700 000002
(1) 040154 077104
12418
12422
(2)
(3)
(2) 040156
12423
(1) 040156 012700 172300
(1) 040162 012701 000010
(1) 040166 005710
(1)
(1) 040170 062700 000002
(1) 040174 077104
12424
12428
(2)
(3)
(2) 040176
12429
(1) 040176 012700 177640
(1) 040202 012701 000010
(1) 040206 005710
(1)
(1) 040210 062700 000002
(1) 040214 077104
12430
12434
(2)
(3)
(2) 040216
12435
(1) 040216 012700 177600
(1) 040222 012701 000010
(1) 040226 005710
(1)
(1) 040230 062700 000002
(1) 040234 077104
12436
12447
(2)
(3)
(2) 040236
12448
12449 040236 012700 177572
12450 040242 012710 160000
12451 040246 000005
12452 040250 011001
12453 040252 001401

```
.....  
T3556:  
      MOV      #KIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.  
      MOV      #10,R1          ;LOAD R1 WITH LOOP COUNT (8)  
2$:   TST      (R0)            ;TRY ADDRESSING A KIPAR  
      ;IF IT TIMES OUT, WILL GO TO 5$  
3$:   ADD      #2,R0           ;PUT NEXT KIPAR ADDRESS IN R0  
      SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED  
.....  
;TEST 357      KERNEL PDR'S TIMEOUT TEST  
;.....  
T3557:  
      MOV      #KIPDR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.  
      MOV      #10,R1          ;LOAD R1 WITH LOOP COUNT (8)  
2$:   TST      (R0)            ;TRY ADDRESSING A KIPDR  
      ;IF IT TIMES OUT, WILL GO TO 5$  
3$:   ADD      #2,R0           ;PUT NEXT KIPDR ADDRESS IN R0  
      SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED  
.....  
;TEST 360      USER PAR'S TIMEOUT TEST  
;.....  
T360:  
      MOV      #UIPAR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.  
      MOV      #10,R1          ;LOAD R1 WITH LOOP COUNT (8)  
2$:   TST      (R0)            ;TRY ADDRESSING A UIPAR  
      ;IF IT TIMES OUT, WILL GO TO 5$  
3$:   ADD      #2,R0           ;PUT NEXT UIPAR ADDRESS IN R0  
      SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED  
.....  
;TEST 361      USER PDR'S TIMEOUT TEST  
;.....  
T361:  
      MOV      #UIPDR0,R0      ;LOAD R0 WITH ADDRESS OF FIRST REG.  
      MOV      #10,R1          ;LOAD R1 WITH LOOP COUNT (8)  
2$:   TST      (R0)            ;TRY ADDRESSING A UIPDR  
      ;IF IT TIMES OUT, WILL GO TO 5$  
3$:   ADD      #2,R0           ;PUT NEXT UIPDR ADDRESS IN R0  
      SOB     R1,2$           ;LOOP BACK TO 2$ UNTIL ALL TESTED  
.....  
;TEST 362      SRO(15:13) BIT TEST & SR2 TEST  
;.....  
T362:  
1$:   MOV      #SRO,R0          ;LOAD ADDRESS OF SRO INTO R0  
      MOV      #160000,(R0)    ;SET BITS <15:13> IN SRO (ERROR BITS)  
      RESET                                ;ISSUE AND "INIT" SIGNAL  
      MOV      (R0),R1         ;READ SRO INTO R1 TO SEE IF CLEAR  
      BEQ     2$
```

```
(2) 040254 104000 EMT ;SRO<15:13> NOT CLEARED BY A "RESET
12454 ;FOR TIGHTER SCOPE LOOP
12455 ;REPLACE ERROR CALL WITH
12456 ;"BR 1#" = 000770
12457 040256 016767 137314 177302 2#: MOV SR2,WASSR2 ;READ CONTENTS OF SR2
12458 040264 012701 040256 MOV #2#,R1 ;LOAD EXPECTED CONTENTS INTO R1
12459 040270 020167 177272 CMP R1,WASSR2 ;IS SR2 TRACKING?
12460 040274 001401 BEQ 3#
(2) 040276 104000 EMT ;SR2 NOT "TRACKING" VIRTUAL ADDRESSES
12461 ;FOR TIGHTER SCOPE LOOP
12462 ;REPLACE ERROR CALL WITH
12463 ;"BR 2#" = 000767
12464 040300 012701 100000 3#: MOV #100000,R1 ;PUT DATA TO BE WRITTEN IN R1
12465 040304 012703 000003 MOV #3,R3 ;SETUP R3 AS A LOOP COUNTER
12466 040310 005010 4#: CLR (R0) ;CLEAR SRO
12467 040312 050110 5#: BIS R1,(R0) ;SET ONE OF THE ERROR BITS IN SRO
12468 040314 011002 MOV (R0),R2 ;READ SRO INTO R2
12469 040316 020102 CMP R1,R2 ;DID RIGHT ERROR BIT GET SET?
12470 040320 001401 BEQ 6#
(2) 040322 104000 EMT ;BITS WERE SET WRONG IN SRO
12471 ;FOR TIGHTER SCOPE LOOP
12472 ;REPLACE ERROR CALL WITH
12473 ;"BR 4#" = 000772
12474 040324 012704 040312 6#: MOV #5#,R4 ;LOAD EXPECTED CONTENTS OF SR2 IN R4
12475 040330 016767 137242 177230 MOV SR2,WASSR2 ;READ SR2
12476 040336 020467 177224 CMP R4,WASSR2 ;DID SR2 LOCK UP WHEN ERROR
12477 ;BIT SET IN SR1?
12478 040342 001401 BEQ 7#
(2) 040344 104000 EMT ;SR2 DID NOT LOCK UP
12479 ;FOR TIGHTER SCOPE LOOP
12480 ;REPLACE ERROR CALL WITH
12481 ;"BR 4#" = 000761
12482 040346 006001 7#: ROR R1 ;CHANGE DATA TO CHECK NEXT ERROR BIT
12483 040350 077321 SOB R3,4# ;LOOP BACK UNTIL <15:13> ALL TESTED
12484 040352 005010 CLR (R0) ;CLEAR SRO BEFORE LEAVING
12485
12495 ;*****
(2) ;TEST 363 SRO & PSW DUAL ADDRESSING TEST
(3) ;*****
(2) 040354 TS363:
12496 040354 005067 137416 1#: CLR PSW ;CLEAR THE PSW
12497 040360 005067 137206 CLR SRO ;CLEAR STATUS REGISTER 0
12498 040364 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
12499 040370 016700 137176 MOV SRO,R0 ;READ STATUS REGISTER 0
12500 040374 001401 BEQ 2#
(2) 040376 104000 EMT ;SRO EFFECTED BY A WRITE TO THE PSW
12502 ;FOR TIGHTER SCOPE LOOP
12503 ;REPLACE ERROR CALL WITH
12504 ;"BR 1#" = 000767
12505 040400 005067 137166 2#: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
12506 040404 005067 137366 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
12507
12515 ;*****
(2) ;TEST 364 TEST THAT SR1 READS ALL ZEROS
(3) ;*****
```

```

(2) 040410
12516 040410 012700 177777
12517 040414 016700 137154
12518 040420 001401
(2) 040422 104000
12519
12520
12521
12522 040424 012767 177777 132064
12523 040432 022767 000060 132056
12524 040440 001401
(2) 040442 104000
12525 040444 004567 010152
12526 040450 000402
12527 040452 000005
12528 040454 000402
12529 040456 005067 132034
12530 040462 005767 132030
12531 040466
(2) 040466 001401
(3) 040470 104000
12532
12540
12541
12542
12543
12544
(2)
(3)
(2) 040472
12545
12546 040472 012700 172340
12547 040476 012703 000010
12548 040502 005010
12549 040504 011001
12550 040506 001401
(2) 040510 104000
12551
12552
12553
12554 040512 012704 077777
12555 040516 005010
12556 040520 050410
12557 040522 011002
12558 040524 020402
12559 040526 001401
(2) 040530 104000
12560
12561
12562
12563 040532 000261
12564 040534 006004
12565 040536 103767
12566 040540 062700 000002
12567 040544 077322
12568 040546 022700 177660

```

```

TS364:
14: MOV #1,R0 ;FILL R0 WITH ALL ONES
MOV SR1,R0 ;READ SR1 INTO R0
BEQ 24
EMT ;SR1 DID NOT READ ALL ZEROS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;000772
24: MOV #-1,SR3 ;TRY TO WRITE ONES TO SR3
CMP #60,SR3 ;ONLY BITS <5:4> SHOULD BE ONES
BEQ 34
EMT ;DIDN'T READ BACK A "60"
34: JSR R5,CHKAPT
BR 904
RESET ;CLEARS SR3
BR 914
904: CLR SR3
914: TST SR3 ;VERIFY THAT IT WAS CLEARED
44: BEQ TS365
EMT ;SR3 DIDN'T READ ALL ZEROS

```

```

;NOTE F11 CHANGES INCLUDED CHECKING ALL BITS<15:0> OF PARS
; INSTEAD OF ONLY BITS<11:0>.

```

```

;*****
;TEST 365 BIT TEST OF KERNEL & USER PAR'S
;*****

```

```

TS365:
14: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR IN R0
24: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
34: CLR (R0) ;CLEAR THE PAR
MOV (R0),R1 ;READ THE PAR INTO R1
BEQ 44
EMT ;PAR WOULD NOT CLEAR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 34" = 000774
44: MOV #077777,R4 ;LOAD "WALKING 0" TEST PATTERN IN R4
54: CLR (R0) ;CLEAR THE PAR BEFORE LOADING DATA
BIS R4,(R0) ;BIT SET THE TEST PATTERN INTO THE PAR
MOV (R0),R2 ;READ THE PAR INTO R2
CMP R4,R2 ;DOES DATA WRITTEN=DATA READ?
BEQ 64
EMT ;PAR BITS DID NOT SET CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 54" = 000767
64: SEC ;SET THE C-BIT FOR THE ROTATE INST.
ROR R4 ;ROTATE THE TEST PATTERN IN R4
BCS 54 ;BRANCH BACK IF MORE BITS TO TEST
ADD #2,R0 ;GET NEXT PAR ADDRESS IN R0
SOB R3,34 ;BRANCH BACK UNTIL ALL PAR'S TESTED
CMP #UIPAR7*2,R0 ;HAVE USER PAR'S BEEN TESTED

```

```

12569 040552 103003
12570 040554 012700 177640
12571 040560 000746
12572
12581
(2)
(3)
(2) 040562
12582
12583 040562 012700 172300
12584 040566 012703 000010
12585 040572 005010
12586 040574 011001
12587 040576 001401
(2) 040600 104000
12588
12589
12590
12591 040602 012704 077777
12592 040606 005010
12593 040610 010401
12594 040612 042701 100361
12595 040616 050110
12596 040620 011002
12597 040622 020102
12598 040624 001401
(2) 040626 104000
12599
12600
12601
12602 040630 000261
12603 040632 006004
12604 040634 103764
12605 040636 062700 000002
12606 040642 077325
12607 040644 022700 177620
12608 040650 103003
12609 040652 012700 177600
12610 040656 000743
12611
12612
12613
12621
(2)
(3)
(2) 040660
12622
12623 040660 012700 172340
12624 040664 012703 000010
12625 040670 012701 177777
12626 040674 005010
12627 040676 110110
12628 040700 011002
12629 040702 042701 177400
12630 040706 020102
12631 040710 001401

```

```

BHIS TS366 ;GET TO NEXT TEST
MOV #UIPARO,R0 ;LOAD FIRST USER PAR ADDR. IN R0
BR 2# ;BRANCH BACK TO TEST USER PAR'S
;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
;*****
;TEST 366 BIT TEST OF KERNEL & USER PDR'S
;*****
TS366:
1#: MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
2#: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
3#: CLR (R0) ;CLEAR THE PDR
MOV (R0),R1 ;READ THE PDR INTO R1
BEQ 4#
EMT ;PDR WOULD NOT CLEAR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 3#" = 000774
4#: MOV #077777,R4 ;LOAD "WALKING 0" TEST PATTERN IN R4
5#: CLR (R0) ;CLEAR THE PDR BEFORE LOADING DATA
MOV R4,R1 ;LOAD DATA INTO R1
BIC #100361,R1 ;MASK UNUSED BITS OUT OF THE DATA
BIS R1,(R0) ;BIT SET THE TEST PATTERN INTO THE PDR
MOV (R0),R2 ;READ THE PDR INTO R2
CMP R1,R2 ;DOES DATA WRITTEN=DATA READ?
BEQ 6#
EMT ;PDR BITS DID NOT SET CORRECTLY
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 5#" = 000767
6#: SEC ;SET THE C-BIT FOR THE ROTATE INST.
ROR R4 ;ROTATE THE TEST PATTERN IN R4
BCS 5# ;BRANCH BACK IF MORE BITS TO TEST
ADD #2,R0 ;GET NEXT PDR ADDRESS IN R0
SOB R3,3# ;BRANCH BACK UNTIL ALL PDR'S TESTED
CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
BHIS TS367 ;GET TO NEXT TEST
MOV #UIPDR0,R0 ;LOAD FIRST USER PDR ADDR. IN R0
BR 2# ;BRANCH BACK TO TEST USER PDR'S
;LEAVE TEST WITH ALL WRITEABLE BITS IN
;ALL PDR'S = 1
;*****
;TEST 367 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
;*****
TS367:
1#: MOV #KIPARO,R0 ;LOAD ADDRESS OF FIRST PAR INTO R0
MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PAR'S
3#: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
CLR (R0) ;CLEAR THE PAR
MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PAR
MOV (R0),R2 ;READ THE ENTIRE PAR INTO R2
BIC #177400,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO
BEQ 5#

```



```
(2) 040712 104000 EMT ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
12632 ;FOR TIGHTER SCOPE LOOP
12633 ;REPLACE ERROR CALL WITH
12634 ;"BR 3$" = 000766
12635 040714 005010 5$: CLR (R0) ;CLEAR THE PAR
12636 040716 012701 177777 MOV #-1,R1 ;LOAD TEST, PATTERN INTO R1
12637 040722 110160 000001 MOVB R1,1(R0) ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
12638 040726 011002 MOV (R0),R2 ;READ THE ENTIRE PAR INTO R2
12639 ;F11 CHANGE WAS #170377
12640 040730 042701 000377 BIC #000377,R1 ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
12641 040734 020102 CMP R1,R2 ;WAS ONLY THE HIGH BYTE WRITTEN TO?
12642 040736 001401 BEQ 6$
(2) 040740 104000 EMT ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
12643 ;FOR TIGHTER SCOPE LOOP
12644 ;REPLACE ERROR CALL WITH
12645 ;"BR 5" = 000765
12646 040742 062700 000002 6$: ADD #2,R0 ;PUT ADDRESS OF NEXT PAR IN R0
12647 040746 077330 SOB R3,3$ ;BRANCH BACK UNTIL 8 PAR'S TESTED
12648 040750 022700 177660 CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
12649 040754 103003 BHIS TS370 ;GET TO NEXT TEST
12650 040756 012700 177640 MOV #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
12651 040762 000742 BR 3$ ;BRANCH BACK TO TEST USER PAR'S
12652
12660 ;*****
(2) ;TEST 370 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
(3) ;*****
(2) 040764 TS370:
12661 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR INTO R0
12662 040764 012700 172300 MOV #10,R3 ;LOAD LOOP COUNTER TO DO 8 PDR'S
12663 040770 012703 000010 3$: MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
12664 040774 012701 177777 CLR (R0) ;CLEAR THE PDR
12665 041000 005010 MOVB R1,(R0) ;WRITE 1'S TO THE LOW BYTE OF THE PDR
12666 041002 110110 MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
12667 041004 011002 BIC #177761,R1 ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
12668 041006 042701 177761 CMP R1,R2 ;WAS ONLY THE LOW BYTE WRITTEN TO?
12669 041012 020102 BEQ 5$
12670 041014 001401 EMT ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
(2) 041016 104000 ;FOR TIGHTER SCOPE LOOP
12671 ;REPLACE ERROR CALL WITH
12672 ;"BR 3$" = 000766
12673 5$: CLR (R0) ;CLEAR THE PDR
12674 041020 005010 MOV #-1,R1 ;LOAD TEST PATTERN INTO R1
12675 041022 012701 177777 MOVB R1,1(R0) ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
12676 041026 110160 000001 MOV (R0),R2 ;READ THE ENTIRE PDR INTO R2
12677 041032 011002 BIC #100377,R1 ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
12678 041034 042701 100377 CMP R1,R2 ;WAS ONLY THE HIGH BYTE WRITTEN TO?
12679 041040 020102 BEQ 6$
(2) 041044 104000 EMT ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
12681 ;FOR TIGHTER SCOPE LOOP
12682 ;REPLACE ERROR CALL WITH
12683 ;"BR 5$" = 000765
12684 041046 062700 000002 6$: ADD #2,R0 ;PUT ADDRESS OF NEXT PDR IN R0
12685 041052 077330 SOB R3,3$ ;BRANCH BACK UNTIL 8 PDR'S TESTED
12686 041054 022700 177620 CMP #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
12687 041060 103003 BHIS TS371 ;GET TO NEXT TEST
```

J15

12688 041062 012700 177600
 12689 041066 000742
 12690
 12702
 (2)
 (3)
 (2) 041070
 12703
 12704 041070 012703 000010
 12705 041074 012700 172300
 12706 041100 004767 007242
 12707 041104 012706 001000
 12708 041110 005010
 12709 041112 004767 007322
 12710 041116 012720 177777
 12711 041122 077310
 12712 041124 012703 000010
 12713 041130 012700 172340
 12714 041134 012706 001000
 12715 041140 005010
 12716 041142 004767 007272
 12717 041146 012720 177777
 12718 041152 077310
 12719 041154 012703 000010
 12720 041160 012700 177600
 12721 041164 012706 001000
 12722 041170 005010
 12723 041172 004767 007242
 12724 041176 012720 177777
 12725 041202 077310
 12726 041204 012703 000010
 12727 041210 012700 177640
 12728 041214 012706 001000
 12729 041220 005010
 12730 041222 004767 007212
 12731 041226 012720 177777
 12732 041232 077310
 12733
 12743
 (2)
 (3)
 (2) 041234
 12744
 12745
 12746 041234 032737 000001 001020
 (1) 041242 001403
 (1) 041244 005737 001006
 (2) 041250 001063
 (1) 041252
 12747 041252 004767 007070
 12748 041256 000005
 12749 041260 012700 172300
 12750 041264 012704 000010
 12751 041270 011001
 12752 041272 022701 077416
 12753 041276 001401

```

MOV    #UIPDRO,RO    ;LOAD ADDRESS OF FIRST USER PDR IN RO
BR     3$             ;BRANCH BACK TO TEST USER PDR'S

;*****
;TEST 371    PAR-PDR DUAL ADDRESSING TEST
;*****
TS371:

MOV    #10,R3        ;LOAD LOOP COUNTER WITH AN 8
MOV    #KIPDRO,RO    ;LOAD ADDRESS OF FIRST KERNEL PDR AND RO
JSR    PC,SETREG     ;SET ALL BITS IN ALL PAR'S IN PDR'S
2$:   MOV    #KERSTK,KSP ;SETUP STACK POINTER
      CLR    (RO)      ;CLEAR ONE OF THE KERNEL PDR'S
      JSR    PC,CMPIREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
      MOV    #-1,(RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PDR
      SOB   R3,2$      ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
      MOV    #10,R3    ;LOAD LOOP COUNTER WITH AN 8
      MOV    #KIPARO,RO ;LOAD ADDRESS OF FIRST KERNEL PAR IN RO
      MOV    #KERSTK,KSP ;SETUP STACK POINTER
      CLR    (RO)      ;CLEAR ONE OF THE KERNEL PAR'S
      JSR    PC,CMPIREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
      MOV    #-1,(RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PAR
      SOB   R3,3$      ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
      MOV    #10,R3    ;LOAD LOOP COUNTER WITH AN 8
      MOV    #UIPDRO,RO ;LOAD ADDRESS OF FIRST USER PDR IN RO
      MOV    #KERSTK,KSP ;SETUP STACK POINTER
      CLR    (RO)      ;CLEAR ONE OF THE USER PDR'S
      JSR    PC,CMPIREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
      MOV    #-1,(RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
      SOB   R3,4$      ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
      MOV    #10,R3    ;LOAD LOOP COUNTER WITH AN 8
      MOV    #UIPARO,RO ;LOAD ADDRESS OF FIRST USER PAR IN RO
      MOV    #KERSTK,KSP ;SETUP STACK POINTER
      CLR    (RO)      ;CLEAR ONE OF THE USER PAR'S
      JSR    PC,CMPIREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
      MOV    #-1,(RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
      SOB   R3,5$      ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED

;*****
;TEST 372    TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
;*****
TS372:

BIT    #1, #ENV      ;ARE WE RUNNING UNDER APT
BEQ    70$           ;IF NO THEN DO TEST
TST    #PASS         ;IS THIS FIRST PASS
BNE    TS373        ;IF NO THEN SHIP TO NEXT TEST

70$:   JSR    PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
      RESET ;ISSUE AN "INIT" BY EXECUTING A RESET
10$:   MOV    #KIPDRO,RO ;LOAD ADDRESS OF FIRST KERNEL PDR IN RO
      MOV    #10,R4    ;LOAD LOOP COUNTER WITH AN 8
      MOV    (RO),R1   ;READ A KERNEL PDR INTO R1
      CMP    #77416,R1 ;ARE ALL THE BITS STILL SET?
      BEQ    3$

```


(1)	042342	012701	111400		MOV	#111400,R1	;LOAD VIRTUAL ADDR. VBA INTO R1	
(1)	042346	012702	125256		MOV	#125256,R2	;LOAD TEST PATTERN INTO R2	
(1)	042352	012704	177664		MOV	#177664,R4	;LOAD R4 WITH PAR VALUE	
(1)	042356	010467	127766		MOV	R4,KIPAR4	;LOAD KERNEL PAR 4 BITS <11:00>	
(1)	042362	011067	175204		MOV	(R0),#TMP0	;SAVE CONTENTS AT TEST LOCATION	
(1)	042366	052767	000020	130122	BIS	#BIT4,SR3	;SET UP FOR 22-BIT ADDRESSING	
(1)	042374	052767	000001	135170	BIS	#BIT0,SRO	;TURN ON "RELOCATION"	
(1)	042402	010211			MOV	R2,(R1)	;LOAD 125256 USING ADDER (PAR4 + VIRT ADDR)	
(1)	042404	005067	135162		CLR	SRO	;TURN OFF MEMORY MGMT.	
(1)	042410	011003			MOV	(R0),R3	;READ 125256 BACK WITHOUT USING MEM. MGMT	
(1)	042412	016710	175154		MOV	#TMP0,(R0)	;RESTORE ORIGINAL CONTENTS TO TEST LOC.	
(1)	042416	020203			CMF	R2,R3	;WAS SAME PATTERN READ BACK THAT WAS	
(1)							;WRITTEN USING "DEST-ONLY-RELOC."?	
(2)	042420	001401			BEG	91		
(3)	042422	104000			EMT		;TEST LOCATION DID NOT HAVE PATTERN	
(1)							;THAT SHOULD HAVE BEEN WRITTEN TO IT.	
(1)							;APPARENTLY PHYSICAL ADDR. WAS	
(1)							;FORMED WRONG BY ADDERS USING	
(1)							;THE VIRTUAL ADDR. AND KIPAR4	
(1)							;FOR TIGHTER SCOPE LOOP	
(1)							;REPLACE ERROR CALL WITH	
(1)							;"BR 81" = 000742	
(1)	042424			91:				
12913								
12937								
(2)							;*****	
(3)							;TEST 375 READ AND WRITE WHILE IN RELOCATE MODE	
(2)	042424						;*****	
12938							TS375:	
12939	042424	005067	135346	11:	CLR	PSW	;START IN KERNEL MODE	
12940	042430	012704	001377		MOV	#1377,R4	;LOAD R4 WITH VALUE FOR PAR4	
12941	042434	012705	001400		MOV	#1400,R5	;LOAD R5 WITH VALUE FOR PAR5	
12942	042440	010467	127704		MOV	R4,KIPAR4	;LOAD KERNEL PAR4	
12943	042444	010567	127702		MOV	R5,KIPAR5	;LOAD KERNEL PAR5	
12944	042450	012700	177640		MOV	#UIPAR0,R0	;LOAD ADDRESS OF FIRST USER PAR IN R0	
12945	042454	005001			CLR	R1	;CLEAR R1	
12946	042456	012702	000007		MOV	#7,R2	;LOAD LOOP COUNTER WITH A 7	
12947	042462	010120		21:	MOV	R1,(R0)+	;MAP USER PAR'S TO PAGES 0-6 (4K EACH)	
12948	042464	062701	000200		ADD	#200,R1		
12949	042470	077204			SOB	R2,21	;LOOP UNTIL UIPAR0-UIPAR6 ARE LOADED	
12950	042472	012710	177600		MOV	#177600,(R0)	;MAP USER PAR7 TO THE I/O PAGE	
12951	042476	012700	177600		MOV	#UIPDR0,R0	;LOAD ADDRESS OF FIRST USER PDR IN R0	
12952	042502	012701	077406		MOV	#77406,R1	;LOAD PDR DATA INTO R1	
12953	042506	012702	000010		MOV	#10,R2	;LOAD LOOP COUNTER WITH AN 8	
12954	042512	010120		31:	MOV	R1,(R0)+	;MAP ALL 8 PAGES 128 BLOCKS, UPWARD	
12955	042514	077202			SOB	R2,31	; EXPANDABLE, READ/WRITE	
12956	042516	012767	042762	135524	MOV	#81,MMVEC	;SET M. M. TRAP VECTOR TO 81	
12957	042524	052767	000020	127764	BIS	#BIT4,SR3	;SET UP FOR 22-BIT ADDRESSING	
12958	042532	012767	000001	135032	MOV	#BIT0,SRO	;TURN ON MEMORY MANAGEMENT	
12959	042540	105067	135044		CLRB	UIPDR4	;MAP USER SPACE NON-RESIDENT WHILE	
12960	042544	105067	135042		CLRB	UIPDR5	; TESTING KERNEL SPACE	
12961	042550	010567	135074		MOV	R5,UIPAR4	;MAP USER PAR'S OPPOSITE OF KIPAR'S	
12962	042554	010467	135072		MOV	R4,UIPAR5		
12963	042560	016767	135212	175004	41:	MOV	PSW,#TMP0	;SAVE PSW IN CASE OF ERROR
12964	042566	012700	100100		MOV	#100100,R0	;PUT VIRTUAL ADDR. THAT USES PAR4 IN R0	
12965	042572	012701	120000		MOV	#120000,R1	;PUT VIRTUAL ADDR. THAT USES PAR5 IN R1	

12966	042576	010010			58:	MOV	R0,(R0)	;WRITE TO TEST LOC. USING PAR4
12967	042600	011102				MOV	(R1),R2	;READ THE SAME LOC., BUT USING PAR4
12968	042602	020002				CMP	R0,R2	;DID WE READ WHAT WE WROTE?
12969	042604	001401				BEG	68	
(2)	042606	104000				EMT		;READING LOC. USING PARS AND A VIRT.
12970								;ADDR. DID NOT FIND DATA WRITTEN WHEN USING
12971								;PAR4 AND VIRT. ADDRESS.
12972								;FOR TIGHTER SCOPE LOOP
12973								;REPLACE ERROR CALL WITH
12974								; "BR 58" = 000765
12975	042610	062700	000100		68:	ADD	#100,R0	;CHANGE VIRTUAL ADDR. TO POINT TO NEXT BLOCK
12976	042614	062701	000100			ACD	#100,R1	
12977	042620	020127	127700			CMP	R1,#127700	;WERE BLOCKS FROM 60000-676000 ALL TRIED?
12978	042624	001364				BNE	58	;BRANCH IF NO
12979	042626	032767	140000	135142		BIT	#140000,PSW	;HAVE WE DONE TEST IN USER MODE YET?
12980	042634	001026				BNE	78	;BRANCH IF YES
12981	042636	010467	135006			MOV	R4,UIPAR4	;LOAD USER PAR4
12982	042642	010567	135004			MOV	R5,UIPAR5	;LOAD USER PARS
12983	042646	112767	000006	134734		MOV	#6,UIPDR4	;MAP USER SPACE R/W TO TEST IT
12984	042654	112767	000006	134730		MOV	#6,UIPDR5	
12985	042662	105067	127422			CLRB	KIPDR4	;MAP KERNEL SPACE NON-RESIDENT WHILE
12986	042666	105067	127420			CLRB	KIPDR5	; TESTING USER SPACE
12987	042672	010567	127452			MOV	R5,KIPAR4	;MAP KERNEL PAR'S OPPOSITE UIPAR'S
12988	042676	010467	127450			MOV	R4,KIPAR5	
12989	042702	012767	140000	135066		MOV	#140000,PSW	;GO TO USER MODE
12990	042710	000723				BR	48	;GO BACK AND READ/WRITE IN USER MODE
12991	042712	005067	135060		78:	CLR	PSW	;GO BACK TO KERNEL MODE BEFORE LEAVING
12992	042716	012767	077406	127364		MOV	#77406,KIPDR4	;REMAP KERNEL PAGES READ/WRITE
12993	042724	012767	077406	127360		MOV	#77406,KIPDR5	
12994	042732	010567	127412			MOV	R5,KIPAR4	;MAP KERNEL AND USER PAR'S 4 & 5
12995	042736	010567	127410			MOV	R5,KIPAR5	; BACK TO 12-16K
12996	042742	010567	134702			MOV	R5,UIPAR4	
12997	042746	010567	134700			MOV	R5,UIPAR5	
12998	042752	012767	021550	135270		MOV	#T0250,MVVEC	;RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE
12999	042760	000404				BR	T5376	;GET TO NEXT TEST
13000	042762	042767	160000	134602	88:	BIC	#160000,SRO	;CLEAR ERROR BITS IN SRO
13001	042770	104000				EMT		;M.M. TRAP WHILE IN RELOCATE MODE
13002								;REFERENCED WRONG SET OF PDR'S
13003								;FOR TIGHTER SCOPE LOOP
13004								;REPLACE ERROR CALL WITH
13005								;A "NOP" = 000240
13006								

 ;TEST 376 W-BIT LOGIC TEST, KERNEL PDR'S

13010	(2)							
13011	(3)							
(2)	042772							
(1)	042772	004767	005262		18:	JSR	PC,TOFF	;TURN T-BIT TRAPPING OFF FOR THIS TEST
(1)	042776	012702	000004			MOV	#4,R2	;SET LOOP COUNTER TO 4
(1)	043002	012700	172346			MOV	#KIPAR3,R0	;LOAD ADDRESS OF PARS INTO R0
(1)	043006	012701	001400			MOV	#1400,R1	;LOAD "24-28K" PAR VALUE INTO R1
(1)	043012	010120			28:	MOV	R1,(R0)+	;MAP PARS 3-6 TO 12-16K
(1)	043014	077202				SQB	R2,28	;LOOP TIL ALL 4 OF THEM LOADED
(1)	043016	012705	172300			MOV	#KIPDR0,R5	;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
(1)	043022	012704	000010			MOV	#10,R4	;SET LOOP COUNTER TO 8
(1)	043026	012703	017776			MOV	#17776,R3	;INITIALIZE VIRTUAL ADDRESS TO BE IN R3

```

(1) 043032 012700 172300 38: MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
(1) 043036 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
(1) 043042 012701 077406 MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
(1) 043046 010120 48: MOV R1,(R0) ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
(1) 043050 077202 SOB R2,48 ;LOOP UNTIL ALL OF THEM SETUP
(1) 043052 011313 MOV (R3),(R3) ;DO "DATO" TO VIRTUAL ADDR. SETTING A W BIT
(1) 043054 031527 600100 BIT (R5),#WBIT ;DID THAT CAUSE W-BIT TO BE SET?
(2) 043060 001001 BNE 58
(3) 043062 104000 EMT ;W-BIT DID NOT GET SET IN PDR
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;"BR 38" = 000763
(1) 043064 012702 000010 58: MOV #10,R2 ;SET LOOP COUNTER TO 8
(1) 043070 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
(1) 043074 031027 000100 68: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
(1) 043100 001403 BEQ 78 ;BRANCH IF YES
(1) 043102 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
(2) 043104 001401 BEQ 78
(3) 043106 104000 EMT ;W-BIT GOT SET IN MORE THAN ONE PDR
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;"BR 38" = 000750
(1) 043110 062700 000002 78: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
(1) 043114 077211 SOB R2,68 ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
(1) 043116 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
(1) 043120 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
(2) 043124 001401 BEQ 88
(3) 043126 104000 EMT ;W-BIT DID NOT CLEAR BY WRITING THE PDR
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;"BR 38" = 000740
(1) 043130 062705 000002 88: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
(1) 043134 062703 020000 ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
(1) 043140 077444 SOB R4,38 ;LOOP BACK TO 38 UNTIL ALL 8 PDR'S TESTED
(1) 043142 004767 005146 JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST

```

```

13012
13016 ;*****
(2) ;TEST 377 W-BIT LOGIC TEST, USER PDR'S
(3) ;*****
(2) TS377:

```

```

13017 043146 012767 140000 134622 18: MOV #140000,PSW ;GO TO USER MODE FOR THIS TEST
13018 043154 004767 005100 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
(1) 043160 012702 000004 MOV #4,R2 ;SET LOOP COUNTER TO 4
(1) 043164 012700 177646 MOV #UIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
(1) 043170 012701 001400 MOV #1400,R1 ;LOAD "24-28K" PAR VALUE INTO R1
(1) 043174 010120 28: MOV R1,(R0) ;MAP PAR3 3-6 TO 12-16K
(1) 043176 077202 SOB R2,28 ;LOOP TIL ALL 4 OF THEM LOADED
(1) 043200 012705 177600 MOV #UIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
(1) 043204 012704 000010 MOV #10,R4 ;SET LOOP COUNTER TO 8
(1) 043210 012703 017776 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
(1) 043214 012700 177600 38: MOV #UIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
(1) 043220 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
(1) 043224 012701 077406 MOV #77406,R1 ;PUT "W-BIT OFF DATA" INTO R1
(1) 043230 010120 48: MOV R1,(R0) ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
(1) 043232 077202 SOB R2,48 ;LOOP UNTIL ALL OF THEM SETUP
(1) 043234 011313 MOV (R3),(R3) ;DO "DATO" TO VIRTUAL ADDR. -SETTING A W-BIT

```

(1)	043236	031527	000100		BIT	(R5),#WBIT	;DID THAT CAUSE W BIT TO BE SET?
(2)	043242	001001			BNE	5:	
(3)	043244	104000			EMT		;W BIT DID NOT GET SET IN PDR
(1)							;FOR TIGHTER SCOPE LOOP
(1)							;REPLACE ERROR CALL WITH
(1)							; "BR 3:" = 000763
(1)	043246	012702	000010	5:	MOV	#10,R2	;SET LOOP COUNTER TO 8
(1)	043252	012700	177600		MOV	#UIPDR0,R0	;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
(1)	043256	031027	000100	6:	BIT	(R0),#WBIT	;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
(1)	043262	001403			BEQ	7:	;BRANCH IF YES
(1)	043264	020500			CMP	R5,R0	;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
(2)	043266	001401			BEQ	7:	
(3)	043270	104000			EMT		;W-BIT GOT SET IN MORE THAN ONE PDR
(1)							;FOR TIGHTER SCOPE LOOP
(1)							;REPLACE ERROR CALL WITH
(1)							; "BR 3:" = 000750
(1)	043272	062700	000002	7:	ADD	#2,R0	;POINT R0 TO NEXT PDR TO BE CHECKED
(1)	043276	077211			SOB	R2,6:	;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
(1)	043300	010115			MOV	R1,(R5)	;WRITE TO THE PDR TESTED TO CLEAR W-BIT
(1)	043302	031527	000100		BIT	(R5),#WBIT	;DID WRITING PDR CLEAR THE W-BIT?
(2)	043306	001401			BEQ	8:	
(3)	043310	104000			EMT		;W-BIT DID NOT CLEAR BY WRITING THE PDR
(1)							;FOR TIGHTER SCOPE LOOP
(1)							;REPLACE ERROR CALL WITH
(1)							; "BR 3:" = 000740
(1)	043312	062705	000002	8:	ADD	#2,R5	;POINT R5 TO THE NEXT PDR TO BE TESTED
(1)	043316	062703	020000		ADD	#20000,R3	;CHANGE VIRT. ADDR TO REF. NEXT PDR
(1)	043322	077444			SOB	R4,3:	;LOOP BACK TO 3: UNTIL ALL 8 PDR'S TESTED
(1)	043324	004767	004764		JSR	PC,TON	;TURN T-BIT BACK ON FOR NEXT TEST
13019	043330	005067	134442		CLR	PSW	;BACK TO KERNEL MODE BEFORE LEAVING

```

;*****
;TEST 400      TEST "W-BIT" SPECIAL CASES
;*****
TS400:

```

13031							
13032	043334	004767	004720	1:	JSR	PC,TOFF	;TURN OFF T-BIT TRAPPING FOR THIS TEST
13033	043340	012701	077406		MOV	#77406,R1	;PUT "W-BIT OFF" VALUE FOR PDR IN R1
13034	043344	010167	126746	2:	MOV	R1,KIPDR7	;LOAD KERNEL PDR 7 TO CLEAR W-BIT
13035	043350	016700	134216		MOV	SRO,R0	;READ PRESENT CONTENTS OF STATUS REG. 0
13036	043354	010067	134212		MOV	R0,SRO	;WRITE PRESENT CONTENTS OF SRO BACK TO ITSELF
13037	043360	016702	126732		MOV	KIPDR7,R2	;READ CONTENTS OF KIPDR7 INTO R2
13038	043364	020102			CMP	R1,R2	;WAS W-BIT LEFT CLEARED?
13039	043366	001401			BEQ	3:	
(2)	043370	104000			EMT		;W-BIT IN KIPDR7 SET WHEN SRO WAS WRITTEN TO
13040							;FOR TIGHTER SCOPE LOOP
13041							;REPLACE ERROR CALL WITH
13042							; "BR 2:" = 000765
13043	043372	010167	126716	3:	MOV	R1,KIPDR6	;LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
13044	043376	012767	043410	134400	MOV	#4:,ERRVEC	;SET UP LOC. 4 TO 4: FOR ODD ADDR. ABORT
13045	043404	005037	140000		CLR	#140000	;CAUSE TIMEOUT ABORT THRU LOC. 4
13046	043410	012706	001000	4:	MOV	#KERSTK,KSP	;RESTORE THE STACK POINTER
13047	043414	016702	126674		MOV	KIPDR6,R2	;READ KIPDR6 INTO R2
13048	043420	052701	000100		BIS	#100,R1	;R1-77506
13049	043424	020102			CMP	R1,R2	;WAS W-BIT SET?
13050	043426	001401			BEQ	5:	

```

(2) 043430 104000          EMT          ;W-BIT WAS NOT SET DURING A TIMEOUT ABOP
13051                                     ;FOR TIGHTER SCOPE LOOP
13052                                     ;REPLACE ERROR CALL WITH
13053                                     ;"BR 31" = 000757
13054 043432 010167 126656 51:  MOV      R1,KIPDR6  ;RESTORE KIPDR6 TO 77406
13055 043436 012767 001400 126710  MOV      #1400,KIPAR6  ;RESTORE KIPAR6 TO 1400
13056 043444 012767 021526 134332  MOV      #T04,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
13057 043452 004767 004636          JSR      PC,T04       ;TURN T-BIT TRAPPING BACK ON
13058
13059 ;*****
13060 ;*
13061 ;* THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
13062 ;* TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
13063 ;* ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
13064 ;* VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
13065 ;* THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
13066 ;* ARE CHECKED IN SRO. SO THE SRO AND SR2 LOGIC AND THE
13067 ;* KT ERROR LOGIC ARE CHECKED.
13068 ;*
13069 ;*****
13070
13079 ;*****
(2) ;TEST 401 NON-RESIDENT ABORT TEST (ACF=0EA)
(3) ;*****
(2) 043456 TS401:
13080
13081 043456 012700 001400 11:  MOV      #1400,R0      ;LOAD DATA FOR PAR'S INTO R0
13082 043462 010067 126660          MOV      R0,KIPAR3    ;MAP KERNEL PAR'S 3EA TO 24-28K
13083 043466 010067 126656          MOV      R0,KIPAR4
13084 043472 010067 134150          MOV      R0,UIPAR3    ;MAP USER PAR'S 3EA TO 24-28K
13085 043476 010067 134146          MOV      R0,UIPAR4
13086 043502 012767 077406 126576  MOV      #77406,KIPDR3 ;MAP KERNEL PDR 3 128 BLKS, READ-WRITE
13087 043510 012767 077406 134070  MOV      #77406,UIPDR3 ;MAP USER PDR 3 128 BLKS, READ-WRITE
13088 043516 012700 060000          MOV      #60000,R0    ;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
13089 043522 012701 100000          MOV      #100000,R1   ;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
13090 043526 012703 100011          MOV      #100011,R3   ;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
13091 043532 012702 077400          MOV      #77400,R2   ;LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
13092 043536 012767 043572 134504 21:  MOV      #51,MMVEC    ;POINT MEM. MGMT. TRAP VECTOR TO 51 BELOW
13093 043544 010267 126540          MOV      R2,KIPDR4   ;LOAD ACF TEST VALUE INTO KIPDR4
13094 043550 010267 134034          MOV      R2,UIPDR4   ;LOAD ACF TEST VALUE INTO UIPDR4
13095 043554 005010          CLR      (R0)        ;CLEAR PHYS. LOC. 140000 USING PDR3
13096 043556 016767 134214 174006  MOV      PSW,#TMP0    ;SAVE PSW IN CASE OF ERROR
13097 043564 005211          41:  INC      (R1)        ;TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 51
13098 043566 001462          BEQ     TS402
(3) 043570 104000          EMT          ;MEM. MGMT. ABORT DID NOT OCCUR
13099                                     ;FOR TIGHTER SCOPE LOOP
13100                                     ;REPLACE ERROR CALL WITH
13101                                     ;"BR 31" = 000772
13102 043572 062706 000004 51:  ADD      #4,SP        ;RESTORE STACK POINTER
13103 043576 005710          TST     (R0)         ;DID INSTRUCTION GET ABORTED & NOT EXECUTE
13104 043600 001401          BEQ     61
(2) 043602 104000          EMT          ;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
13105                                     ;FOR TIGHTER SCOPE LOOP
13106                                     ;REPLACE ERROR CALL WITH
13107                                     ;"BR 31" = 000764
13108 043604 016767 133762 173752 61:  MOV      SRO,WASSRO  ;READ STATUS REGISTER 0

```

H.F.

13109	043612	016767	133760	173746		MOV	SR2,WASSR2	;READ STATUS REGISTER 2
13110	043620	020367	173740			CMP	R3,WASSR0	;DID SRO REPORT NON RESIDENT ERROR CORRECTLY?
13111	043624	001401				BEQ	7:	
(2)	043626	104000				EMT		;SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
13112								;FOR TIGHTER SCOPE LOOP
13113								;REPLACE ERROR CALL WITH
13114								; "BR 3:" = 000752
13115	043630	012704	043564		7:	MOV	#4:,R4	;LOAD R4 WITH WHAT SR2 SHOULD READ
13116	043634	020467	173726			CMP	R4,WASSR2	;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4:)?
13117	043640	001401				BEQ	8:	
(2)	043642	104000				EMT		;SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
13118								;FOR TIGHTER SCOPE LOOP
13119								;REPLACE ERROR CALL WITH
13120								; "BR 3:" = 000744
13121	043644	042767	160000	133720	8:	BIC	#160000,SRO	;CLEAR THE ERROR BITS IN SRO
13122	043652	032767	140000	173712		BIT	#140000,#TMP0	;HAS ACF=0E4 BEEN TESTED IN USER YET
13123	043660	001006				BNE	9:	;BRANCH IF YES
13124	043662	012703	100151			MOV	#100151,R3	;LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG. 4
13125	043666	012767	140000	134102		MOV	#140000,PSW	;GO TO USER MODE
13126	043674	000720				BR	2:	;REPEAT TEST IN USER MODE
13127	043676	022702	077404		9:	CMP	#77404,R2	;HAS ACF=4 BEEN TESTED YET?
13128	043702	001407				BEQ	10:	;BRANCH IF YES
13129	043704	012702	077404			MOV	#77404,R2	;THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
13130	043710	012703	100011			MOV	#100011,R3	;LOAD R3 WITH WHAT SRO SHOULD READ-N.R.,KERNEL,PG. 4
13131	043714	005067	134056			CLR	PSW	;GO BACK TO KERNEL MODE
13132	043720	000706				BR	2:	;GO BACK & TEST ACF=4 IN SAME MODE
13133	043722	005067	134050		10:	CLR	PSW	;GO BACK TO KERNEL MODE BEFORE LEAVING
13134	043726	012767	021550	134314		MOV	#T0250,MMVEC	;RESTORE ADDRESS OF NORMAL MEMORY
13135								;MANAGEMENT ERROR ROUTINE TO MMVEC
13136								
13145								;*****
(2)								;TEST 402 READ-ONLY ABORT TEST (ACF=2)
(3)								;*****
(2)	043734							TS402:
13146	043734							1:
13147								;KERNEL & USER PAR'S 3 & 4 AND PDR 3
13148	043734	012700	060000			MOV	#60000,R0	;ARE SETUP FROM LAST TEST
13149	043740	012701	100000			MOV	#100000,R1	;LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
13150	043744	012703	020011			MOV	#20011,R3	;LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
13151	043750	012702	077402			MOV	#77402,R2	;LOAD R3 WITH WHAT SRO SHOULD READ - R/O, KERNEL, PG. 4
13152	043754	012767	044006	134266	2:	MOV	#5:,MMVEC	;LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
13153	043762	010267	126322			MOV	R2,KIPDR4	;POINT MEM. MGMT. TRAP VECTOR TO 5: BELOW
13154	043766	010267	133616			MOV	R2,UIPDR4	;LOAD ACF=2 INTO KIPDR4
13155	043772	005010			3:	CLR	(R0)	;LOAD ACF=2 INTO UIPDR4
13156	043774	016767	133776	173570		MOV	PSW,#TMP0	;CLEAR PHYS. LOC. 140000 USING PDR3
13157	044002	005211			4:	INC	(R1)	;SAVE PSW IN CASE OF ERROR
13158	044004	104000				EMT		;TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5:
13159								;MEM. MGMT. ABORT DID NOT OCCUR
13160								;FOR TIGHTER SCOPE LOOP
13161								;REPLACE ERROR CALL WITH
13162	044006	062706	000004		5:	ADD	#4,SP	; "BR 3:" = 000772
13163	044012	005710				TST	(R0)	;RESTORE STACK POINTER
13164	044014	001401				BEQ	6:	;DID INSTRUCTION GET ABORTED & NOT EXECUTE
(2)	044016	104000				EMT		
13165								;INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
13166								;FOR TIGHTER SCOPE LOOP
								;REPLACE ERROR CALL WITH

```

13167
13168 044020 016767 133546 173536 68: MOV SRO,WASSRO ;"BR 38" = 000764
13169 044026 016767 133544 173532 MOV SR2,WASSR2 ;READ STATUS REG. 0
13170 044034 020367 173524 CMP R3,WASSRO ;READ STATUS REG. 2
13171 044040 001401 BEQ 75 ;DID SRO REPORT READ-ONLY ERROR CORRECTLY?
(2) 044042 104000 EMT ;SRO DID NOT REPORT R/O ERROR CORRECTLY
13172 ;FOR TIGHTER SCOPE LOOP
13173 ;REPLACE ERROR CALL WITH
13174 ;"BR 38" = 000752
13175 044044 012704 044002 78: MOV #4,R4 ;LOAD R4 WITH WHAT SR2 SHOULD READ
13176 044050 020467 173512 CMP R4,WASSR2 ;DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4)?
13177 044054 001401 BEQ 88
(2) 044056 104000 EMT ;SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
13178 ;FOR TIGHTER SCOPE LOOP
13179 ;REPLACE ERROR CALL WITH
13180 ;"BR 38" = 000744
13181 044060 042767 160000 133504 88: BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
13182 044066 032767 140000 173476 BIT #140000,#TMP0 ;HAS ACF=2 BEEN TESTED IN USER MODE?
13183 044074 001006 BNE 98 ;BRANCH IF YES
13184 044076 012703 020151 MOV #20151,R3 ;LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4
13185 044102 012767 140000 133666 MOV #140000,PSW ;GO TO USER MODE
13186 044110 000721 BR 28 ;REPEAT TEST IN USER MODE
13187 044112 005067 133660 98: CLR PSW ;GO BACK TO KERNEL MODE BEFORE LEAVING
13188 044116 012767 021550 134124 MOV #T0250,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
13189 ;MANAGEMENT ERROR ROUTINE TO MMVEC.
13190
13191
13192
13193
13194
13195
13196
13197
13198
13199
13200
13201
13202
13203
13204
13205
13206
13207
13208
13209
13210
13211
13212
13213
13214
13215
13216
13217
13218
13219
13231

```

;NOTE: MACRO MSG31A WAS DELETED AS IT DIDN'T APPLY TO F11.

```

*****
;
; THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
; COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
; AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
; PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
; ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
; EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
; "PAGE LENGTH ABORT" WHILE THE OTHER TWO WON'T.
;
; STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
; ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
; THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
; IS LOCKED UP.
;
*****

```

```
13232 ;*****  
(2) ;TEST 403 PAGE LENGTH FAULTS UPWARD EXPANSION  
(3) ;*****  
(2) 044124 TS403:  
13233 044124 012767 077406 126154 1$: MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W  
13234 044132 012767 077406 126152 MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W  
13235 044140 012700 044340 MOV #DALTB1,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.  
13236 044144 012704 044356 MOV #PDRTB1,R4 ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).  
13237 044150 012701 000006 MOV #6,R1 ;SET UP LOOP COUNTER.  
13238 044154 012767 044316 134066 MOV #9$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS  
13239 044162 012706 001000 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP  
13240  
13241 ;TEST NON-ABORT CASES (VBA < OR = PLF)  
13242 044166 012467 126116 2$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
13243 044172 005730 TST B(R0)+ ;ACCESS VIRTUAL ADDR. (VBA < OR = PLF)  
13244 ;NO ABORT SHOULD OCCUR!!!  
13245 044174 077104 SOB R1,2$ ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.  
13246  
13247 ;TEST ABORT CASES (VBA > PLF)  
13248 044176 012701 000005 3$: MOV #5,R1 ;SET UP LOOP COUNTER.  
13249 044202 012700 044374 MOV #DALTB2,R0 ;DAL TABLE  
13250 044206 012704 044410 MOV #PDRTB2,R4 ;PDR TABLE  
13251 044212 012767 044232 134030 MOV #6$,MMVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT  
13252  
13253 044220 012467 126064 4$: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE  
13254 044224 005730 5$: TST B(R0)+ ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)  
13255 044226 001476 BEQ TS404  
(3) 044230 104000 EMT ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR  
13256 ;FOR TIGHTER SCOPE LOOP  
13257 ;REPLACE ERROR CALL WITH  
13258 ;"BR 5$" = 000776  
13259 044232 012706 001000 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT  
13260 044236 016767 133330 173320 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0  
13261 044244 016767 133326 173314 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2  
13262 044252 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2  
13263 044256 020267 173302 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?  
13264 044262 001401 BEQ 7$  
(2) 044264 104000 EMT ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY  
13265 ;FOR TIGHTER SCOPE LOOP  
13266 ;REPLACE ERROR CALL WITH  
13267 ;"BR 5$" = 000757  
13268 044266 012703 044224 7$: MOV #5$,R3 ;PUT EXPECTED SR2 CONTENTS IN R3  
13269 044272 020367 173270 CMP R3,WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?  
13270 044276 001401 BEQ 8$  
(2) 044300 104000 EMT ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY  
13271 ;FOR TIGHTER SCOPE LOOP  
13272 ;REPLACE ERROR CALL WITH  
13273 ;"BR 5$" = 000751  
13274 044302 042767 160000 133262 8$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO  
13275 044310 077135 SOB R1,4$ ;DONE?...NO - GET NEXT DAL & PDR PAIR  
13276 044312 000167 000010 JMP 10$ ;YES...  
13277 044316 042767 160000 133246 9$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO  
13278 044324 104000 EMT ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED  
13279 ;FOR TIGHTER SCOPE LOOP  
13280 ;REPLACE ERROR CALL WITH  
13281 ;A "NOP" = 240
```

```
13282
13283 044326 012767 021550 133714 10#: MOV #T0250,MMVEC ;RESTORE NORMAL M.M. TRAP HANDLER
13284 ;ADDRESS TO M.M. TRAP VECTOR
13285 044334 000167 J00064 JMP TS404 ;GET TO NEXT TEST
13286
13287 ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
13288 044340 100000 DALTB1: 100000
13289 044342 106100 106100
13290 044344 102300 102300
13291 044346 102500 102500
13292 044350 113700 113700
13293 044352 104600 104600
13294 044354 117700 117700
13295
13296 ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
13297 044356 000006 PDRTB1: 000006
13298 044360 052006 052006
13299 044362 045006 045006
13300 044364 052006 052006
13301 044366 074406 074406
13302 044370 025006 025006
13303 044372 077406 077406
13304
13305 ;DAL TABLE (ABORT CASES)
13306 044374 100100 DALTB2: 100100
13307 044376 110100 110100
13308 044400 116600 116600
13309 044402 112700 112700
13310 044404 117000 117000
13311 044406 117700 117700
13312
13313 ;PDR TABLE (ABORT CASES)
13314 044410 000006 PDRTB2: 000006
13315 044412 030406 030406
13316 044414 046406 046406
13317 044416 042006 042006
13318 044420 073406 073406
13319 044422 077006 077006
13320
13333 ;*****
(2) ;TEST 404 PAGE LENGTH FAULTS-DOWNWARD EXPANSION
(3) ;*****
(2) 044424 TS404:
13334 044424 012700 044624 1#: MOV #DALTB3,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
13335 044430 012704 044642 MOV #PDRTB3,R4 ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
13336 044434 012701 000006 MOV #6,R1 ;SET UP LOOP COUNTER.
13337 044440 012767 044602 133602 MOV #9#,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
13338 044446 012706 001000 MOV #KERSTK,KSP ;MAKE SURE STACK POINTER IS ALL SET UP
13339
13340 ;TEST NON-ABORT CASES (VBA > OR = PLF)
13341 044452 012467 125632 2#: MOV (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
13342 044456 005730 TST B(R0)+ ;ACCESS VIRTUAL ADDR. (VBA > OR = PLF)
13343 ;NO ABORT SHOULD OCCUR!!!
13344 044460 077104 SOB R1,2# ;DONE?...NO- TEST NEXT COMBINATION OF DAL & PDR.
13345
```



```

3346 ;TEST ABORT CASES (VBA < PLF)
3347 044462 012701 000005 3$: MOV #5,R1 ;SET UP LOOP COUNTER.
3348 044466 012700 044660 MOV #DALTB4,R0 ;DAL TABLE
3349 044472 012704 044674 MOV #PDRTB4,R4 ;PDR TABLE
3350 044476 012767 044516 133544 MOV #6$,MVEC ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
3351
3352 044504 012467 125600 4$: MOV (R4),KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGHT VALUE
3353 044510 005730 5$: TST B(R0) ;ACCESS VIRTUAL ADDR. (VBA < PLF - ABORT TO 6$)
3354 044512 001476 BEQ TS405
(3) 044514 104000 EMT ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
3355 ;FOR TIGHTER SCOPE LOOP
3356 ;REPLACE ERROR CALL WITH
3357 ;"BR 5$" = 000776
3358 044516 012706 001000 6$: MOV #KERSTK,KSP ;RESTORE STACK POINTER FOLLOWING ABORT
3359 044522 016767 133044 173034 MOV SRO,WASSRO ;READ M.M. STATUS REG. 0
3360 044530 016767 133042 173030 MOV SR2,WASSR2 ;READ M.M. STATUS REG. 2
3361 044536 012702 040011 MOV #40011,R2 ;PUT EXPECTED SRO CONTENTS IN R2
3362 044542 020267 173016 CMP R2,WASSRO ;DID SRO REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
3363 044546 001401 BEQ 7$
(2) 044550 104000 EMT ;SRO DID NOT REPORT PG. LENGTH ABORT CORRECTLY
3364 ;FOR TIGHTER SCOPE LOOP
3365 ;REPLACE ERROR CALL WITH
3366 ;"BR 5$" = 000757
3367 044552 012703 044510 7$: MOV #5$,R3 ;PUT EXPECTED SR2 CONTENTS IN R3
3368 044556 020367 173004 CMP R3,WASSR2 ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
3369 044562 001401 BEQ 8$
(2) 044564 104000 EMT ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
3370 ;FOR TIGHTER SCOPE LOOP
3371 ;REPLACE ERROR CALL WITH
3372 ;"BR 5$" = 000751
3373 044566 042767 160000 132776 8$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
3374 044574 077135 SOB R1,4$ ;DONE?...NO - GET NEXT DAL & PDR PAIR
3375 044576 000167 000010 JMP 10$ ;YES...
3376 044602 042767 160000 132762 9$: BIC #160000,SRO ;CLEAR ERROR BITS IN SRO
3377 044610 104000 EMT ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
3378 ;FOR TIGHTER SCOPE LOOP
3379 ;REPLACE ERROR CALL WITH
3380 ;A "NOP" = 000240
3381
3382 044612 012767 021550 133430 10$: MOV #T0250,MVEC ;RESTORE NORMAL M.M. TRAP HANDLER
3383 ;ADDRESS TO M.M. TRAP VECTOR
3384 044620 000167 000064 JMP TS405 ;GET TO NEXT TEST
3385
3386 ;DAL TABLE FOR DOWNWARD EXPANSION (NON-ABORT CASES)
3387 044624 117700 DALTB3: 117700
3388 044626 111600 111600
3389 044630 115400 115400
3390 044632 115200 115200
3391 044634 104000 104000
3392 044636 113100 113100
3393 044640 100000 100000
3394
3395 ;PDR TABLE (NON-ABORT CASES)
3396 044642 077416 PDRTB3: 77416
3397 044644 025416 25416
3398 044646 032416 32416

```

3399 044650 025416
3400 044652 003016
3401 044654 052416
3402 044656 000016

25416
J3016
52416
00016

;DAL TABLE (ABORT CASES)

DALTBA: 117600
107600
101100
105000
100700
100000

;PDR TABLE (ABORT CASES)

PDRTB4: 77416
47016
31016
35416
04016
00416

3403
3404
3405 044660 117600
3406 044662 107600
3407 044664 101100
3408 044666 105000
3409 044670 100700
3410 044672 100000

3411
3412
3413 044674 077416
3414 044676 047016
3415 044700 031016
3416 044702 035416
3417 044704 004016
3418 044706 000416

3419
3420
3434
3435

;TEST 405 SR2 BIT TEST

(2)

(3)

(2) 044710

3436 044710 012767 001400 125430

3437 044716 012767 001400 125424

3438 044724 012767 077406 125354

3439 044732 012767 077402 125350

3440 044740 012700 060002

3441 044744 012701 100002

3442 044750 012767 044776 133272

3443 044756 012720 010727

3444 044762 005020

3445 044764 012720 000137

3446 044770 012710 044776

3447 044774 010107

3448 044776 012706 001000

3449 045002 016767 132570 172556

3450 045010 020167 172552

3451 045014 001401

(2) 045016 104000

3452

3453

3454

3455 045020 042767 160000 132544

3456 045026 060101

3457 045030 010100

3458 045032 052701 100000

3459 045036 052700 060000

3460 045042 020127 110000

3461 045046 101743

3462

3463 045050 012767 077406 125232

TS405:
1\$: MOV #1400,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 24-28K
MOV #1400,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 24-28K
MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
MOV #60002,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
MOV #100002,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
MOV #3\$,MMVEC ;SET M.M. TRAP VECTOR TO 3\$
2\$: MOV #010727,(R0)+ ;LOAD "MOV PC,(PC)+ " INSTRUCTION AT ADDR.
CLR (R0)+ ; REACHED THRU PDR/PAR 4.
MOV #000137,(R0)+ ;LOAD "JMP @#3\$" INSTRUCTION AT VIRT. ADDR.
MOV #3\$,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO "PAGE 4 INSTRUCTIONS"
3\$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
CMP R1,WASSR2 ;WAS ADDR. OF "RELOCATED - R/O ABORT" LOCKED UP?
4\$: BEQ 4\$
;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2\$" = 000757
4\$: BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
ADD R1,R1 ;SETUP TO FORM NEXT VIRTUAL ADDRESS
MOV R1,R0 ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD
BIS #100000,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
BIS #60000,R0 ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD
CMP R1,#110000 ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?
BLOS 2\$;BRANCH IF NO
MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS

```

13464 045056 012767 021550 133164      MOV      @T0250,MMVEC      ;RESTORE ADDRESS OF NORMAL M.M.
13465                                     ;TRAP HANDLER TO M.M. VECTOR
13466
13480
13481                                     ;*****
(2)                                     ;TEST 406      MORE CHECKS OF SRO & SR2
(3)                                     ;*****
(2) 045064      TS406:
13482 045064 012767 001400 125260 18:      MOV      @1400,KIPAR5      ;MAP KERNEL PAGE 5 TO 24-28K
13483 045072 012767 000406 125210      MOV      @406,KIPDR4      ;SETUP PDR4 FOR PAGE LENGTH ABORT
13484 045100 012767 077402 125204      MOV      @77402,KIPDR5    ;SETUP PDR5 FOR R/O ABORT
13485 045106 016767 132464 172452 28:      MOV      SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
13486 045114 012701 045106      MOV      @28,R1           ;PUT EXPECTED VIRTUAL PC IN R1
13487 045120 020167 172442      CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 28?
13488 045124 001401      BEQ      48
(2) 045126 104000      EMT
13489                                     ;SR2 NOT TRACKING CORRECTLY
13490                                     ;FOR TIGHTER SCOPE LOOP
13491                                     ;REPLACE ERROR CALL WITH
13492 045130 016767 132442 172430 48:      MOV      SR2,WASSR2      ;READ SR2 TO SEE IF ITS TRACKING
13493 045136 012701 045130      MOV      @48,R1           ;PUT EXPECTED VIRTUAL PC IN R1
13494 045142 020167 172420      CMP      R1,WASSR2      ;DID SR2 CONTAIN VIRTUAL PC AT 48?
13495 045146 001401      BEQ      68
(2) 045150 104000      EMT
13496                                     ;SR2 NOT TRACKING CORRECTLY
13497                                     ;FOR TIGHTER SCOPE LOOP
13498                                     ;REPLACE ERROR CALL WITH
13499 045152 012767 045170 133070 68:      MOV      @78,MMVEC      ;PUT ADDRESS OF 78 IN M.M. TRAP VECTOR
13500 045160 005067 172410      CLR      $TMP1           ;CLEAR ERROR INDICATOR
13501 045164 005237 100500      INC      @0100500        ;CAUSE PAGE LENGTH ABORT - TRAP TO 78
13502 045170 012706 001000      MOV      @KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
13503 045174 016767 132372 172370 78:      MOV      SRO,$TMP0      ;SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
13504 045202 016767 132370 172366      MOV      SR2,$TMP2      ;SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
13505 045210 012767 045222 133032      MOV      @88,MMVEC      ;PUT ADDRESS OF 88 IN M.M. TRAP VECTOR
13506 045216 005237 120000      INC      @0120000        ;CAUSE R/O ABORT - TRAP TO 88
13507 045222 012706 001000      MOV      @KERSTK,KSP     ;RESTORE STACK POINTER AFTER ABORT
13508 045226 016767 132340 172330 88:      MOV      SRO,WASSR0      ;READ SRO FOLLOWING SECOND KT ABORT
13509 045234 016767 132336 172324      MOV      SR2,WASSR2      ;READ SR2 FOLLOWING SECOND KT ABORT
13510 045242 026767 172324 172314      CMP      $TMP0,WASSR0    ;IS SRO STILL HOLDING INFO ON FIRST ABORT?
13511 045250 001402      BEQ      98
13512 045252 005267 172316      INC      $TMP1           ;BRANCH IF YES
13513 045256 026767 172314 172302 98:      CMP      $TMP2,WASSR2    ;SET ERROR INDICATOR
13514 045264 001402      BEQ      108             ;DOES SR2 STILL HOLD PC OF FIRST ABORT?
13515 045266 005267 172302      INC      $TMP1           ;BRANCH IF YES
13516 045272 005767 172276      TST     $TMP1           ;SET ERROR INDICATOR
13517 045276 001401      BEQ      118             ;WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
(2) 045300 104000      EMT
13518                                     ;ONE OF STATUS REGS. CHANGED BY SECOND ABORT
13519                                     ;FOR TIGHTER SCOPE LOOP
13520                                     ;REPLACE ERROR CALL WITH
13521 045302 005067 172276 118:      CLR      $TMP1           ;"BR 68" = 000726
13522 045306 000005      RESET                    ;CLEAR ERROR INDICATOR
13523 045310 005067 132256      CLR      SRO            ;EXECUTE A RESET, APPLYING AN "INIT"
13524 045314 016767 132252 172242      MOV      SRO,WASSR0      ;READ SRO
13525 045322 005767 172236      TST     WASSR0          ;WAS SRO CLEARED BY THE RESET?
13526 045326 001402      BEQ      128             ;BRANCH IF YES

```

13527	045330	005267	172240			INC	\$TMP1	;SRO NOT CLEARED BY A RESET	
13528	045334	016767	132236	172224	12:	MOV	SR2,WASSR2	;READ SR2	
13529	045342	022767	045334	172216		CHP	#12,WASSR2	;WAS SR2 UNLOCKED BY A RESET?	
13530	045350	001402				BEQ	13:	;BRANCH IF YES	
13531	045352	005267	172216			INC	\$TMP1	;SR2 NOT UNLOCKED BY A RESET	
13532	045356	005767	172212		13:	TST	\$TMP1	;WERE SRO & SR2 BOTH "RESET" BY A RESET?	
13533	045362	001401				BEQ	14:		
(2)	045364	104000				EMT		;SRO OR SR2 NOT "RESET" BY A RESET	
13534								;FOR TIGHTER SCOPE LOOP	
13535								;REPLACE ERROR CALL WITH	
13536								; "BR 6" = 000676	
13537	045366	012767	000001	132176	14:	MOV	#1,SRO	;TURN MEMORY MANAGEMENT BACK ON	
13538	045374	016767	132176	172164	15:	MOV	SR2,WASSR2	;READ SR2 TO SEE IF ITS TRACKING AGAIN	
13539	045402	012701	045374			MOV	#15,R1	;PUT EXPECTED VIRTUAL PC IN R1	
13540	045406	020167	172154			CHP	R1,WASSR2	;DID SR2 CONTAIN VIRTUAL PC AT 15:	
13541	045412	001401				BEQ	16:		
(2)	045414	104000				EMT		;SR2 NOT TRACKING CORRECTLY	
13542								;FOR TIGHTER SCOPE LOOP	
13543								;REPLACE ERROR CALL WITH	
13544								; "BR 6" = 000663	
13545	045416	012767	077406	124664	16:	MOV	#77406,KIPDR4	;RESET PDR4 TO 128 BLKS, R/W	
13546	045424	012767	077406	124660		MOV	#77406,KIPDR5	;RESET PDR5 TO 128 BLKS, R/W	
13547	045432	012767	021550	132610		MOV	#T0250,MMVEC	;RESTORE ADDRESS OF NORMAL MEMORY	
13548								;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR	
13549									
13561									
13562									
(2)									
(3)									
(2)	045440								
13563	045440	004767	002614			1:	JSR	PC,TOFF	;TURN OFF T-BIT TRAPPING FOR THIS TEST
13564	045444	005067	132326			2:	CLR	PSW	;GO TO KERNEL MODE
13565	045450	012706	001000				MOV	#KERSTK,KSP	;SETUP KERNEL STACK PTR.
13566	045454	012767	001400	132156			MOV	#1400,UIPARO	;MAP USER PAGE 0 TO 24K
13567	045462	012737	045532	000004			MOV	#4,B04	;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4:
13568	045470	012737	000340	000006			MOV	#340,B06	;LOAD VECTOR+2 WITH NEW PSW
13569	045476	012767	140000	132272			MOV	#140000,PSW	;GO TO USER MODE
13570	045504	012706	000600				MOV	#USESTK,USP	;SETUP USER STACK PTR.
13571	045510	012737	045530	000004			MOV	#3,B04	;LOAD USER VECTOR 4 (LOC. 60004) WITH 3:
13572	045516	012737	000340	000006			MOV	#340,B06	;LOAD VECTOR+2 WITH NEW PSW
13573	045524	005767	112250				TST	160000	;CAUSE TIMEOUT ERROR TRAP TO "4"
13574									;SHOULD PICK UP NEW PC=4: FROM KERNEL
13575									;LOC. 4, NOT PC=3: FROM USER LOC. 4 (~60004)
13576	045530				3:				
13577	045530	104000				EMT		;DID NOT TRAP THRU KERNEL SPACE	
13578								;FOR TIGHTER SCOPE LOOP	
13579								;REPLACE ERROR CALL WITH	
13580								; "BR 2" = 000740	
13581	045532	005067	132240		4:	CLR	PSW	;BE SURE BACK IN KERNEL MODE	
13582	045536	012706	001000			MOV	#KERSTK,KSP	;RESTORE KERNEL S.P. IN CASE IT CHANGED	
13583	045542	005067	132072			CLR	UIPARO	;REMAP USER PAGE 0 TO 0-4K	
13584	045546	012767	140000	132222		MOV	#140000,PSW	;GO TO USER MODE	
13585	045554	012706	000600			MOV	#USESTK,USP	;RESTORE USER STACK POINTER	
13586	045560	005067	132212			CLR	PSW	;GO BACK TO KERNEL MODE	
13587	045564	012737	021526	000004		MOV	#T04,B04	;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4	
13588	045572	004767	002516			JSR	PC,TON	;TURN T-BIT TRAPPING BACK ON	

;TEST 407 USER ABORT PICKS UP KERNEL SPACE VECTOR
;*****

TS407:

01

```
13589
13596 (2)
13597 (3)
13598 (2) 045576
13598 045576 012702 170000
13599 045602 010267 132170
13600 045606 012746 000340
13601 045612 012746 045620
13602 045616 000002
13603 045620 016701 132152
13604 045624 042701 007437
13605 045630 005067 132142
13606 045634 020201
13607 045636 001401
13608 (3) 045640 104000
13609
13610
13611
13623
13624 (2)
13625 (3)
13626 (2) 045642
13625 045642 012705 077006
13626 045646 010567 124444
13627 045652 012737 045672 000004
13628 045660 012737 045674 000250
13629 045666 005237 177700
13630 045672
13631 045672 104000
13632
13633
13634
13635 045674 012706 001000
13636 045700 016767 131666 171656
13637 045706 016767 131664 171652
13638 045714 012700 040017
13639 045720 020067 171640
13640 045724 001401
13641 (2) 045726 104000
13642
13643
13644 045730 012701 045666
13645 045734 020167 171626
13646 045740 001401
13647 (2) 045742 104000
13648
13649
13650 045744 042767 160000 131620
13651 045752 012737 021526 000004
13652 045760 012737 021550 000250
```

```

;*****
;TEST 410 RTI IN USER MODE DOES NOT CHANGE PSW
;*****
TS410:
2$: MOV #170000,R2 ;LOAD "PRESENT & EXPECTED" PSW VALUE INTO R2
MOV R2,PSW ;GO TO USER MODE-PRIORITY 0
MOV #340,-(SP) ;PUT A NEW PSW (PRIORITY=7) ON STACK
MOV #3$,-(SP) ;PUT NEW PC ON THE STACK
RTI ;DO AN RTI FROM USER MODE
3$: MOV PSW,R1 ;READ NEW PSW INTO R1
BIC #7437,R1 ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
CLR PSW ;GO BACK TO KERNEL MODE
CMP R2,R1 ;DID PSW STAY IN USER, PRIORITY=0?
BEQ TS411
EHT ;PSW CHANGED BY AN RTI FROM USER
;FOR A TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR=2$" = 000760

;*****
;TEST 411 KT ERROR SERVICED BEFORE TIMEOUT ERROR
;*****
TS411:
1$: MOV #77006,R5 ;LOAD PDR7 DATA INTO R5
MOV R5,KIPDR7 ;MAP PAGE 7 R/W PLF=176
MOV #3$,R04 ;SET CPU TRAP VECTOR TO ADDRESS OF 3$
MOV #4$,R0250 ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$
2$: INC @R177700 ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT
3$: EHT ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000776
4$: MOV #KERSTK,KSP ;RESTORE STACK POINTER AFTER TRAPPING
MOV SRO,WASSR0 ;READ STATUS REG. 0
5$: MOV SR2,WASSR2 ;READ STATUS REG. 2
MOV #40017,R0 ;LOAD EXPECTED SRO CONTENTS INTO R0
CMP R0,WASSR0 ;SRO PLF ERROR BIT SET?
BEQ 6$
EHT ;SRO DIDN'T REPORT PLF ERROR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000741
6$: MOV #2$,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1
CMP R1,WASSR2 ;WAS SR2 LOCKED BY PLF ABORT?
BEQ 7$
EHT ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;"BR 2$" = 000741
7$: BIC #160000,SRO ;CLEAR ERROR BITS THAT WERE SET IN SRO
MOV #T04,R04 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
MOV #T0250,R0250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
```

13653 045766 012767 077406 124322
 13654
 13669
 13670
 (2)
 (3)
 (2) 045774
 13671 045774 004767 002260
 13672 046000 012767 001400 131640
 13673 046006 012767 001400 131634
 13674 046014 012767 077402 131564
 13675 046022 012767 077406 131560
 13676 046030 012737 046076 000004
 13677 046036 012737 140017 000006
 13678 046044 012737 046076 000250
 13679 046052 012737 000340 000252
 13680 046060 012767 140000 131710
 13681 046066 012706 100002
 13682 046072 005737 177700
 13683
 13684 046076 016601 000002
 13685 046102 011603
 13686 046104 016767 131462 171452
 13687 046112 016767 131460 171446
 13688 046120 042767 160000 131444
 13689 046126 005067 131644
 13690 046132 012706 001000
 13691 046136 012767 140000 131632
 13692 046144 012706 000600
 13693 046150 005067 131622
 13694 046154 005067 171412
 13695 046160 020127 170017
 13696
 13697
 13698
 13699 046164 001402
 13700 046166 005267 171400
 13701 046172 020327 046076
 13702
 13703 046176 001402
 13704 046200 005267 171366
 13705 046204 026727 171354 020147
 13706 046212 001402
 13707 046214 005267 171352
 13708 046220 026727 171342 046072
 13709
 13710 046226 001402
 13711 046230 005267 171336
 13712 046234 005767 171332
 13713 046240 001401
 (2) 046242 104000
 13714
 13715
 13716
 13717
 13718

```

MOV #77406,KIPDR7 ;REMAP PAGE 7 TO READ/WRITE PLF=177

;*****
;TEST 412 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR
;*****
TS412:
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
MOV #1400,UIPAR3 ;MAP USER PAGE 3 TO 24-28K
MOV #1400,UIPAR4 ;MAP USER PAGE 4 TO 24-28K
MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
MOV #4$,B#4 ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
MOV #140017,B#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR*2
MOV #4$,B#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
MOV #340,B#252 ;LOAD A KERNEL PSW IN M#VEC*2
2$: MOV #140000,PSW ;GO TO USER MODE
MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
3$: TST B#177700 ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
;R/O ERROR WHEN TRY TO SAVE OLD PC
4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2
BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
CLR PSW ;BE SURE IN KERNEL MODE
MOV #KERSTK,KSP ;RESTORE KERNEL STACK POINTER
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;RESTORE USER STACK POINTER
CLR PSW ;GO BACK TO KERNEL MODE
CLR $TMP0 ;CLEAR ERROR INDICATOR
CMP R1,#170017 ;WAS THE PSW SAVED THE ONE PICKED UP BY THE
;TIMEOUT TRAP FROM ERRVEC*2?
;VALUE 170017 = PSW FROM LOC. 6 WITH
;PREVIOUS MODE BITS = USER
5$: BEQ 5$ ;BRANCH IF YES
INC $TMP0 ;WRONG PSW SAVED DURING "DOUBLE ERROR" SEQUENCE
CMP R3,#3$.4 ;WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
;SAVED ON THE STACK?
6$: BEQ 6$ ;BRANCH IF YES
INC $TMP0 ;WRONG PC SAVED DURING TRAP SEQUENCE
CMP WASSRO,#20147 ;DID SRO REPORT - USER, PAGE 3, R/O ABORT?
7$: BEQ 7$ ;BRANCH IF YES
INC $TMP0 ;SRO DID NOT REPORT R/O ABORT
CMP WASSR2,#3$ ;DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
;INSTRUCTION SUCCESSFULLY FETCHED?
8$: BEQ 8$ ;BRANCH IF YES
INC $TMP0 ;SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
9$: TST $TMP0 ;ANY "ERRORS" DURING TRAP SEQUENCE?
BEQ 9$
EMT ;THE WRONG PC OR PSW WERE SAVED
;OR SRO OR SR2 DID NOT REPORT R/O
;ERROR DURING TIMEOUT - KT TRAP
;SEQUENCE
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH

```

```
13719 ;"BR 24" = 000710
13720 046244 012737 021526 000004 94: MOV #T04,B04 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
13721 046252 012737 000340 000006 MOV #340,B06 ;RELOAD ERRVEC+2 WITH KERNEL PSW
13722 046260 012737 021550 000250 MOV #T0250,B0250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
13723 046266 012767 077406 131312 MOV #77406,UIPDR3 ;REMAP USER PAGE 3 READ/WRITE
13724 046274 004767 002014 JSR PC,TON ;TURN T-BIT TRAPPING BACK ON
13725 ;*****
13726 ;*
13727 ;* THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
13728 ;* THE "MOVE FROM PREVIOUS" AND MOVE TO PREVIOUS" INSTRUCTIONS.
13729 ;*
13730 ;*****
13731
13732
13744 ;*****
(2) ;TEST 413 MOVE FROM PREVIOUS (USER) I-SPACE
(3) ;*****
(2) 046300 TS413:
13745 046300 005067 124034 14: CLR KIPAR0 ;MAP KERNEL PAGE 0 TO 0-4K
13746 046304 012767 000200 124030 MOV #200,KIPAR1 ;MAP KERNEL PAGE 1 TO 4-8K
13747 046312 012767 000400 124024 MOV #400,KIPAR2 ;MAP KERNEL PAGE 2 TO 8-12K
13748 046320 012767 000600 124020 MOV #600,KIPAR3 ;MAP KERNEL PAGE 3 TO 12-16K
13749 046326 012767 001400 124014 MOV #1400,KIPAR4 ;MAP KERNEL PAGE 4 TO 24-28K
13750 046334 012767 007600 124014 MOV #7600,KIPAR7 ;MAP KERNEL PAGE 7 TO THE I/O PAGE
13751 046342 012700 077406 MOV #77406,R0 ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
13752 ;READ/WRITE, LENGTH 200 BLOCKS
13753 046346 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
13754 046352 012701 172300 MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
13755 046356 010021 24: MOV R0,(R1)+ ;LOAD PDR WITH 77406
13756 046360 077202 SOB R2,24 ;LOOP TO 24 UNTIL ALL PDRS LOADED
13757 046362 012702 000010 MOV #10,R2 ;SET LOOP COUNTER TO 8
13758 046366 012701 177600 MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
13759 046372 010021 34: MOV R0,(R1)+ ;LOAD PDR WITH 77406
13760 046374 077202 SOB R2,34 ;LOOP TO 34 UNTIL ALL PDRS LOADED
13761 046376 012767 000000 131234 MOV #000,UIPAR0 ;MAP USER I PAGE 0 TO 0-4K
13762 046404 012767 000200 131230 MOV #200,UIPAR1 ;MAP USER I PAGE 1 TO 4-8K
13763 046412 012767 000400 131224 MOV #400,UIPAR2 ;MAP USER I PAGE 2 TO 8-12K
13764 046420 012767 000600 131220 MOV #600,UIPAR3 ;MAP USER I PAGE 3 TO 12-16K
13765 046426 012767 007600 131222 MOV #7600,UIPAR7 ;MAP USER I PAGE 7 TO THE I/O PAGE
13766 046434
13767 046434 012767 077406 123646 44: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
13768 046442 012767 001400 123700 MOV #1400,KIPAR4 ;MAP KERNEL I PAGE 4 TO 24K
13769 046450 012767 001400 131172 MOV #1400,UIPAR4 ;MAP USER I PAGE 4 TO 24K
13770 046456 012700 036514 MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
13771 046462 010037 100000 MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 140000
13772 046466 012767 046764 131554 MOV #234,MVEC ;SET M.M. VECTOR TO 234
13773 046474 105067 123610 CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
13774 ;THE FOLLOWING WILL TEST DSTH=0 MFPI
13775 ;
13776 046500 012767 030340 131270 54: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
13777 046506 006506 64: MFPI USP ;PUT USER STACK POINTER ON KERNEL
13778 ;STACK
13779 046510 022706 001000 CMP #KERSTK,KSP ;WAS SOMETHING PUSHED ON STACK AT 64
13780 046514 001405 BEQ 74 ;BRANCH IF NOTHING WAS PUSHED
13781 046516 012600 MOV (KSP)+,R0 ;POP KERNEL STACK INTO R0
13782 046520 012701 000600 MOV #USESTK,R1 ;EXPECTING TO GET 700 AS USP
```

```
13783 046524 020001          CMP      R0,R1          ;DID YOU GET THE RIGHT POINTER?
13784 046526 001401          BEQ      8:
      (1) 046530          7:
      (2) 046530 104000          EMT
13785
13786
13787
13788 046532          8:          ;THE FOLLOWING WILL TEST DSTH=1 MFPI.
13789 046532 012700 036514          MOV      #36514,R0      ;RELOAD DATA PATTERN IN R0
13790 046536 012767 030340 131232 9:          MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13791 046544 012702 100000          MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
13792 046550 006512          MFPI     (R2)          ;READ FROM PHYSICAL 140000
13793 046552 012601          MOV      (KSP)+,R1     ;POP KERNEL STACK INTO R1
13794 046554 020001          CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
13795 046556 001401          BEQ      10:
      (2) 046560 104000          EMT
13796
13797
13798
13799 046562          10:         ;THE FOLLOWING WILL TEST DSTH=2 MFPI.
13800 046562 012767 030340 131206 11:        MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13801 046570 012702 100000          MOV      #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
13802 046574 006522          MFPI     (R2)+        ;READ FROM PHYSICAL 140000
13803 046576 012601          MOV      (KSP)+,R1     ;POP KERNEL STACK INTO R1
13804 046600 020001          CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
13805 046602 001401          BEQ      12:
      (2) 046604 104000          EMT
13806
13807
13808
13809 046606          12:         ;THE FOLLOWING WILL TEST DSTH=3 MFPI.
13810 046606 012767 030340 131162 13:        MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13811 046614 006537 100000          MFPI     #100000       ;READ FROM PHYSICAL 140000
13812 046620 012601          MOV      (KSP)+,R1     ;POP KERNEL STACK INTO R1
13813 046622 020001          CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
13814 046624 001401          BEQ      14:
      (2) 046626 104000          EMT
13815
13816
13817
13818 046630          14:         ;THE FOLLOWING WILL TEST DSTH=4 MFPI.
13819 046630 012767 030340 131140 15:        MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13820 046636 012702 100002          MOV      #100002,R2    ;LOAD VIRTUAL ADDRESS INTO R2
13821 046642 006542          MFPI     -(R2)        ;READ FROM PHYSICAL 140000
13822 046644 012601          MOV      (KSP)+,R1     ;POP KERNEL STACK INTO R1
13823 046646 020001          CMP      R0,R1        ;WAS DATA FETCHED SAME AS STORED
13824 046650 001401          BEQ      16:
      (2) 046652 104000          EMT
13825
13826
13827
13828 046654          16:         ;THE FOLLOWING WILL TEST DSTH=5 MFPI.
13829
13830
13831 046654 012767 030340 131114 17:        MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13832 046662 012767 100000 170706          MOV      #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
```


CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58 P11 07 JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 19
T413 MOVE FROM PREVIOUS (USER) I SPACE

SEQ 0215

13864
13865
13866
13867
13868

046766 012767 021550 131254 224: MOV @T0250,MMVEC

;REPLACE ERROR CALL WITH
;"BR 214" = 000762
;SET M.M. VECTOR TO NORMAL ROUTINE

```
13881 ;*****  
(2) ;TEST 414 MOVE TO PREVIOUS (USER) I-SPACE  
(3) ;*****  
(2) 046774 TS414:  
13882 046774 012767 077406 123306 1$: MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE  
13883 047002 012767 077406 130600 MOV #77406,UIPDR4 ;USER I-SPACE PAGE 4 READ/WRITE  
13884 047010 012767 001400 123332 MOV #1400,KIPAR4 ;MAP KERNEL I PAGE 4 TO 24K  
13885 047016 012767 001400 130624 MOV #1400,UIPAR4 ;MAP USER I PAGE 4 TO 24K  
13886 047024 012767 047516 131216 MOV #20$,MMVEC ;SET M.M. VECTOR TO 20$  
13887 ;THE FOLLOWING WILL TEST DSTH=0 MTPI  
13888 ;  
13889 047032 012767 030340 130736 2$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER  
13890 047040 012746 007777 MOV #7777,-(KSP) ;PUSH DATA ON KERNEL STACK  
13891 047044 006606 MTPI USP ;LOAD USER STACK POINTER  
13892 047046 006506 MFPI USP ;READ USER STACK POINTER  
13893 047050 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1  
13894 047052 022701 007777 CMP #7777,R1 ;WAS USER STACK POINTER CHANGED  
13895 047056 001401 BEQ 3$  
(2) 047060 104000 EMT ;USER STACK POINTER NOT CHANGED  
13896 ;FOR TIGHTER SCOPE LOOP  
13897 ;REPLACE ERROR CALL WITH  
13898 ;"BR 2$" = 000764  
13899 047062 012767 030340 130706 3$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER  
13900 047070 012746 000600 MOV #USESTK,-(KSP) ;GET READY TO RESTORE USER S. POINT  
13901 047074 006606 MTPI USP ;RESTORE USER STACK POINTER  
13902 047076 4$: ;THIS WILL TEST DSTH = 1 MTPI.  
13903 047076 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2  
13904 047102 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0  
13905 047106 010046 5$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK  
13906 047110 105067 123174 CLR8 KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT  
13907 047114 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 140000  
13908 047116 112767 000006 123164 MOV8 #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT  
13909 047124 011201 MOV (R2),R1 ;READ FROM ADDRESS 140000  
13910 047126 020001 CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE  
13911 047130 001401 BEQ 6$  
(2) 047132 104000 EMT ;INCORRECT STORE  
13912 ;FOR TIGHTER SCOPE LOOP  
13913 ;REPLACE ERROR CALL WITH  
13914 ;"BR 5$" = 000765  
13915 047134 6$: ;THE FOLLOWING WILL TEST DSTH=2 MTPI.  
13916 ;  
13917 047134 012767 030340 130634 MOV #030340,PSW ;MAKE PREVIOUS MODE USER  
13918 047142 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0  
13919 047146 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2  
13920 047152 010046 8$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK  
13921 047154 105067 123130 CLR8 KIPDR4 ;MAKE KERNEL PAGE 4 NON-RESIDENT  
13922 047160 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 140000  
13923 047162 112767 000006 123120 MOV8 #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT  
13924 047170 013701 100000 MOV #100000,R1 ;READ FROM ADDRESS 140000  
13925 047174 020001 CMP R0,R1 ;SEE IF DATA WAS STORED CORRECTLY  
13926 047176 001401 BEQ 9$  
(2) 047200 104000 EMT ;INCORRECT STORE  
13927 ;FOR TIGHTER SCOPE LOOP  
13928 ;REPLACE ERROR CALL WITH  
13929 ;"BR 8$" = 000764  
13930 047202 9$: ;THIS WILL TEST DSTH = 3 MTPI.
```

13931	047202	012767	030340	130566		MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
13932	047210	012700	052525			MOV	#52525,R0	;LOAD TEST DATA INTO R0
13933	047214	010046			10:	MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
13934	047216	105067	123066			CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
13935	047222	006637	100000			MTPI	#100000	;LOAD TEST DATA INTO PHYSICAL 140000
13936	047226	112767	000006	123054		MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
13937	047234	013701	100000			MOV	#100000,R1	;READ FROM ADDRESS 140000
13938	047240	020001				CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
13939	047242	001531				BEQ	TS415	
(3)	047244	104000				EMT		
13940	047246	001401				BEQ	11:	
(2)	047250	104000				EMT		;INCORRECT STORE
13941								;FOR TIGHTER SCOPE LOOP
13942								;REPLACE ERROR CALL WITH
13943								; "BR 10:" = 000763
13944	047252				11:		; THIS WILL TEST DSTM = 4	MTPI.
13945	047252	012767	030340	130516		MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
13946	047260	012700	125252			MOV	#125252,R0	;LOAD TEST DATA INTO R0
13947	047264	010046			12:	MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
13948	047266	012702	100002			MOV	#100002,R2	;LOAD VIRTUAL ADDRESS INTO R2
13949	047272	105067	123012			CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
13950	047276	006642				MTPI	-(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
13951	047300	112767	000006	123002		MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
13952	047306	013701	100000			MOV	#100000,R1	;READ FROM ADDRESS 140000
13953	047312	020001				CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
13954	047314	001401				BEQ	13:	
(2)	047316	104000				EMT		;INCORRECT STORE
13955								;FOR TIGHTER SCOPE LOOP
13956								;REPLACE ERROR CALL WITH
13957								; "BR 12:" = 000762
13958	047320				13:		; THE FOLLOWING WILL TEST	DSTM=5 MTPI.
13959								
13960	047320	012767	030340	130450		MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
13961	047326	012700	052525			MOV	#52525,R0	;LOAD TEST DATA INTO R0
13962	047332	012702	037600			MOV	#<\$TMP2+2>,R2	;LOAD ADDR. OF LOC. \$TMP2+2 INTO R2
13963	047336	012767	100000	170232		MOV	#100000,\$TMP2	;LOAD VIRT. ADDR. OF TEST LOC. INTO \$TMP2
13964	047344	010046			14:	MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
13965	047346	105067	122736			CLRB	KIPDR4	;MAKE KERNEL PAGE 4 NON-RESIDENT
13966	047352	006652				MTPI	#-(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
13967	047354	112767	000006	122726		MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT
13968	047362	013701	100000			MOV	#100000,R1	;READ FROM ADDRESS 140000
13969	047366	020001				CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY
13970	047370	001401				BEQ	15:	
(2)	047372	104000				EMT		;INCORRECT STORE
13971								;FOR TIGHTER SCOPE LOOP
13972								;REPLACE ERROR CALL WITH
13973								; "BR 14:" = 000764
13974	047374				15:		; THIS WILL TEST DSTM = 6	MTPI.
13975								
13976	047374	012767	030340	130374		MOV	#030340,PSW	;MAKE PREVIOUS MODE USER
13977	047402	012700	052525			MOV	#52525,R0	;LOAD TEST DATA INTO R0
13978	047406	005002				CLR	R2	;MAKE REGISTER 2 ZERO
13979	047410	010046			16:	MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK
13980	047412	105067	122672			CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT
13981	047416	006662	100000			MTPI	100000(R2)	;LOAD TEST DATA INTO PHYSICAL 140000
13982	047422	112767	000006	122660		MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT

```

13983 047430 013701 100000      MOV      @#100000,R1      ;READ FROM ADDRESS 140000
13984 047434 020001              CMP      R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
13985 047436 001401              BEQ      17$            ;
(2) 047440 104000              EMT                    ;INCORRECT STORE
13986                                ;FOR TIGHTER SCOPE LOOP
13987                                ;REPLACE ERROR CALL WITH
13988                                ;"BR 16$" = 000763
13989 047442              17$:      ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
13990                                ;
13991 047442 012767 030340 130326      MOV      #030340,PSW    ;MAKE PREVIOUS MODE USER
13992 047450 012700 125252              MOV      #125252,R0    ;LOAD TEST DATA INTO R0
13993 047454 012767 100000 170114      MOV      #100000,#TMP2 ;LOAD VIRT. ADDR. OF TEST LOCATION
13994                                ;INTO LOCATION #TMP2
13995 047462 012702 037576              MOV      #TMP2,R2      ;LOAD ADDRESS OF #TMP2 INTO R2
13996 047466 010046              18$:      MOV      R0,-(KSP)      ;PUSH TEST DATA ON KERNEL STACK
13997 047470 105067 122614              CLRB    KIPDR4        ;MAKE KERNEL PAGE 4 NON-RESIDENT
13998 047474 006672 000000              MTPI    @0(R2)        ;LOAD TEST DATA INTO PHYSICAL 140000
13999 047500 112767 000006 122602      MOV     #006,KIPDR4    ;MAKE KERNEL PAGE 4 RESIDENT
14000 047506 013701 100000              MOV     @#100000,R1    ;READ FROM ADDRESS 140000
14001 047512 020001              CMP     R0,R1          ;SEE IF DATA WAS STORED CORRECTLY
14002 047514 001401              BEQ     19$            ;
(1) 047516              20$:      EMT                    ;INCORRECT STORE
(2) 047516 104000              ;FOR TIGHTER SCOPE LOOP
14003                                ;REPLACE ERROR CALL WITH
14004                                ;"BR 18$" = 000763
14005                                ;
14006 047520 012767 021550 130522      19$:      MOV     #T0250,MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
14007
14008
14009
14021                                ;*****
(2)                                ;TEST 415      MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE
(3)                                ;*****
(2) 047526              TS415:
14022 047526 012700 077406              1$:      MOV     #77406,R0     ;MAKE ALL USER I-SPACE PAGES RESIDENT
14023                                ;READ/WRITE, LENGTH 200 BLOCKS
14024 047532 012702 000010              MOV     #10,R2         ;SET LOOP COUNTER TO 8
14025 047536 012701 177600              MOV     #UIPDR0,R1    ;LOAD ADDRESS OF FIRST PDR IN R1
14026 047542 010021              2$:      MOV     R0,(R1)+      ;LOAD PDR WITH 77406
14027 047544 077202              SOB     R2,2$         ;LOOP UNTIL 8 USER PDRS LOADED
14028 047546 012767 140340 130222      3$:      MOV     #140340,PSW   ;GO TO USER MODE FOR THIS TEST
14029 047554 012767 077406 122526      MOV     #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
14030 047562 012767 001400 122560      MOV     #1400,KIPAR4  ;MAP KERNEL I PAGE 4 TO 24K
14031 047570 012767 001400 130052      MOV     #1400,UIPAR4  ;MAP USER I PAGE 4 TO 24K
14032 047576 012700 036514              MOV     #36514,R0     ;LOAD DATA PATTERN INTO R0
14033 047602 010037 100000              MOV     R0,@#100000   ;LOAD DATA PATTERN INTO PHY 140000
14034 047606 012702 100000              MOV     #100000,R2    ;LOAD VIRTUAL ADDRESS INTO R2
14035                                ;THE FOLLOWING WILL TEST DSTM=0 MFPI
14036                                ;
14037 047612 012767 050110 130430      MOV     #21$,MMVEC    ;SET M.M. VECTOR TO 21$
14038 047620 105067 127764              CLRB    UIPDR4        ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
14039 047624 012767 140340 130144      MOV     #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
14040 047632 006506              4$:      MFPI    KSP           ;PUT KERNEL STACK POINTER ON USER STACK
14041 047634 022706 000600              CMP     #USESTK,USP   ;WAS SOMETHING PUSHED ON STACK AT 1$
14042 047640 001405              BEQ     5$            ;BRANCH IF NOTHING WAS PUSHED
14043 047642 012600              MOV     (USP)+,R0     ;POP USER STACK INTO R0

```

```
14044 047644 012701 001000      MOV    #KERSTK,P'    ;EXPECTING 1100 AS KSP
14045 047650 020001              CMP    R0,R1        ;DID YOU GET THE RIGHT POINTER?
14046 047652 001401              BEQ    6$
      (1) 047654              5$:
      (2) 047654 104000          EMT
14047
14048
14049
14050 047656              6$:      ;THE FOLLOWING WILL TEST DSTH=1 MFPI.
14051 047656 012767 140340 130112 7$:      MOV    #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
14052 047664 012700 036514      MOV    #36514,R0    ;LOAD DATA EXPECTED INTO R0
14053 047670 012702 100000      MOV    #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
14054 047674 006512              MFPI   (R2)         ;READ FROM PHYSICAL 140000
14055 047676 012601              MOV    (USP)+,R1    ;POP USER STACK INTO R1
14056 047700 020001              CMP    R0,R1        ;WAS DATA FETCHED SAME AS STORED
14057 047702 001401              BEQ    9$
      (2) 047704 104000          EMT
14058
14059
14060
14061              ;THE FOLLOWING WILL TEST DSTH=2 MFPI.
14062 047706 012767 140340 130062 9$:      MOV    #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
14063 047714 012702 100000      MOV    #100000,R2   ;LOAD VIRTUAL ADDRESS INTO R2
14064 047720 006522              MFPI   (R2)+       ;READ FROM PHYSICAL 140000
14065 047722 012601              MOV    (USP)+,R1    ;POP USER STACK INTO R1
14066 047724 020001              CMP    R0,R1        ;WAS DATA FETCHED SAME AS STORED
14067 047726 001401              BEQ    11$
      (2) 047730 104000          EMT
14068
14069
14070
14071              ;THE FOLLOWING WILL TEST DSTH=3 MFPI.
14072 047732 012767 140340 130036 11$:     MOV    #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
14073 047740 006537 100000      MFPI   #100000      ;READ FROM PHYSICAL 140000
14074 047744 012601              MOV    (USP)+,R1    ;POP USER STACK INTO R1
14075 047746 020001              CMP    R0,R1        ;WAS DATA FETCHED SAME AS STORED
14076 047750 001401              BEQ    13$
      (2) 047752 104000          EMT
14077
14078
14079
14080              ;THE FOLLOWING WILL TEST DSTH=4 MFPI.
14081 047754 012767 140340 130014 13$:     MOV    #140340,PSW   ;MAKE PREVIOUS MODE DERNEL PRESENT USER
14082 047762 012702 100002      MOV    #100002,R2   ;LOAD VIRTUAL ADDRESS INTO R2
14083 047766 006542              MFPI   -(R2)        ;READ FROM PHYSICAL 140000
14084 047770 012601              MOV    (USP)+,R1    ;POP USER STACK INTO R1
14085 047772 020001              CMP    R0,R1        ;WAS DATA FETCHED SAME AS STORED
14086 047774 001401              BEQ    15$
      (2) 047776 104000          EMT
14087
14088
14089
14090              ;THE FOLLOWING WILL TEST DSTH=5 MFPI.
14091
14092 050000 012767 140340 127770 15$:     MOV    #140340,PSW   ;MAKE PREVIOUS MODE KERNEL PRESENT USER
14093 050006 012767 100000 167562      MOV    #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR INTO LOC $TMP2
```


14148	050160	012746	007777	4:	MOV	07777,(KSP)	;	PUSH DATA ON KERNEL STACK
14149	050164	106606			MTPD	USP	;	LOAD THE USER STACK POINTER
14:50	050166	106506			MFPD	USP	;	READ USER STACK POINTER
14151	050170	012601			MOV	(KSP),R1	;	POP KERNEL STACK INTO R1
14152	050172	022701	007777		CMP	07777,R1	;	WAS USER STACK POINTER CHANGED?
14153	050176	001401			BEQ	5:		
(2)	050200	104000			ENT		;	USER STACK POINTER NOT CHANGED
14154							;	FOR TIGHTER SCOPE LOOP
14155							;	REPLACE ERROR CALL WITH
14156							;	"BR 4:" = 000767
14157	050202	012746	000600	5:	MOV	#USESTK,-(KSP)	;	GET READY TO RESTORE USER STK. PTR.
14158	050206	106606			MTPD	USP	;	RESTORE USER STACK POINTER
14159								
14168							;
(2)							;	TEST 417 MOVE FROM PREVIOUS I-SPACE (PREVIOUS-CURRENT-KERNEL)
(3)							;
(2)	050210						TS417:	
14169	050210	005037	177776	1:	CLR	#PSW	;	SET PREVIOUS = CURRENT = KERNEL
14170	050214	012700	001000		MOV	#KERSTK,RO	;	SETUP VALUE FOR STACK POINTER
14171	050220	010006			MOV	RO,KSP	;	LOAD STACK POINTER
14172	050222	006506			MFPD	KSP	;	THE VALUE "STACK" SHOULD BE PUSHED
14173							;	BEFORE BEING DECREMENTED
14174	050224	011601			MOV	(KSP),R1	;	READ DATA WHICH WAS PUSHED
14175	050226	020001			CMP	RO,R1	;	WAS THE ORIGINAL VALUE OF THE
14176							;	STACK POINTER PUSHED?
14177	050230	001401			BEQ	2:		
(2)	050232	104000			ENT		;	MFPD FETCHED WRONG DATA
14178							;	FOR TIGHTER SCOPE LOOP
14179							;	REPLACE ERROR CALL WITH
14180							;	"BR 1:" = 000766
14181	050234	005740		2:	TST	-(RO)	;	SETUP EXPECTED STACK POINTER VALUE
14182	050236	020600			CMP	KSP,RO	;	WAS THE STACK POINTER DECREMENTED?
14183	050240	001401			BEQ	3:		
(2)	050242	104000			ENT		;	STACK NOT PUSHED BY THE MFPD
14184							;	FOR TIGHTER SCOPE LOOP
14185							;	REPLACE ERROR CALL WITH
14186							;	"BR 1:" = 000762
14187	050244	012706	001000	3:	MOV	#KERSTK,KSP	;	RESTORE STACK POINTER
14188	050250	005067	127316		CLR	SRO	;	TURN OFF MEMORY MANAGEMENT UNIT
14189	050254	000167	000476		JMP	EXATST	;	GET OVER SUBROUTINES TO EXTENDED ADRS TESTS


```

14191          .SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****
14192
14193          .SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW
14194          ;*****
14195          ;*
14196          ;* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW
14197          ;* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN "TBITPS" SO THAT
14198          ;* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS
14199          ;* WARRANT T-BIT TRAPPING.
14200          ;*
14201          ;*****
14202 050260 036727 127512 000020 TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?
14203 050266 001411          BEQ 1# ;EXIT IF NO
14204 050270 016746 127502          MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK
14205 050274 011667 167270          MOV (SP),TBITPS ;ALSO SAVE IT IN "TBITPS" FOR
14206          ;RESTORING LATER
14207 050300 042716 000020          BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW
14208 050304 012746 050312          MOV #1,-(SP) ;PUSH PC OF "RTS" ON STACK
14209 050310 000006          RTT ;"RETURN" TO 1# WITH T-BIT OFF
14210 050312 000207 1#: RTS PC ;RETURN TO PROGRAM
14211
14212          .SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW
14213          ;*****
14214          ;*
14215          ;* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS
14216          ;* PREVIOUS CONDITION BY RESTORING THE "T-BIT PSW" SAVED BY THE
14217          ;* "TOFF" SUBROUTINE IN THE "TBITPS" LOCATION.
14218          ;*
14219          ;*****
14220 050314 036727 167250 000020 TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?
14221 050322 001410          BEQ 1# ;EXIT IF NO
14222 050324 016746 167240          MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK
14223 050330 012767 000340 167232          MOV #340,TBITPS ;RESET THE "TBITPS" LOCATION
14224 050336 012746 050344          MOV #1,-(SP) ;PUSH PC OF "RTS" ON STACK
14225 050342 000006          RTT ;"RETURN" TO 1# WITH T-BIT RESTORED
14226 050344 000207 1#: RTS PC ;RETURN TO PROGRAM
14227
14228          .SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
14229          ;*****
14230          ;*
14231          ;* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST
14232          ;* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND
14233          ;* PDR'S TO A 1. THE "INITIAL STATE" OF HAVING ALL BITS=1 IS
14234          ;* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED I' RESPONSE TO
14235          ;* A SINGLE PAR OR PDR ADDRESS.
14236          ;*
14237          ;*****
14238
14239 050346 012702 000010 SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
14240 050352 012701 172300          MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
14241 050356 012721 177777 1#: MOV #-1,(R1) ;SET BITS IN KERNEL PDR TO 1
14242 050362 077203          SOB R2,1# ;LOOP TO 1# UNTIL ALL KERNEL PDR'S LOADED
14243 050364 012702 000010          MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
14244 050370 012701 172340          MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
14245 050374 012721 177777 2#: MOV #-1,(R1) ;SET BITS IN A KERNEL PAR TO 1
14246 050400 077203          SOB R2,2# ;LOOP TO 2# UNTIL ALL KERNEL PAR'S LOADED

```

```

14247 050402 012702 000010          MOV    #10,R2          ;LOAD LOOP COUNTER WITH AN 8
14248 050406 012701 177600          MOV    @UIPDR0,R1     ;LOAD ADDRESS OF FIRST PDR INTO R1
14249 050412 012721 177777          3$:   MOV    #-1,(R1)•   ;SET BITS IN A USER PDR TO 1
14250 050416 077203                   SOB    R2,3$          ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
14251 050420 012702 000010          MOV    #10,R2          ;LOAD LOOP COUNTER WITH AN 8
14252 050424 012701 177640          MOV    @UIPAR0,R1     ;LOAD ADDRESS OF FIRST PAR INTO R1
14253 050430 012721 177777          4$:   MOV    #-1,(R1)•   ;SET BITS IN A USER PAR TO 1
14254 050434 077203                   SOB    R2,4$          ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
14255 050436 000207                   RTS    PC              ;RETURN TO TEST
14256
14257                               .SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S
14258                               ;*****
14259                               ;*
14260                               ;* THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
14261                               ;* READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
14262                               ;* WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
14263                               ;* ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
14264                               ;* BE REPORTED BY THIS SUBROUTINE.
14265                               ;*
14266                               ;*****
14267 050440                               CMPREG:
14268 050440 012701 172300          MOV    @KIPDR0,R1     ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
14269 050444 012704 000010          MOV    #10,R4          ;LOAD LOOP COUNTER WITH AN 8
14270 050450 012705 077416          MOV    #77416,R5      ;PUT EXPECTED PDR CONTENTS IN R5
14271 050454 021105          1$:   CMP    (R1),R5      ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
14272 050456 001403          BEQ    2$              ;BRANCH IF YES
14273 050460 020100          CMP    R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
14274 050462 001401          BEQ    2$              ;
14275 (2) 050464 104000          EMT                    ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PRO
14276                               ;FOR TIGHTER SCOPE LOOP
14277                               ;REPLACE ERROR CALL WITH
14278 050466 062701 000002          2$:   ADD    #2,R1          ;FORM NEXT ADDRESS
14279 050472 077410          SOB    R4,1$          ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
14280 050474 012701 172340          MOV    @KIPAR0,R1     ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
14281 050500 012704 000010          MOV    #10,R4          ;LOAD LOOP COUNTER WITH AN 8
14282                               ;****F11 CHANGE**** FROM #7777 TO #177777
14283 050504 012705 177777          MOV    #177777,R5     ;PUT EXPECTED PAR CONTENTS IN R5
14284 050510 021105          3$:   CMP    (R1),R5      ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
14285 050512 001403          BEQ    4$              ;BRANCH IF YES
14286 050514 020100          CMP    R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
14287 050516 001401          BEQ    4$              ;
14288 (2) 050520 104000          EMT                    ;A PAR WAS EFFECTED BY CLEARING A DIFFENENT PAR/PDR
14289                               ;FOR TIGHTER SCOPE LOOP
14290                               ;REPLACE ERROR CALL WITH
14291 050522 062701 000002          4$:   ADD    #2,R1          ;FORM NEXT ADDRESS
14292 050526 077410          SOB    R4,3$          ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
14293 050530 012701 177600          MOV    @UIPDR0,R1     ;LOAD ADDRESS OF FIRST USER PDR IN R1
14294 050534 012704 000010          MOV    #10,R4          ;LOAD LOOP COUNTER WITH AN 8
14295 050540 012705 077416          MOV    #77416,R5      ;PUT EXPECTED PDR CONTENTS IN R5
14296 050544 021105          5$:   CMP    (R1),R5      ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
14297 050546 001403          BEQ    6$              ;BRANCH IF YES
14298 050550 020100          CMP    R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
14299 050552 001401          BEQ    6$              ;
14300 (2) 050554 104000          EMT                    ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR

```

```

14300                                     ;FOR TIGHTER SCOPE LOOP
14301                                     ;REPLACE ERROR CALL WITH
14302                                     ;AN "RTS PC" = 000207
14303 050556 062701 000002          64:  ADD    #2,R1          ;FORM NEXT ADDRESS
14304 050562 077410                SOB    R4,54          ;LOOP TO 54 UNTIL ALL USER PDR'S CHECKED
14305 050564 012701 177640          MOV    #UIPAR0,R1     ;LOAD ADDRESS OF FIRST USER PAR IN R1
14306 050570 012704 000010          MOV    #10,R4        ;LOAD LOOP COUNTER WITH AN 8
14307                                     ;****F11 CHANGE**** FROM #7777 TO #177777
14308 050574 012705 177777          MOV    #177777,R5    ;PUT EXPECTED PAR CONTENTS IN R5
14309 050600 021105                74:  CMP    (R1),R5      ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
14310 050602 001403                BEQ    #8             ;BRANCH IF YES
14311 050604 020100                CMP    R1,R0         ;WAS IT THE REG. THAT WAS CLEARED?
14312 050606 001401                BEQ    #8
14313 (2) 050610 104000                EMT                    ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
14314                                     ;FOR TIGHTER SCOPE LOOP
14315                                     ;REPLACE ERROR CALL WITH
14316 050612 062701 000002          84:  ADD    #2,R1          ;FORM NEXT ADDRESS
14317 050616 077410                SOB    R4,74          ;LOOP TO 74 UNTIL ALL USER PAR'S CHECKED
14318 050620 000207                RTS    PC             ;RETURN TO TEST
14319
14320 .SBTTL INHIBIT 'RESETS' WHILE UNDER APT
14321 ;*****
14322 ;*
14323 ;* THIS SUBROUTINE CONTROLS THE USAGE OF RESET INST'S WHILE
14324 ;* RUNNING UNDER APT. RESETS ARE ALLOWED DURING THE FIRST
14325 ;* PASS OF THE DIAGNOSTIC.
14326 ;*
14327 050622 126727 130172 000001  CHKAPT: CMPB   #ENV,#1      ;ARE WE RUNNING UNDER APT?
14328 050630 001003                BNE    #1             ;NO BRANCH
14329 050632 005767 130150          TST    #PASS         ;IS THIS THE FIRST PASS?
14330 050636 001002                BNE    #RETA         ;NO BRANCH
14331 050640 062705 000002          14:  ADD    #2,R5         ;BUMP RETURN ADDRESS FOR NORMAL TESTING
14332 050644 000205                RETA:  RTS    R5      ;RETURN
14333
14334
14335 .SBTTL ERROR ROUTINE FOR MEMORY MANAGEMENT TEST
14336 ;*****
14337 ;* THIS IS THE ONLY ERROR REPORT FOR ALL THE MMU TESTS
14338 ;*****
14339
14340 050646 004767 062556                ERROR3: JSR    PC,ABORT      ;ARE WE UNDER UFD ?
14341 050652 012737 000002 001002     MOV    #2,#FATAL     ;SET UP FATAL ERROR NUMBER
14342 050660 012767 000001 130112     MOV    #1,#MSGTY     ;SET FATAL ERROR FLAG
14343 050666 032737 000001 001020     BIT    #1,#ENV       ;UNDER APT ?
14344 050674 001004                BNE    #MMULT
14345 050676 012700 050710                MOV    #MMUMSG,R0
14346 050702 004767 062456                JSR    PC,TYPE
14347 050706 000777                MMULT: BR    .        ;STAY IN LOOP
14348
14349 050710 040506 046111 042105     MMUMSG: .ASCIZ /FAILED DURING MMU TESTING/<12><15>
14350 050716 042040 051125 047111
14350 050724 020107 046515 020125
14350 050732 042524 052123 047111
14350 050740 005107 000015

```

14351
14352
14353
14354
14355
14356
14357
14358
14359
14360 050744 000000
14361 050746 000000
14362 050750 000000
14363 050752 000002
14364
 (2)
 (3)
 (2) 050756
14365
14366
14367
14368
14369
14370
14371
14372 050756 012767 177600 121372
14373 050764 012737 000007 001004
14374 050772 012737 051550 000030
14375 051000 000244
14376 051002 032777 000200 150566
14377 051010 001001
14378 051012 000470
14379
14380 051014 016767 126764 177722
14381 051022 016767 126760 177716
14382 051030 012767 051132 126746
14383 051036 012767 000340 126742
14384 051044 012767 004000 121276
14385 051052 012767 000001 126512
14386 051060 012767 000020 121430
14387
14388 051066 012767 000000 177656
14389 051074 026727 177652 000005
14390 051102 001420
14391 051104 013737 100000 100000
14392 051112 000240
14393 051114 012767 000000 126450
14394 051122 012767 000000 121366
14395 051130 104000
14396 051132 006367 121212
14397 051136 005267 177610
14398 051142 000754
14399 051144 016767 177574 126632
14400 051152 016767 177570 126626
14401 051160 012767 000000 126404
14402 051166 012767 000000 121322
14403

.EVEN

; STORAGE AREAS FOR THE FOLLOWING FEW TESTS

TMP: .WORD 0
TMP1: .WORD 0
MEM: .WORD 0
CNTR: .BLKW 2

;TEST 420 TEST ADDRESS BITS 17-21

TS420:

; THIS TEST WILL DETERMINE WHETHER THE KDF11-B CAN
; DRIVE ADDRESS BITS 17-21.

EXATST: MOV #177600,KIPAR7 ;I 0 PAGE
MOV #7,#1TESTN
MOV #ERRORA,#30
CLZ ;CLR THE Z BIT
BIT #200,BSWR ;WANT TO TEST THIS ?
BNE ESR3
BR NEXT
ESR3: MOV ERRVEC,TMP
MOV ERRVEC+2,TMP1
MOV #ESR0,ERRVEC ;PREPARE FOR NEW
MOV #340,ERRVEC+2 ;INTERRUPT
MOV #4000,KIPAR4 ;SET FOR ADRS BIT 17
MOV #BIT0,SRO
MOV #BIT4,SR3 ;TURN 22 BIT ADRSG AND
;MEMORY MANG ON
ESR1: MOV #0,CNTR
CMP CNTR,#5
BEQ ESR2
MOV #100000,#100000 ;TRY ADRSG IT
MOV #0,SRO
MOV #0,SR3 ;TURN OFF MM AND 22 BIT
;ERROR
ESR0: ASL KIPAR4 ;SHOULD TIME OUT
INC CNTR ;SHIFT 1 TO THE LEFT
BR ESR1
ESR2: MOV TMP,ERRVEC
MOV TMP1,ERRVEC+2 ;RESTORE
MOV #0,SRO
MOV #0,SR3 ;TURN OFF MM AND 22 BIT

14404
 14405
 (2)
 (3)
 (2) 051174

14406
 14407
 14408
 14409
 14410
 14411
 14412
 14413 051174 012767 000001 126370
 14414 051202 016767 126576 177534
 14415 051210 016767 126572 177530
 14416 051216 012767 051300 126560
 14417 051224 012767 000340 126554
 14418 051232 012737 123456 000000
 14419 051240 012767 002000 121102
 14420 051246 013737 100000 100000
 14421 051254 023727 000000 123456
 14422 051262 001407
 14423 051264 012767 000000 126300
 14424 051272 012767 000000 121216
 14425 051300 104000
 14426 051302 005067 126264
 14427 051306 016767 177432 126470
 14428 051314 016767 177426 126464
 14429
 14430
 14431
 14432
 (2)
 (3)
 (2) 051322

```

;*****
;TEST 421 TEST ADDRESS BIT 16
;*****
TS421:
;*****

```

```

; THIS TEST WILL CHECK TO SEE IF THE 17TH BIT (16) OF AN 18 BIT
; ADDRESS CAN BE DRIVEN.
;*****

```

```

NEXT:  MOV    #BIT0,SRO                ;TURN ON MEM MANG
      MOV    ERR.ZC,TMP
      MOV    ERRVEC+2,TMP1
      MOV    #ESR4,ERRVEC
      MOV    #340,ERRVEC+2
      MOV    #123456,B#C                ;MODIFY LOCATION ZERO
      MOV    #2000,KIPAR4
      MOV    #100000,B#100000
      CMP    B#0,#123456                ;CHANGED ?
      BEQ    ESR5
      MOV    #0,SRO
      MOV    #0,SR3                    ;TURN OFF MM AND 22 BIT
      ESR4:  EMT                        ;LOC 0 WAS ALTERED
      ESR5:  CLR    SRO                  ;TURN OFF MM
      MOV    TMP,ERRVEC
      MOV    TMP1,ERRVEC+2              ;RESTORE VECTORS

```

14433
 14434
 14435
 14436
 14437
 14438
 14439
 14440
 14441
 14442 051322 000244
 14443 051324 032777 001000 150244
 14444 051332 001002
 14445 051334 000167 000434
 14446
 14447 051340 012767 000001 000374
 14448 051346 016767 126432 177376
 14449 051354 016767 126426 177372
 14450 051362 012767 051400 126414
 14451 051370 012704 172100
 14452 051374 005714
 14453 051376 000407

```

;*****
;TEST 422 TEST PARITY ERROR DETECTION LOGIC
;*****
TS422:
;*****

```

```

; THIS TEST WILL USE THE MEMORY PARITY CSR TO GENERATE
; A PARITY ERROR THAT THE CPU SHOULD DETECT.
;*****

```

```

      CLZ                ;CLR THE Z BIT
      BIT    #BIT9,BSWR  ;TEST THIS ?
      BNE    ESR57
      JMP    Q22TST      ;NO
      ESR57: MOV    #1,ERR0RE          ;DUMMY BIT
      MOV    ERRVEC,CNTR
      MOV    ERRVEC+2,CNTR+2
      MOV    #21,ERRVEC
      MOV    #172100,R4
      14:  TST    (R4)
      BR    ESR7

```

```
14454 051400 022704 172136      28:  CMP      #172136,R4      ;TOP YET ?
14455 051404 100403
14456 051406 062704 000002      ADD      #2,R4
14457 051412 000770      BR
14458 051414 104000      ESR6:  EMT          ;COULD NOT FIND MEM PAR CSR
14459 051416 016767 126472 177320      ESR7:  MOV      114,TMP      ;SAVE CONTENTS OF 114
14460 051424 012767 051472 126462      MOV      #ESR8,114      ;PARITY TRAP VECTOR
14461 051432 012714 000005      MOV      #5,(R4)        ;BITS 0,2 IN MEM CSR
14462 051436 010167 177306      MOV      R1,MEM
14463 051442 042714 000004      BIC      #4,(R4)        ;TURN OFF WRT WRG PAR BIT
14464 051446 016701 177276      MOV      MEM,R1         ;SOME DATA TO MEMORY FROM
14465 051452 012714 000000      ESR9:  MOV      #0,(R4)   ;TURN OFF P ERR
14466 051456 010167 177266      MOV      R1,MEM
14467 051462 016767 177256 126424      MOV      TMP,114        ;RESTORE VECTOR CONTENTS
14468
14469 051470 104000      EMT
14470 051472 032714 100000      ESR8:  BIT      #BIT15,(R4) ;SEE IF BIT 15 SET IN MEM CSR
14471 051476 001765      BEQ      ESR9           ;Z IS SET IF AN ERROR
14472 051500 005000      CLR      R0
14473 051502 005200      ESR19: INC      R0        ;TIME FOR LED TO BE SEEN
14474 051504 005700      TST      R0
14475 051506 100375      BPL      ESR19
14476 051510 012714 000000      MOV      #0,(R4)
14477 051514 010167 177230      MOV      R1,MEM
14478 051520 016767 177220 126366      MOV      TMP,114
14479 051526 016767 177220 126250      MOV      CNTR,ERRVEC
14480 051534 016767 177214 126244      MOV      CNTR+2,ERRVEC+2
14481 051542 005067 000174      CLR      ERRERE        ;RESTORE DUMMY BIT
14482 051546 000512      BR      Q22TST        ;GO FIND A Q22BE
14483
14484 ;*****
14485 051550 004767 061654      ERRORA: JSR      PC,ABORT      ;ARE WE UNDER UFD ?
14486 051554 012737 000007 001002      MOV      #7,#FATAL
14487 051562 012767 000001 127210      MOV      #1,#MSGTY
14488 051570 032737 000001 001020      BIT      #1,#ENV        ;UNDER APT ?
14489 051576 001013      BNE      EXADHT
14490 051600 026727 000136 000001      CMP      ERRERE,#1      ;DUMMY BIT THERE ?
14491 051606 001003      BNE      ESR34
14492 051610 012700 051673      MOV      #CSRMSG,R0
14493 051614 000402      BR      ESR34+4
14494 051616 012700 051630      ESR34: MOV      #EXTMSG,R0
14495 051622 004767 061536      JSR      PC,TYPE
14496 051626 000777      EXADHT: BR
14497
14498
14499 051630 040506 046111 042105      EXTMSG: .ASCIZ  /FAILED DURING EXTENDED ADRS TEST/<12><15>
051636 042040 051125 047111
051644 020107 054105 042524
051652 042116 042105 04 140
051660 051104 020123 042524
051666 052123 006412 000
14500
14501 051673 106 044501 042514      CSRMSG: .ASCIZ  /FAILED MEM PARITY ERROR DETECT TEST/<12><15>
051700 020104 042515 020115
051706 040520 044522 054524
051714 042440 051122 051117
```

051722 042040 052105 041505
051730 020124 042524 052123
051736 006412 000

14502
14503
14504 051742 000000
14505
14506
14507 051744 000510
14508 051746 000000
14509 051750 170000
14510 051752 000000
14511 051754 000000
14512 051756 000000
14513 051760 000000
14514 051762 000000
14515 051764 000000
14516 051766 000000
14517 051770 000000
14518 051772 000000
14519
14520
14521
14522
(2)
(3)
(2) 051774
14523
14524
14525
14526
14527
14528
14529
14530
14531
14532 051774 000244
14533 051776 032777 000100 147572
14534 052004 001002
14535 052006 000167 001502
14536 052012 012737 000011 001004
14537 052020 012737 053402 000030
14538
14539
14540 052026 005067 125540
14541 052032 016767 125746 176710
14542 052040 012767 052230 125736
14543 052046 016767 177676 177676
14544 052054 016767 177664 177664
14545 052062 005777 177664
14546
14547 052066 016767 177660 177660
14548 052074 016767 177654 177654
14549 052102 062767 000002 177646
14550 052110 016767 177640 177642
14551 052116 062767 000004 177634

.EVEN
ERRORE: .WORD 0 ;DUMMY BIT
;*****
VECT1: .WORD 510 ;FIRST DEV VECTOR Q22BE
DEVECT: .WORD 0
DEV1: .WORD 170000 ;FIRST DEV ADRS Q22BE
DEVADR: .WORD 0
CSR1: .WORD 0
CSR2: .WORD 0
BA: .WORD 0
WC: .WORD 0
DATA: .WORD 0
LATCNT: .WORD 0
MVL CNT: .WORD 0
SINGOA: .WORD 0

;*****
;TEST 423 SEE IF A Q22BE(QBE) IS THERE
;*****
TS423:

;*****
; ROUTINE TO SIZE FOR THE Q22BE(QBE) DEVICE ADDRESS
; WE WILL LOOK FOR A Q22BE(QBE). IF IT ISN'T THERE
; WE WILL CAUSE AN ERROR VIA THE EMT INSTRUCTION.
;*****

Q22TST: CLZ
BIT #BIT6,BSWR
BNE 1#
JMP BDVTST
1#: MOV #11,0#1TESTN ;TEST NUM IN MAILBOX
MOV #ERRORB,0#30 ;SET UP FOR CORRECT EMT
CLR SRO
MOV ERRVEC, MEM ;STORE ERRVEC CONTENTS
MOV #ESR99,ERRVEC
MOV DEV1,DEVADR
MOV VECT1,DEVECT
ESR11: TST #DEVADR ;SEE IF IT RESPONDS
MOV DEVADR,CSR1
MOV CSR1,CSR2
ADD #2,CSR2 ;YES IT DID
MOV CSR1,BA
ADD #4,BA

```
14552 052124 016767 177624 177630      MOV      CSR1,WC
14553 052132 062767 000006 177622      ADD      #6,WC
14554 052140 016767 177610 177616      MOV      CSR1,DATA
14555 052146 062767 000010 177610      ADD      #10,DATA
14556 052154 016767 177574 177604      MOV      CSR1,LATCNT
14557 052162 062767 000012 177576      ADD      #12,LATCNT
14558 052170 016767 177560 177572      MOV      CSR1,MVLCNT
14559 052176 062767 000014 177564      ADD      #14,MVLCNT
14560 052204 016767 177544 177560      MOV      CSR1,SIMGOA
14561 052212 062767 000016 177552      ADD      #16,SIMGOA
14562 052220 016767 176524 125556      MOV      MEM,ERRVEC      ;RESTORE ERROR VECTOR
14563 052226 000413                BR        ESR98
14564 052230 062767 000020 177514  ESR99:  ADD      #20,DEVADR
14565 052236 062767 000004 177502      ADD      #4,DEVECT
14566 052244 026727 177476 000550      CMP      DEVECT,#550
14567 052252 001303                BNE      ESR11           ;GO TRY ANOTHER ADRS
14568 052254 104000                EMT
14569 052256 016700 177464  ESR98:  MOV      DEVECT,R0
14570 052262 062700 000002      ADD      #2,R0
14571 052266 012710 000340      MOV      #340,(R0)
14572
14573
14574      ;*****
(2)      ;TEST 424      USE Q22BE(QBE) TO ALTER THE INTERRUPT LEVEL BITS
(3)      ;*****
(2) 052272      TS424:
14575      ;*****
14576      ;
14577      ;
14578      ;      GENERATE A Q22BE(QBE) SOFTWARE INTR AT LEVEL 4.
14579      ;      THE FOLLOWING CODE WILL USE THE Q22(QBE) BUS EXERCISER
14580      ;      TO GENERATE INTERRUPTS THAT THE CPU SHOULD EITHER
14581      ;      HONOR OR IGNORE DEPENDING ON THE INTR LEVEL BITS.
14582      ;
14583      ;*****
14584
14585
14586 052272 012777 052336 177446      MOV      #ESR21,#DEVECT      ;INTERRUPT TO ESR21
14587 052300 012777 000001 177446      MOV      #1,BCSR1
14588 052306 106427 000140                MTPS     #140      ;CPU AT LEVEL THREE
14589 052312 012777 000003 177436      MOV      #3,BCSR2
14590 052320 000240                NOP
14591 052322 012777 000002 177426      MOV      #2,BCSR2
14592 052330 000240                NOP
14593 052332 000240                NOP
14594 052334 104000                EMT
14595 052336 012777 052372 177402  ESR21:  MOV      #ESR22,#DEVECT      ;CHANGE INTR VECTOR
14596 052344 106427 000200                MTPS     #200
14597 052350 012777 000003 177400      MOV      #3,BCSR2
14598 052356 000240                NOP
14599 052360 012777 000000 177370      MOV      #0,BCSR2      ;INTR LEVEL IS 4
14600 052366 000240                NOP      ;INTR SHOULD NOT HONORED
14601 052370 000401                BR        ESR23
14602 052372 104000  ESR22:  EMT      ;INTR HONORED A ERROR
14603      ;*****
14604      ;      GENERATE AN INTERRUPT AT LEVEL FIVE
```



```
14605 ;*****
14606
14607 052374 012777 052436 177344 ESR23: MOV #ESR24, @DEVECT
14608 052402 106427 000200 MTPS #200 ;CPU AT LEVEL FOUR
14609 052406 012777 000001 177340 MOV #1, @CSR1 ;TRY TO CAUSE AN INTERRUPT
14610 052414 012777 000007 177334 MOV #7, @CSR2 ;AT LEVEL FIVE
14611 052422 000240 NOP
14612 052424 012777 000006 177324 MOV #6, @CSR2 ;CLRS GO, SETS DONE
14613 052432 000240 NOP
14614 052434 104000 EMT ;INTR DID NOT HAPPEN
14615 052436 012777 052472 177302 ESR24: MOV #ESR27, @DEVECT ;ALTER INTR VECTOR
14616 052444 106427 000240 MTPS #240 ;CHANGE LEVEL TO FIVE
14617 052450 012777 000007 177300 MOV #7, @CSR2
14618 052456 000240 NOP
14619 052460 012777 000000 177270 MOV #0, @CSR2
14620 052466 000240 NOP
14621 052470 000401 BR ESR28
14622 052472 104000 ESR27: EMT ;IF HERE, AN ERROR
14623 ;*****
14624 ; TRY INTERRUPT AT LEVEL SIX
14625 ;*****
14626 052474 012777 052532 177244 ESR28: MOV #ESR29, @DEVECT
14627 052502 012777 000001 177244 MOV #1, @CSR1 ;INTR RQST BITS TO 6
14628 052510 012777 000013 177240 MOV #13, @CSR2
14629 052516 000240 NOP
14630 052520 012777 000012 177230 MOV #12, @CSR2 ;INTR SHOULD BE HONORED
14631 052526 000240 NOP
14632 052530 104000 EMT ;OTHERWISE AN ERROR
14633 052532 012777 052566 177206 ESR29: MOV #ESR31, @DEVECT ;ALTER CPU INTR VECTOR
14634 052540 106427 000300 MTPS #300
14635 052544 012777 000013 177204 MOV #13, @CSR2 ;SETS GO, CLRS DONE
14636 052552 000240 NOP
14637 052554 012777 000000 177174 MOV #0, @CSR2
14638 052562 000240 NOP
14639 052564 000401 BR ESR30
14640 052566 104000 ESR31: EMT ;INTR SHOULD NOT BE HONORED
14641 ; BUT IF HERE, IT WAS
14642 ;*****
14643 ; GENERATE AN INTERRUPT AT LEVEL SEVEN
14644 ;*****
14645 052570 012777 052626 177150 ESR30: MOV #ESR33, @DEVECT
14646 052576 012777 000001 177150 MOV #1, @CSR1 ;CPU AT LEVEL 6, Q22 AT 7
14647 052604 012777 000033 177144 MOV #33, @CSR2
14648 052612 000240 NOP
14649 052614 012777 000032 177134 MOV #32, @CSR2
14650 052622 000240 NOP
14651 052624 104000 EMT
14652 052626 012777 052702 177112 ESR33: MOV #ESR35, @DEVECT ;MODIFY INTR VECTOR IN CPU
14653 052634 106427 000340 MTPS #340
14654 052640 012777 000033 177110 MOV #33, @CSR2
14655 052646 000240 NOP
14656 052650 012777 000032 177100 MOV #32, @CSR2
14657 052656 000240 NOP
14658 052660 012777 000001 177070 MOV #1, @CSR2 ;THIS WILL REMOVE THE
14659 052666 012777 000000 177062 MOV #0, @CSR2 ;PENDING INTERRUPT
14660 052674 106427 000004 MTPS #4 ;RESTORE THE PSW
```

14661 052700 000401
14662 052702 104000
14663
14664
(2)
(3)
(2) 052704
14665
14666
14667
14668
14669
14670
14671
14672 052704 012777 065432 177052
14673 052712 012777 001601 177034
14674 052720 005077 177032
14675 052724 012777 177776 177030
14676 052732 012777 050752 177020
14677 052740 005067 176006
14678 052744 005067 176004
14679 052750 012777 000001 177014
14680 052756 032777 000200 176772
14681 052764 001001
14682 052766 000773
14683 052770 022767 065432 175754
14684 052776 001004
14685 053000 022767 065432 175746
14686 053006 001401
14687 053010 104000
14688
14689
14690
14691
(2)
(3)
(2) 053012
14692
14693
14694
14695
14696
14697
14698
14699
14700
14701 053012 000244
14702 053014 032777 000400 146554
14703 053022 001002
14704 053024 000167 000464
14705
14706 053030 016767 124770 175706
14707 053036 016767 124764 175702
14708 053044 012767 053116 124752
14709 053052 012767 000340 124746
14710 053060 010667 175666

```
BR Q22TS1 ;NADA SHOULD INTERRUPT
ESR35: EMT ;ERROR IF IT DOES

;*****
;TEST 425 DMA DATO TRANSFER
;*****
TS425:

;*****
;
; THIS TEST WILL SET UP AN ACTUAL DMA TRANSFER
;
;*****

Q22TS1: MOV #65432,BDATA
MOV #1601,BCSR1
CLR BCSR2 ;NO INTERRUPT BITS
MOV #177776,BWC ;TWO WORDS
MOV #CNTR,BBA
CLR CNTR
CLR CNTR+2 ;CLEAR THESE FIRST
MOV #1,BSIMGOA ;SIMULT. GO BIT
ESR26: BIT #BIT7,BCSR2 ;WAIT FOR DONE
BNE ESR25
BR ESR26
ESR25: CMP #65432,CNTR
BNE ESR32
CMP #65432,CNTR+2 ;GOOD DATA
BEQ Q22TS2
ESR32: EMT ;NO. BAD DATA

;*****
;TEST 426 TEST PWR OK LOGIC
;*****
TS426:

;*****
;
; THIS TEST WILL DETERMINE IF THE CPU WILL RESPOND
; TO THE BPOK LINE BEING ALTERED. THE Q22BE
; WILL BE USED TO CONTROL THAT LINE ON THE Q BUS.
;
;*****

Q22TS2: CLZ
BIT #BIT8,BSWR ;IS A Q BUS EXCER THERE
BNE ESR20 ;NO. IT'S A Q22
JMP BDVTST ;YES. THEN BYPASS THESE TESTS

ESR20: MOV 24,TMP
MOV 26,TMP1 ;SAVE FOR LATER
MOV #ESR12,24
MOV #340,26 ;SET UP NEW VECTOR
MOV SP,CNTR ;SAVE STACK POINTER
```

```
14711 053064 012706 000420          MOV    #420,SP          ;MOVE SP WAY DOWN
14712 053070 012767 053130 124706    MOV    #ESR13,4        ;WHERE A STK OVFL WLD GO
14713 053076 012767 000340 124702    MOV    #340,6
14714 053104 012777 000040 176644    MOV    #40,BCSR2      ;SET BIT 5 IN Q22BE CSR2
14715 053112 000240          NOP                    ;WILL PULL BPOK H LOW
14716 053114 104000          EMT                    ;DIDNT CAUSE CPU TO GO
14717 053116 012777 000000 176632 ESR12: MOV    #0,BCSR2    ;THRU LOC 24
14718 053124 000240          NOP                    ;BPOK H WILL GO HIGH
14719 053126 000401          BR     ESR14
14720 053130 104000          ESR13: EMT            ;IF HERE, AN ERROR BY GOING
14721          ;THRU LOC 4
14722 053132 016706 175614          ESR14: MOV    CNTR,SP    ;RESTORE SOME STUFF
14723 053136 016767 175602 124660    MOV    TMP,24
14724 053144 016767 175576 124654    MOV    TMP1,26
14725 053152 000167 000000          JMP    Q22TS3         ;GO TO NEXT TEST
14726
14727
14728
14729          ;*****
(2)          ;TEST 427 TEST INDIVIDUAL EXTENDED ADRS BITS
(3)          ;*****
(2) 053156 TS427:
14730          ;*****
14731          ;
14732          ; THIS TEST WILL UTILIZE THE Q22BE LATENCY CNTR
14733          ; TO CAPTURE THE EXTENDED ADDRESS BITS ON THE Q BUS.
14734          ; THE Q22BE LATENCY COUNTER BITS 15-12 CORRESPOND
14735          ; TO ADDRESS BITS 21-18.
14736          ;
14737          ;*****
14738
14739
14740 053156 016767 124622 175560 Q22TS3: MOV    4,TMP
14741 053164 016767 124616 175554          MOV    6,TMP1         ;STORE FOR LATER
14742 053172 012767 053270 124604          MOV    #ESR15,4
14743 053200 012767 000340 124600          MOV    #340,6        ;NEW VECTOR AND PSW
14744
14745 053206 017700 176554          MOV    @LATCNT,R0     ;READ IT TO CLR IT
14746 053212 012767 000020 117276    MOV    #BIT4,SR3      ; 22 BIT ADRSNG
14747 053220 012767 010000 117122    MOV    #10000,KIPAR4  ;SET FOR ADRS BIT 18
14748 053226 012767 010000 175520    MOV    #10000,CNTR+2
14749 053234 012767 000000 175510    MOV    #0,CNTR        ;ZERO THESE
14750 053242 012767 000001 124322    MOV    #BIT0,SRO      ;TURN ON MEM MANG
14751 053250 026727 175476 000017 ESR17: CMP    CNTR,#17   ;FINISHED ?
14752 053256 001433          BEQ    ESR16
14753 053260 105037 100000          CLRB   #100000        ;IF THERE IS NO MEMORY HERE
14754 053264 000240          NOP                    ;SHOULD HAVE TIMED OUT
14755 053266 000240          NOP                    ;WE ARE IN BIG TROUBLE
14756
14757 053270 017700 176472          ESR15: MOV    @LATCNT,R0 ;READ LATENCY COUNTER
14758 053274 042700 007777          BIC    #7777,R0       ;CLR DONT CARES
14759 053300 026700 175450          CMP    CNTR+2,R0      ;EXPECTED ?
14760 053304 001407          BEQ    ESR18          ;EQUALS RECVD
14761 053306 012767 000000 124256    MOV    #0,SRO         ;TURN OFF MM AND 22 BITS
14762 053314 012767 000000 117174    MOV    #0,SR3
14763 053322 104000          EMT                    ;DATA NOT GOOD
```

14764 053324 005267 175422
 14765 053330 062767 010000 117012
 14766 053336 062767 010000 175410
 14767 053344 000741
 14768
 14769 053346 016767 175372 124430
 14770 053354 016767 175366 124424
 14771 053362 012767 000000 124202
 14772 053370 012767 000000 117120
 14773 053376 000167 000112
 14774
 14775
 14776
 14777
 14778 053402 004767 060022
 14779 053406 012737 000011 001002
 14780 053414 012767 000001 125356
 14781 053422 032737 000001 001020
 14782 053430 001004
 14783 053432 012700 053444
 14784 053436 004767 057722
 14785 053442 000777
 14786 053444 040506 046111 051125
 053452 020105 052504 044522
 053460 043516 050440 041040
 053466 051525 042440 04 30
 053474 041522 051511 051105
 053502 052040 051505 051524
 053510 006412 000
 053514

14787
 14788
 14789
 14790
 14791
 14792
 14793
 14794
 14795
 14796
 14797
 14798
 14799
 14800
 (2)
 (3)
 (2) 053514
 14801
 14802
 14803
 14804 053514
 14805 053514 012767 000017 124002
 14806 053522 012737 000012 001004
 14807 053530 012737 054502 000030
 14808 053536 000244
 14809 053540 032777 002000 146030
 14810 053546 001410

ESR18: INC CNTR
 ADD #10000,KIPAR4
 ADD #10000,CNTR*2 ;INCREASE ADRS BITS BY ONE
 BR ESR17 ;CONTINUE

ESR16: MOV TMP,4
 MOV TMP1,6 ;RESTORE THIS VECTOR
 MOV #0,SRO ;TURN OFF MM AND 22 BITS
 MOV #0,SR3
 JMP BDVTST

ERRORB: JSR PC,ABORT ;ARE WE UNDER UFD ?
 MOV #11,#FATAL
 MOV #1,#MSGTY
 BIT #1,#ENV
 BNE Q22HLT
 MOV #Q22MSG,RO
 JSR PC,TYPE
 Q22HLT: BR
 Q22MSG: .ASCIZ /FAILURE DURING Q BUS EXERCISER TESTS/<12><15>

.EVEN

BDV TESTS

 ;TEST 430 TEST FOR FUNCTIONALITY OF R/W REGISTER
 ;*****
 TS430:

BDVTST: MOV #17,LSREG ;TURN OFF THE 4 LEDS
 MOV #12,#TESTN ;TST NUMBER FOR APT
 MOV #ERRORD,#30 ;ERROR TRAP
 CLZ
 BIT #BIT10,#SWR ;WANT TO TEST E102 ?
 BEQ ESR10 ;NO

14811	053550	016767	123750	175166		MOV	LSREG,TMP		;RD THIS TO GET SWITCHES
14812	053556	026727	175162	000317		CMP	TMP,#317		;SHOULD BE THIS VALUE
14813	053564	001401				BEQ	ESR10		;EQUAL
14814	053566	104000				EMT			
14815	053570	012737	054402	000030	ESR10:	MOV	#ERRORC,#30		;ERROR TRAP
14816	053576	005067	123720			CLR	RWREG		;THE NEXT FEW INSTRUCTIONS
14817	053602	016701	123714			MOV	RWREG,R1		;WILL TEST THE R/W ABILITY
14818	053606	020127	000000			CMP	R1,#0		;OF THE R/W MAINT. REGISTER
14819	053612	001401				BEQ	ESR40		
14820	053614	104000				EMT			
14821	053616	012767	177777	123676	ESR40:	MOV	#-1,RWREG		
14822	053624	016701	123672			MOV	RWREG,R1		
14823	053630	022701	177777			CMP	#177777,R1		
14824	053634	001401				BEQ	ESR41		
14825	053636	104000				EMT			
14826	053640	012767	125252	123654	ESR41:	MOV	#125252,RWREG		
14827	053646	016701	123650			MOV	RWREG,R1		
14828	053652	020127	125252			CMP	R1,#125252		
14829	053656	001401				BEQ	ESR42		
14830	053660	104000				EMT			
14831	053662	105067	123634		ESR42:	CLRB	RWREG		;BYTES
14832	053666	016701	123630			MOV	RWREG,R1		
14833	053672	020127	125000			CMP	R1,#125000		
14834	053676	001401				BEQ	ESR43		
14835	053700	104000				EMT			
14836	053702	000367	123614		ESR43:	SWAB	RWREG		
14837	053706	016701	123610			MOV	RWREG,R1		
14838	053712	020127	000252			CMP	R1,#252		
14839	053716	001401				BEQ	ESR44		
14840	053720	104000				EMT			
14841	053722	012767	052525	123572	ESR44:	MOV	#52525,RWREG		
14842	053730	016701	123566			MOV	RWREG,R1		
14843	053734	020127	052525			CMP	R1,#52525		
14844	053740	001401				BEQ	ESR45		
14845	053742	104000				EMT			
14846	053744				ESR45:				
14847	053744	105067	123553			CLRB	RWREG+1		
14848	053750	016701	123546			MOV	RWREG,R1		
14849	053754	020127	000125			CMP	R1,#125		
14850	053760	001401				BEQ	ESR46		
14851	053762	104000				EMT			
14852	053764	000367	123532		ESR46:	SWAB	RWREG		
14853	053770	016701	123526			MOV	RWREG,R1		
14854	053774	020127	052400			CMP	R1,#52400		
14855	054000	001401				BEQ	ESR47		
14856	054002	104000				EMT			
14857	054004	005067	123512		ESR47:	CLR	RWREG		
14858	054010	052767	100000	123504		BIS	#BIT15,RWREG		;SET 15
14859	054016	016701	123500			MOV	RWREG,R1		
14860	054022	020167	123474		ROTLP1:	CMP	R1,RWREG		;ARE THEY THE SAME
14861	054026	001401				BEQ	ESR48		
14862	054030	104000				EMT			
14863	054032	006001			ESR48:	ROR	R1		;NEXT BIT TO THE RIGHT
14864	054034	001403				BEQ	ESR49		
14865	054036	006067	123460			ROR	RWREG		
14866	054042	000767				BR	ROTLP1		;CONTINUE TESTING

14867 054044 012767 177777
 14868 054052 042767 100000
 14869 054060 016701 123436
 14870 054064 026701 123432
 14871 054070 001401
 14872 054072 104000
 14873 054074 000261
 14874 054076 006067 123420
 14875 054102 006001
 14876 054104 020127 077777
 14877 054110 001365
 14878 054112 000431
 14879
 14880
 14881
 (2)
 (3)
 (2) 054114
 14882
 14883
 14884
 14885
 14886
 14887
 14888
 14889
 14890
 14891
 14892
 14893
 14894
 14895
 14896
 14897
 14898 054114 000000
 14899 054116 000000
 14900 054120 000000
 14901 054122 000001
 14902 054124 000000
 14903 054126 000000
 14904 054130 000000
 14905
 14906
 14907
 14908
 14909
 14910
 14911
 14912
 14913
 14914
 14915 054132 012701 173776
 14916 054136 066701 177754
 14917 054142 005067 177762
 14918 054146 012702 173000
 14919 054152 066702 177740

ESR49: MOV # -1, R15, R15REG
 BIC #BIT15, R15REG
 MOV R15REG, R1
 ROTLP2: CMP R15REG, R1 ; SAME ?
 BEQ ESR50 ; YES
 EMT
 ESR50: SEC ; SET THE C BIT
 ROR R15REG
 ROR R1
 CMP R1, #077777
 BNE ROTLP2 ; NOT FINISHED YET
 BR BDVTS2

;.....
 ;TEST 431 ROM CHECKSUM TEST
 ;.....
 TS431:

;.....
 ;
 ; THE PREVIOUS TEST CHECKED OUT THE ROM R/W REGISTER
 ; NOW WE WILL TEST THE BK DIAGNOSTIC ROM FOR THE
 ; CORRECT CHECKSUM AND CHECKWORD
 ;
 ;.....

;.....
 ;
 ; DATA SECTION FOR THE NEXT TEST
 ;
 ;.....

VRTPCR: .WORD 0 ;VIRTUAL PAGE CONTROL REGISTER
 BCF: .WORD 0
 COUNTR: .WORD 0
 ANSR: .WORD 1
 RFLAG: .WORD 0
 EXPSUM: .WORD 0
 ACTSUM: .WORD 0

;.....
 ;FUNCTIONAL DESCRIPTION:
 ;SUBROUTINE TO COMPUTE A CHECKSUM IN A ROM/EPROM
 ;INPUT: CONTENTS OF BCF
 ;IMPLICIT INPUTS: CONTENTS OF PCR
 ;OUTPUT: A CHECKSUM VALUE STORED IN LOCATION ACTSUM
 ;CALLING SEQUENCE: JSR PC,CHKSUM
 ;.....

CHKSUM: MOV #173776, R1 ;STORE THE HIGHEST ADDRESS IN THE ROM
 ADD BCF, R1 ;FOR EITHER LOW OR HIGH BYTES
 CLR ACTSUM ;CLEAR LOCATION WHICH WILL HOLD THE CHECKSUM
 MOV #173000, R2 ;COMPUTE THE LOWEST ADDRESS IN THE ROM
 ADD BCF, R2 ;WHERE THE DATA WILL START

14920 054156 111204
 14921 054160 060467 177744
 14922 054164 062702 000002
 14923 054170 020201
 14924 054172 002771
 14925 054174 000207
 14926
 14927
 14928
 14929
 14930
 14931
 14932
 14933
 14934
 14935 054176 012767 000400 000172
 14936 054204 016767 000166 177702
 14937 054212 016767 177676 123300
 14938 054220 012767 000040 177672
 14939 054226 012767 000001 177670
 14940 054234 005067 177656
 14941 054240 122737 177777 173774
 14942 054246 001001
 14943 054250 104000
 14944 054252 004767 177654
 14945 054256 113767 173776 177642
 14946 054264 066767 177640 177634
 14947 054272 105767 177630
 14948 054276 001401
 14949 054300 104000
 14950 054302 012767 000001 177606
 14951 054310 122737 177777 173775
 14952 054316 001001
 14953 054320 104000
 14954 054322 004767 177604
 14955 054326 113767 173777 177572
 14956 054334 066767 177570 177564
 14957 054342 105767 177560
 14958 054346 001401
 14959 054350 104000
 14960 054352 062767 001002 177534
 14961 054360 016767 177530 123132
 14962 054366 005367 177526
 14963 054372 001320
 14964
 14965
 14966
 14967
 14968
 14969 054374 000510
 14970
 14971
 14972
 14973
 14974 054376 000000
 14975 054400 000000

```

18:  MOVB  (R2),R4      ;GET DATA IN BYTES
     ADD   R4,ACTSUM   ;ADD CONTENTS OF EACH LOCATION TO THE CHECKSUM
     ADD   #2,R2       ;ADJUST ADDRESS
     CMP   R2,R1       ;COMPARE CURRENT ADDRESS WITH HIGHEST ADDRESS
     BLT  18           ;BR IF LESS THAN
     RTS   PC          ;RETURN
  
```

```

;*****
;TEST TO PERFORM CHECKSUM AND CHECKWORD VERIFICATION ON THE 8K
;OF DIAGNOSTIC ROM. IN UNATTENDED MODE, THE ROM WILL BE ADDRESSED
;FROM 0-8K. IN STAND-ALONE MODE, THE OPERATOR MAY CHANGE THE
;ADDRESS BY RESPONDING TO QUESTIONS GENERATED ON THE FIRST PASS.
;*****
  
```

```

BOVTS2: MOV   #400,DRLP      ;STORE STARTING ADDRESS
        MOV   DRLP,VRTPCR   ;SET UP PCR
        MOV   VRTPCR,PCR
        MOV   #40,COUNTR    ;SET NUMBER OF CHECKWORDS TO CHECK
        MOV   #1,RFLAG     ;INDICATE ROM
DLOOP:  CLR   BCF           ;SIGNAL LOW BYTES ARE BEING CHECKED
        CMPB #1,#173774    ;DOES THE ROM EXIST?
        BNE  18           ;BR IF YES
        EMT
18:     JSR   PC,CHKSUM      ;COMPUTE THE ACTUAL CHECKSUM
        MOVB #173776,EXPSUM ;GET THE STORED CHECKSUM
        ADD  ACTSUM,EXPSUM  ;ADD THE EXPECTED AND ACTUAL CHECKSUMS
        TSTB EXPSUM        ;BYTE RESULT = 0?
        BEQ  28           ;BR IF YES
        EMT
28:     MOV   #1,BCF        ;SET BCF TO DENOTE HIGH BYTES
        CMPB #1,#173775    ;DOES THE ROM EXIST?
        BNE  38           ;BR IF YES
        EMT
38:     JSR   PC,CHKSUM      ;COMPUTE THE ACTUAL CHECKSUM
        MOVB #173777,EXPSUM ;GET EXPECTED CHECKSUM
        ADD  ACTSUM,EXPSUM  ;ADD THE EXPECTED AND ACTUAL CHECKSUMS
        TSTB EXPSUM        ;BYTE RESULT = 0?
        BEQ  48           ;BR IF YES
        EMT
48:     ADD  #1002,VRTPCR    ;NEXT PAGE IN PCR
        MOV  VRTPCR,PCR
        DEC  COUNTR        ;DECREMENT CHECKWORD COUNT
        BNE DLOOP         ;LOOP UNTIL ALL 20 PAGES HAVE BEEN CHECKED
  
```

```

;
; GO DO THE FLOATING POINT TESTS
;
  
```

```

COMPLE: BR    FPSTRT
  
```

```

DRLP:  .WORD  0
PNTR:  .WORD  0      ;WILL BE USED AS A POINTER
  
```

14976
14977
14978
14979
14980
14981
14982
14983 054402 004767 057022
14984 054406 012737 000012 001002
14985 054414 012767 000001 124356
14986 054422 032737 000001 001020
14987 054430 001004
14988 054432 012700 054444
14989 054436 004767 056722
14990 054442 000777
14991
14992 054444 040506 046111 042105
054452 042040 051125 047111
054460 020107 044124 020105
054466 042102 020126 042524
054474 052123 005123 000013

```

;*****
;
;      ERROR ROUTINE FOR THE BDV TESTING
;
;*****
ERRORC: JSR      PC,ABORT                ;ARE WE UNDER UFD ?
        MOV      #12,#FATAL
        MOV      #1,#MSGTY
        BIT      #1,#ENV                ;UNDER APT ?
        BNE      BDVHLT
        MOV      #BDVMSG,R0
        JSR      PC,TYPE
BDVHLT: BR
BDVMSG: .ASCIZ  /FAILED DURING THE BDV TESTS/<12><15>

```

14993
14994
14995 054502 004767 056722
14996 054506 012737 000012 001002
14997 054514 012767 000001 124256
14998 054522 032737 000001 001020
14999 054530 001344
15000 054532 012700 054544
15001 054536 004767 056622
15002 054542 000000
15003 054544 044103 041505 020113
054552 053523 052111 044103
054560 051505 047440 020116
054566 030505 031060 020054
054574 042522 052123 051101
054602 020124 052101 031040
054610 030060 006412 000

```

ERRORD: JSR      PC,ABORT                ;ARE WE UNDER UFD ?
        MOV      #12,#FATAL
        MOV      #1,#MSGTY
        BIT      #1,#ENV                ;UNDER APT ?
        BNE      BDVHLT
        MOV      #SWMSG,R0
        JSR      PC,TYPE
        HALT
SWMSG:  .ASCIZ  /CHECK SWITCHES ON E102. RESTART AT 200/<12><15>

```

15004
15005
15006
15014
15015
15016
15023
15024
15031
15047
15076
15077 000244
15078
15079 000000
15080 000001
15081 000002
15082 000003
15083 000004

```

.EVEN
;* *****
FPVECT=244
.SBTTL  FPP REGISTER DEFINITIONS
AC0     =#0
AC1     =#1
AC2     =#2
AC3     =#3
AC4     =#4

```


15084 000005
15085 000006
15086 000007

AC5 =#5
AC6 =#6
AC7 =#7

15088
15089 054616 012706 001000
15090 054622 000244
15091 054624 032777 000002 144744
15092 054632 001002
15093 054634 000167 047776
15094 054640 012737 124452 000030
15095 054646 012737 000003 001004

FPSTRT: MOV #STBOT,SP ;SET UP STACK POINTER
CLZ
BIT #2,@SWR
BNE 1#
JMP SLU1ST
1#: MOV #ERROR4,@#30 ;SETUP FOR CORRECT ERROR CALL
MOV #3,@#TESTN ;PUT TEST NUMBER IN MAILBOX

15096
15097
15108 ;*****
(2) ;TEST 432 LDFPS, STFPS AND DATA PATHS TEST
(3) ;*****

(2) 054654
15109 054654 012700 177777
15110 054660 012737 054732 000244
15111 054666 012737 054732 000010
15112 054674 012737 054732 000004

TS432: MOV #-1,R0 ;INITIALIZE THE COUNT PATTERN.
MOV #AERR1,@FPVECT ;SET UP FOR UNABLE TO DECODE
MOV #AERR1,@#10 ;FPP INSTRUCTION TRAP TO 244 OR 10.
MOV #AERR1,@ERRVECT ;IF EITHER INSTRUCTION
;FAILS TO GO THROUGH THE
;CORRECT SRC OR DST MODE AN
;ODD ADDRESS TRAP WILL OCCUR.

15113
15114
15115
15116 054702
15117 054702 010004
15118 054704 042704 030020
15119 054710 170104
15120
15121 054712 012701 177777
15122 054716 170201
15123 054720 010004
15124 054722 042704 030020
15125 054726 020401
15126

A1:
A11: MOV R0,R4
BIC #30020,R4
LDFPS R4 ;TEST INSTRUCTION.

MOV #-1,R1
A12: STFPS R1 ;TEST INSTRUCTION.
MOV R0,R4 ;MASK OFF UNSETTABLE BITS.
BIC #30020,R4
CMP R4,R1 ;COMPARE DATA EXPECTED WITH
;THE DATA READ.

BEQ A2
AERR1: EMT ;

A2: MOV #1,R0 ;NEXT PATTERN WILL BE ALL ZERO
SOB R0,A1 ;DECREMENT COUNT PATTERN

15127 054730 001401
(1) 054732
(2) 054732 104000
15128
15129 054734 012700 000001
15130 054740 077020
15131 054742
(1) 054742 004767 047604
(1)
(1)
(1)
(1)
(1)

ADONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

15132
15133
15139
(2)
(3)
(2) 054746
15140 054746 012700 000017
15141

;*****
;TEST 433 CFCC TEST
;*****
TS433: MOV #17,R0 ;R0 CONTAINS TO TEST PATTERN.

```
15142 054752
15143 054752 170100
15144
15145 054754
15146 054754 170000
15147
15148 054756 013703 177776
15149 054762 042703 177760
15150 054766 020003
15151 054770 001401
(2) 054772 104000
15152 054774 077012
15153 054776
(1) 054776 004767 047550
(1)
(1)
(1)
(1)
15154
15165
(2)
(3)
(2) 055002
15166 055002 005000
15167
15168 055004 170100
15169 055006 170001
15170
15171 055010 170201
15172 055012 005002
15173 055014 020201
15174 055016 001401
(2) 055020 104000
15175 055022 012700 147757
15176
15177 055026 170100
15178 055030 170001
15179
15180 055032 170201
15181 055034 012702 147557
15182 055040 020102
15183 055042 001401
(2) 055044 104000
15184 055046 012700 147757
15185
15186 055052 170100
15187 055054 170011
15188
15189 055056 170201
15190 055060 012702 147757
15191 055064 020102
15192 055066 001401
(2) 055070 104000
15193 055072 005000
15194 055074 170100
15195 055076 170011
```

```

B1:          LDFPS  R0          ;LOAD THE TEST PATTERN
B2:          CFCC              ;COPY CONDITION CODES.
              MOV      @#PSW,R3  ;SEE IF PATTERN TRANSFERED.
              BIC     #177760,R3
              CMP     R0,R3
              BEQ     B3
              EMT
B3:          SOB      R0,B1
BDONE:       JSR      PC,.RSET   ;GO INITIALIZE THE FPS AND STACK; AND
                                      ;SEE IF THE USER HAS EXPRESSED
                                      ;THE DESIRE TO CHANGE THE SOFTWARE
                                      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                      ;THE USER TYPED CONTROL G?).
;*****
;TEST 434      SETF, SETD, SETI AND SETL TEST
;*****
TS434:
              CLR     R0
C15:         LDFPS  R0          ;CLEAR THE FPS.
              SETF   ;TEST INSTRUCTION.
              STFPS  R1          ;GET RESULT.
              CLR   R2
              CMP   R2,R1        ;DID AN ERROR OCCUR?
              BEQ   C2
              EMT
C2:          MOV    #147757,R0
C25:         LDFPS  R0          ;PUT 147757 IS FPS
              SETF   ;CLEAR FD BIT.
              STFPS  R1          ;GET RESULT
              MOV    #147557,R2
              CMP   R1,R2        ;RESULT CORRECT.
              BEQ   C3
              EMT
C3:          MOV    #147757,R0
C35:         LDFPS  R0          ;LOAD 147757 INTO FPS.
              SETD   ;SETD FD BIT.
              STFPS  R1
              MOV    #147757,R2
              CMP   R1,R2        ;RESULT CORRECT?
              BEQ   C4
              EMT
C4:          CLR    R0
              LDFPS R0
C45:         SETD
```

```

15196
15197 055100 170201          STFPS  R1          ;GET RESULT.
15198 055102 012702 000200  MOV    #200,R2
15199 055106 020102          CMP    R1,R2      ;RESULT CORRECT?
15200 055110 001401          BEQ    C5
      (2) 055112 104000          EMT
15201 055114 005000          CS:  CLR    R0
15202
15203 055116 170100          LDFPS  R0          ;CLEAR FPS
15204 055120 170002          C55: SETI          ;CLEAR FL BIT.
15205
15206 055122 170201          STFPS  R1          ;GET RESULT.
15207 055124 005002          CLR    R2
15208 055126 020201          CMP    R2,R1      ;RESULT CORRECT?
15209 055130 001401          BEQ    C6
      (2) 055132 104000          EMT
15210 055134 012700 147777  C6:  MOV    #147757,R0
15211 055140 170100          LDFPS  R0          ;PUT 147757 INTO FPS
15212 055142 170002          C65: SETI          ;CLEAR FL BIT.
15213
15214 055144 170201          STFPS  R1          ;GET THE RESULT.
15215 055146 012702 147657  MOV    #147657,R2
15216 055152 020102          CMP    R1,R2      ;RESULT CORRECT?
15217 055154 001401          BEQ    C7
      (2) 055156 104000          EMT
15218 055160 012700 147757  C7:  MOV    #147757,R0
15219 055164 170100          LDFPS  R0          ;SET FPS TO 147757.
15220 055166 170012          C75: SETL          ;SET FL BIT.
15221
15222 055170 170201          STFPS  R1          ;GET THE RESULT.
15223 055172 012702 147757  MOV    #147757,R2
15224 055176 020102          CMP    R1,R2      ;RESULT CORRECT?
15225 055200 001401          BEQ    C8
      (2) 055202 104000          EMT
15226 055204 005000          C8:  CLR    R0
15227 055206 170100          LDFPS  R0          ;CLEAR FPS.
15228 055210 170012          C85: SETL          ;SET FL BIT.
15229
15230 055212 170201          STFPS  R1
15231 055214 012702 000100  MOV    #100,R2
15232 055220 020102          CMP    R1,R2      ;RESULT CORRECT.
15233 055222 001401          BEQ    CDONE
      (2) 055224 104000          EMT
15234 055226 004767 047320  CDONE: JSR    PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
      (1)
      (1) ;SEE IF THE USER HAS EXPRESSED
      (1) ;THE DESIRE TO CHANGE THE SOFTWARE
      (1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1) ;THE USER TYPED CONTROL G?).
15235
15236
15256 ;*****
      (2) ;TEST 435      ILLEGAL FPP OP CODES AND STST TEST
      (3) ;*****
      (2) 055232
15257 055232 012705 170003  TS435: MOV    #170003,R5 ;INITIAL OP CODE.

```

```

15258 055236 012737 055276 000004      MOV      @DERR2,@#ERRVECT
15259 055244 012737 055330 000244      MOV      @DERR1,@#FPVECT
15260
15261 055252 005000      D1:      CLR      R0
15262 055254 170100      LDFPS   R0          ;CLEAR FPS.
15263 055256 005002      CLR      R2
15264 055260 010537 055264      MOV      R5,@#D2    ;SET UP THE ILLEGAL INSTRUCTION.
15265 055264 000000      D2:      .WORD   0
15266 055266 170000      D3:      CFCC
15267 055270 005202      INC      R2
15268 055272 005202      D4:      INC      R2
15269
15270 055274 170201      STFPS   R1          ;REPORT FAILURE. DID NOT TRAP.
15271 055276
(2) 055276 104000      DERR2:
15272 055300 022705 170010      D5:      EMT
          CMP      @170010,R5      ;
          BNE     D6              ;COMPUTE NEXT OP CODE
15273 055304 001003      BNE     D6
15274 055306 012705 170013      MOV      @170013,R5
15275 055312 000757      BR      D1
15276
15277 055314 022705 170077      D6:      CMP      @170077,R5
15278 055320 001001      BNE     D7
15279 055322 000424      BR      DDONE
15280 055324 005205      D7:      INC      R5
15281 055326 000751      BR      D1
15282
15283 055330 022716 055266      DERR1:  CMP      @D3,(SP)          ;DID TRAP OCCUR ON TEST INSTRUCTION?
15284 055334 001401      BEQ     14
(2) 055336 104000      EMT
15285 055340 022626      14:     CMP      (SP)+,(SP)+      ;
15286 055342 170201      STFPS   R1          ;GET THE FPS AND SEE IF IT IS
15287 055344 022701 100000      CMP      @100000,R1      ;SET CORRECTLY.
15288 055350 001401      BEQ     34
(2) 055352 104000      EMT
15289 055354 012704 000001      34:     MOV      @1,R4
15290 055360 170304      D8:     STST   R4          ;GET THE FEC CODE. NOTE THAT
15291                                     ;IF THE DESTINATION MODE IS
15292                                     ;IMPROPERLY DECODED AN ODD
15293                                     ;ADDRESS TRAP TO 4 SHOULD OCCUR.
15294 055362 022704 000002      CMP      @2,R4          ;WAS FEC CORRECT?
15295 055366 001001      BNE     D9
15296 055370 000743      BR      D5
15297
15298 055372      D9:     ;REPORT STST FAILURE
15299 055372 104000      EMT
15300 055374      DDONE:
(1) 055374 004767 047152      JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
(1)                                     ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
15301
15309      ;*****
(2)      ;TEST 436      FID, INTERRUPT DISABLE, BIT TEST
(3)      ;*****
(2) 055400      TS436:

```

```

15310 055400 012737 055440 000244      MOV     @EERR0,@#FVVECT      ;SETUP FOR THE INTERRUPT.
15311
15312 055406 012700 040000      E1:    MOV     @40000,R0
15313 055412 170100                LDFPS  R0                    ;SET FID.
15314 055414 170020                E3:    .WORD 170020          ;ILLEGAL FPP INSTRUCTION.
15315 055416 170000                E4:    CFCC
15316
15317 055420 170201                STFPS  R1                    ;SEE IF ERROR WAS DETECTED
15318 055422 022701 140000      CMP     @140000,R1
15319 055426 001004                BNE    EERR0
15320
15321 055430 170304                STST   R4                    ;SEE IF FEC=2
15322 055432 022704 000002      CMP     @2,R4
15323 055436 001401                BEQ    EDONE
15324
(1) 055440                EERR0: EMT                    ;
(2) 055440 104000                EDONE: JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
(1) 055442 004767 047104                ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
15325
15326
15339
(2)
(3)
(2) 055446
15340
15341 055446 005000                CLR     R0
15342 055450 170100                LDFPS  R0
15343 055452 170011                SETD
15344 055454 012701 055716      MOV     @FDAT10,R1          ;SET UP THE LOAD DATA.
15345 055460 012702 055762      MOV     @FXDAT0,R2
15346 055464 012703 000010      MOV     @10,R3
15347
15348 055470 012221                F2:    MOV     (R2)+,(R1)+
15349 055472 077302                SOB    R3,F2
15350
15351 055474 012700 055726      MOV     @FDAT14,R0          ;SETUP R0 FOR THE LDD (R0),AC0.
15352 055500 012737 055714 000004  MOV     @FERR20,@#ERRVECT ;IF THE SRC FLOWS FAIL THEN
15353                                     ;AN ODD ADDRESS MAY OCCUR.
15354 055506 005003                CLR     R3
15355
15356 055510 172410                F3:    LDD     (R0),AC0
15357 055512 005203                F4:    INC    R3
15358 055514 005203                INC    R3
15359
15360 055516 020027 055726      CMP     R0,@FDAT14          ;WAS R0 AFFECTED?
15361 055522 001401                BEQ    F5
(2) 055524 104000                EMT
15362 055526 020327 000002      F5:    CMP     R3,@2          ;SEE IF THE PC WAS ADVERSELY
15363 055532 001401                BEQ    1$
(2) 055534 104000                EMT
15364 055536 012701 055716      1$:    MOV     @FDAT10,R1      ;MAKE SURE THE SOURCE DATA WAS
15365 055542 012702 055762      MOV     @FXDAT0,R2          ;NOT AFFECTED.

```

```

;*****
;TEST 437      LDD AND STD, WITH SRC AND DST MODE 1, TEST
;*****
TS437:

```

15366	055546	012703	000010		MOV	#10,R3	
15367	055552	022122		24:	CMP	(R1),.(R2).	
15368	055554	001401			BEQ	34	
(2)	055556	104000			EMT		
15369	055560	077304		34:	SQB	R3,24	
15370							
15371	055562	170201			STFPS	R1	;MAKE SURE THE FPS IS CORRECT.
15372	055564	022701	000200		CMP	#200,R1	
15373	055570	001401			BEQ	F6	
(2)	055572	104000			EMT		
15374	055574	012703	177777	F6:	MOV	#-1,R3	
15375	055600	012704	000010		MOV	#10,R4	
15376	055604	012705	055740		MOV	#FDAT00,R5	;SET UP THE OUTPUT DATA BUFFER.
15377	055610	010325		F7:	MOV	R3,(R5).	
15378	055612	077402			SQB	R4,F7	
15379							
15380	055614	012700	055750		MOV	#FDAT04,R0	;SET UP R0 FOR DST MODE 1 REG 0.
15381	055620	012737	055714	000004	MOV	#FERR20,#ERRVECT	;IF THE DST FLOWS FAIL AN ODD ;ADDRESS COULD OCCUR.
15382							
15383	055626	005003			CLR	R3	
15384							
15385	055630	174010		F10:	STD	AC0,(R0)	;TEST INSTRUCTION.
15386	055632	005203		F11:	INC	R3	
15387	055634	005203			INC	R3	
15388							
15389	055636	020027	055750		CMP	R0,#FDAT04	;WAS R0 MODIFIED?
15390	055642	001401			BEQ	F12	
(2)	055644	104000			EMT		
15391	055646	020327	000002	F12:	CMP	R3,#2	;WAS THE PC AFFECTED CORRECTLY?
15392	055652	001401			BEQ	F135	
(2)	055654	104000			EMT		
15393	055656	012701	055740	F135:	MOV	#FDAT00,R1	
15394	055662	012702	055762		MOV	#FXDAT0,R2	
15395	055666	012703	000010		MOV	#10,R3	;SETUP LOOP COUNT
15396	055672	022122		F13:	CMP	(R1),.(R2).	;WAS DATA OUTPUT CORRECTLY
15397	055674	001401			BEQ	F14	
(2)	055676	104000			EMT		
15398	055700	077304		F14:	SQB	R3,F13	;SUBTRACT 1 FROM LOOP COUNT AND LOOP IF NOT ZERO
15399	055702	005001		F22:	CLR	R1	
15400	055704	170201			STFPS	R1	;MAKE SURE FPS IS CORRECT.
15401	055706	022701	000200		CMP	#200,R1	
15402	055712	001433			BEQ	FDONE	
(1)	055714			FERR20:			
(2)	055714	104000			EMT		
15403							
15404	055716	177777		FDATIO:	-1		
15405	055720	177777		FDATI1:	-1		
15406	055722	177777		FDATI2:	-1		
15407	055724	177777		FDATI3:	-1		
15408	055726	177777		FDATI4:	-1		
15409	055730	177777		FDATI5:	-1		
15410	055732	177777		FDATI6:	-1		
15411	055734	177777		FDATI7:	-1		
15412	055736	177777			-1		
15413	055740	177777		FDAT00:	-1		
15414	055742	177777		FDAT01:	-1		

CJKL580 LCP-5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 20-28
T437 LDD AND STD, WITH SRC AND DST MODE 1, TEST

SEQ 0244

15415 055744 177777
 15416 055746 177777
 15417 055750 177777
 15418 055752 177777
 15419 055754 177777
 15420 055756 177777
 15421 055760 177777
 15422 055762 177777
 15423 055764 177777
 15424 055766 177777
 15425 055770 177777
 15426 055772 052525
 15427 055774 031463
 15428 055776 007417
 15429 056000 000477
 15430
 15431
 15432 056002
 (1) 056002 004767 046544
 (1)
 (1)
 (1)
 (1)
 15433
 15434
 15440
 (2)
 (3)
 (2) 056006
 15441 056006
 15442 056006 170011
 15443 056010 012700 056274
 15444 056014 012701 056244
 15445 056020 012702 000004
 15446 056024 012120
 15447 056026 077202
 15448
 15449 056030 012700 056274
 15450 056034 172510
 15451
 15452 056036 012700 056254
 15453 056042 172410
 15454
 15455 056044 012701 000001
 15456 056050 172401
 15457 056052 000240
 15458 056054 000240
 15459
 15460 056056 012700 056264
 15461 056062 174010
 15462
 15463 056064 012700 056264
 15464 056070 012701 056274
 15465 056074 012702 000004
 15466 056100 022021
 15467 056102 001401

FDATE2: 1
 FDATE3: 1
 FDATE4: -1
 FDATE5: -1
 FDATE6: 1
 FDATE7: 1
 -1
 FXDATE: -1
 FXDATE1: -1
 FXDATE2: -1
 FXDATE3: -1
 FXDATE4: 052525
 FXDATE5: 031463
 FXDATE6: 007417
 FXDATE7: 000477

FDONE: JSR PC, RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 440 FSRC MODE 0 TEST

TS440:
 I1: SETD ;SET FD.
 MOV #IDATIO,R0
 MOV #IPATIO,R1
 MOV #4,R2
 I2: MOV (R1)+,(R0)+ ;SET UP THE INPUT DATA BUFFER.
 SOB R2,I2
 MOV #IDATIO,R0 ;LOAD AC1
 LDD (R0),AC1
 MOV #IPAT20,R0 ;LOAD ACO
 LDD (R0),ACO
 MOV #1,R1 ;IN CASE THE FSRC FLOWS FAIL
 I3: LDD AC1,ACO ;TEST INSTRUCTION.
 I4: NOP
 I5: NOP
 MOV #IDATIO,R0 ;GET ACO, THE RESULTS.
 STD ACO,(R0)
 MOV #IDATIO,R0 ;SEE IF DATA IS CORRECT.
 MOV #IDATIO,R1
 MOV #4,R2
 I6: CMP (R0)+,(R1)+
 BEQ I105

```

(2) 056104 104000
15468 056106 077204
15469
15470
15471
15472 056110 012700 056244
15473 056114 012701 056274
15474 056120 012702 000004
15475 056124 012021
15476 056126 077202
15477
15478 056130 012700 056274
15479 056134 172510
15480
15481 056136 012700 056254
15482 056142 172410
15483
15484 056144 012701 000001
15485 056150 170001
15486
15487 056152 172401
15488 056154 000240
15489 056156 000240
15490
15491 056160 170200
15492 056162 022700 000004
15493 056166 001401
(2) 056170 104000
15494 056172
15495 056172 170011
15496
15497 056174 012700 056264
15498 056200 174010
15499
15500 056202 012737 177777 056300
15501 056210 012737 177777 056302
15502 056216 012700 056264
15503 056222 012701 056274
15504 056226 012702 000004
15505 056232 022021
15506 056234 001401
(2) 056236 104000
15507 056240 077204
15508 056242 000420
15509
15510 056244 000000
15511 056246 170360
15512 056250 016161
15513 056252 052525
15514
15515 056254 177777
15516 056256 177777
15517 056260 177777
15518 056262 177777
15519
15520 056264 000000

```

```

EMT ;
I105: SOB R2,I6 ;
;NOW TEST THE LOAD INSTRUCTION WITH FSRC MODE ZERO AND FD CLEAR.
I12: MOV #IPAT10,R0
MOV #IDATIO,R1
MOV #4,R2
I13: MOV (R0)+,(R1)+
SOB R2,I13
MOV #IDATIO,R0 ;SET UP AC1
LDD (R0),AC1
MOV #IPAT20,R0 ;SET UP ACO
LDD (R0),ACO
MOV #1,R1
SETF ;CLEAR FD.
I14: LDF AC1,ACO ;TEST INSTRUCTION.
I15: NOP
I16: NOP
STFPS R0 ;SEE IF FPS IS STILL CLEAR.
CMP #4,R0
BEQ I17
EMT ;
I17: SETD ;RESET TO DOUBLE MODE.
MOV #IDAT00,R0
STD ACO,(R0) ;GET ACO
MOV #-1,#IDATI2
MOV #-1,#IDATI3
MOV #IDAT00,R0
MOV #IDATIO,R1
MOV #4,R2
I20: CMP (R0)+,(R1)+ ;SEE IF ACO WAS CORRECT.
BEQ I23
EMT ;
I23: SOB R2,I20
BR IDONE ;NO ERRORS.
IPAT10: 0
IPAT11: 170360
IPAT12: 016161
IPAT13: 052525
IPAT20: -1
IPAT21: -1
IPAT22: -1
IPAT23: -1
IDAT00: 0

```


15521 056266 000000
 15522 056270 000000
 15523 056272 000000
 15524
 15525 056274 000000
 15526 056276 000000
 15527 056300 000000
 15528 056302 000000
 15529
 15530 056304
 (1) 056304 004767 046242
 (1)
 (1)
 (1)
 (1)
 15531
 15537
 15538
 (2)
 (3)
 (2) 056310
 15539 056310 170011
 15540 056312 012700 056550
 15541 056316 012701 056600

IDAT01: 0
 IDAT02: 0
 IDAT03: 0
 IDATIO: 0
 IDATI1: 0
 IDATI2: 0
 IDATI3: 0

IDONE:

JSR PC..RSET

;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 441 FDST MODE 0 TEST
 ;*****
 TS441:

SETD ;SET FD
 MOV @TPAT10,R0
 MOV @TDATIO,R1

B4

CJKLSBO LCP 5 CPU CLSTR DIAG
CJKLSB.P11 07 JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 21
T441 FDST MODE 0 TEST

SFO 0247

15543 056322 012702 000004

MOV 04.R2

CJKLS80 LCP 5 CPU CLSTR DIAG
CJKLS8.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 22
T441 FDST MODE 0 TEST

C4

SEG 0248

15545 056326 012021
15546 056330 077202
15547

T2: MOV (R0)..(R1).
SOB R2,T2

;SET UP THE INPUT DATA BUFFER.

CJKL580	LCP-5	CPU	CLSTR	DIAG	MACY11	30(1046)	07-JAN-85	09:28	PAGE 23
CJKL58	P11	07	JAN-85	09:05	T441		FDST	MODE 0	TEST
15549	056332	012700	056600				MOV	@TDATIO,R0	;LOAD ACO
15550	056336	172410					LDD	(R0),ACO	
15551									
15552	056340	012700	056560				MOV	@TPAT20,R0	;LOAD AC1
15553	056344	172510					LDD	(R0),AC1	
15554									
15555	056346	012701	000001				MOV	@1,R1	;IF THE (BUT FDST) FORK FAILS
15556	056352	174001			T3:		STD	ACO,AC1	
15557	056354	000240			T4:		NOP		
15558	056356	000240			T5:		NOP		
15559									
15560	056360	012700	056570				MOV	@TDAT00,R0	
15561	056364	174110					STD	AC1,(R0)	;GET THE DATA.
15562									
15563	056366	012703	056570				MOV	@TDAT00,R3	;SEE IF THE DATA IS CORRECT.
15564	056372	012704	056600				MOV	@TDATIO,R4	
15565	056376	012705	000004				MOV	@4,R5	
15566	056402	022324			T6:		CHP	(R3)*,(R4).	
15567	056404	001401					BEG	T105	
(2)	056406	104000					ENT		
15568	056410	077504			T105:		SOB	R5.T6	
15569									
15570									;NOW TEST THE STF ACO,AC1 INSTRUCTION.
15571									
15572	056412	012700	056550		T12:		MOV	@TPAT10,R0	;SET UP THE INPUT DATA BUFFER.
15573	056416	012701	056600				MOV	@TDATIO,R1	
15574	056422	012702	000004				MOV	@4,R2	
15575	056426	012021			T13:		MOV	(R0)*,(R1).	
15576	056430	077202					SOB	R2.T13	
15577									
15578	056432	012700	056600				MOV	@TDATIO,R0	;SET UP ACO
15579	056436	172410					LDD	(R0),ACO	
15580									
15581	056440	012700	056560				MOV	@TPAT20,R0	;SET UP AC1
15582	056444	172510					LDD	(R0),AC1	
15583									
15584	056446	012701	000001				MOV	@1,R1	
15585	056452	170001					SETF		;CLEAR FD
15586	056454	174001			T14:		STF	ACO,AC1	
15587	056456	000240			T15:		NOP		
15588	056460	000240			T16:		NOP		
15589									
15590	056462	005000					CLR	R0	
15591	056464	170200					STFPS	R0	;SEE IF FPS IS CLEAR.
15592	056466	022700	000010				CHP	@10,R0	
15593	056472	001401					BEG	T17	
(2)	056474	104000					ENT		
15594	056476				T17:				
15595	056476	170011					SETD		;SET FD.
15596									
15597	056500	012700	056570				MOV	@TDAT00,R0	
15598	056504	174110					STD	AC1,(R0)	;PICK UP AC1.
15599									
15600	056506	012737	177777	056604			MOV	@-1,@TDAT12	
15601	056514	012737	177777	056606			MOV	@-1,@TDAT13	
15602	056522	012703	056570				MOV	@TDAT00,R3	

15603 056526 012704 056600
 15604 056532 012705 000004
 15605 056536 022324
 15606 056540 001401
 (2) 056542 104000
 15607 056544 077504
 15608 056546 000420
 15609
 15610
 15611 056550 000000
 15612 056552 170360
 15613 056554 016161
 15614 056556 052525
 15615
 15616 056560 177777
 15617 056562 177777
 15618 056564 177777
 15619 056566 177777
 15620
 15621 056570 000000
 15622 056572 000000
 15623 056574 000000
 15624 056576 000000
 15625
 15626 056600 000000
 15627 056602 000000
 15628 056604 000000
 15629 056606 000000
 15630
 15631 056610
 (1) 056610 004767 045736
 (1)
 (1)
 (1)
 (1)
 (1)
 15632
 15633
 15644
 15645
 (2)
 (3)
 (2) 056614
 15646 056614 170011
 15647
 (1) 056616 012700 060350
 (1) 056622 012701 060410
 (1) 056626 004737 060222
 (1) 056632 012703 000102
 (1) 056636
 (1) 056636 172410
 (1) 056640 174000
 (1) 056642 172400
 (1) 056644 174011
 (1) 056646 004737 060320
 (1)
 (1) 056652 005737 060344

MOV @TDATIO,R4
 MOV #4,R5
 T20: CMP (R3), (R4) ; WAS THE DATA TRANSFERRED CORRECTLY?
 BEQ T23
 EMT ;
 T23: SOB R5, T20
 BR TDONE

TPAT10: 0
 TPAT11: 170360
 TPAT12: 016161
 TPAT13: 052525

TPAT20: -1
 TPAT21: 1
 TPAT22: -1
 TPAT23: -1

TDAT00: 0
 TDAT01: 0
 TDAT02: 0
 TDAT03: 0

TDATIO: 0
 TDATI1: 0
 TDATI2: 0
 TDATI3: 0

TDONE: JSR PC, .RSET ; GO INITIALIZE THE FPS AND STACK; AND
 ; SEE IF THE USER HAS EXPRESSED
 ; THE DESIRE TO CHANGE THE SOFTWARE
 ; VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ; THE USER TYPED CONTROL G?).

 ; TEST 442 ACCUMULATORS DATA PATTERNS TEST
 ;*****

TS442: SETD ; SET FD.
 ; TEST ACCUMULATOR 0 WITH FLOATING ONE
 MOV @GPAT00, R0
 MOV @GDAT00, R1
 JSR PC, @GSETUP ; LOAD TEST PATTERN.
 MOV @I02, R3
 G1: LDD (R0), ACO
 STD ACO, ACO
 LDD ACO, ACO ; STORE THE TEST PATTERN.
 STD ACO, (R1)
 JSR PC, @GCMP ; COMPARE THE DATA READ WITH
 ; THAT WHICH WAS WRITTEN.
 TST @GFLAG1

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58 P11 07-JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 23 2
T442 ACCUMULATORS DATA PATTERNS TEST

SEQ 0251

```

(1) 056656 001004      BNE      G2
(1) 056660 005137 060344  COM      @GFLAG1
(1) 056664 000261      SEC
(1) 056666 000401      BR       G3
(1) 056670 000241      G2:     CLC
(1) 056672 006160 000006  G3:     ROL      6(R0)      ;GENER E THE NEXT TEST PATTERN.
(1) 056676 006160 000004      ROL      4(R0)
(1) 056702 006160 000002      ROL      2(R0)
(1) 056706 006110      ROL      (R0)
(1) 056710 004737 060300  JSR      PC,@GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                     ;BUFFER.
(1) 056714 077330      SOB      R3,G1

;TEST ACCUMULATOR 0 WITH FLOATING ZERO
(1) 056716 012700 060360  MOV      @GPAT00,R0
(1) 056722 012701 060410  MOV      @GDAT00,R1
(1) 056726 004737 060222  JSR      PC,@GSETUP      ;LOAD TEST PATTERN.
(1) 056732 012703 000102  MOV      #102,R3
(1) 056736      G4:     LDD      (R0),ACO
(1) 056736 172410      STD      ACO,ACO
(1) 056740 174000      LDD      ACO,ACO      ;STORE THE TEST PATTERN.
(1) 056742 172400      STD      ACO,(R1)
(1) 056744 174011      JSR      PC,@GCMP      ;COMPARE THE DATA READ WITH
(1) 056746 004737 060320      ;THAT WHICH WAS WRITTEN.
(1) 056752 005737 060344  TST      @GFLAG1
(1) 056756 001004      BNE      G5
(1) 056760 005137 060344  COM      @GFLAG1
(1) 056764 000241      CLC
(1) 056766 000401      BR       G6
(1) 056770 000261      G5:     SEC
(1) 056772 006160 000006  G6:     ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 056776 006160 000004      ROL      4(R0)
(1) 057002 006160 000002      ROL      2(R0)
(1) 057006 006110      ROL      (R0)
(1) 057010 004737 060300  JSR      PC,@GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                     ;BUFFER.
(1) 057014 077330      SOB      R3,G4

;TEST ACCUMULATOR 1 WITH FLOATING ONE
(1) 057016 012700 060350  MOV      @GPAT00,R0
(1) 057022 012701 060410  MOV      @GDAT00,R1
(1) 057026 004737 060222  JSR      PC,@GSETUP      ;LOAD TEST PATTERN.
(1) 057032 012703 000102  MOV      #102,R3
(1) 057036      G7:     LDD      (R0),ACO
(1) 057036 172410      STD      ACO,AC1
(1) 057040 174001      LDD      AC1,ACO      ;STORE THE TEST PATTERN.
(1) 057042 172401      STD      ACO,(R1)
(1) 057044 174011      JSR      PC,@GCMP      ;COMPARE THE DATA READ WITH
(1) 057046 004737 060320      ;THAT WHICH WAS WRITTEN.
(1) 057052 005737 060344  TST      @GFLAG1
(1) 057056 001004      BNE      G10
(1) 057060 005137 060344  COM      @GFLAG1
(1) 057064 000261      SEC
(1) 057066 000401      BR       G11

```

```

(1) 057070 000241          G10:  CLC
(1) 057072 006160 000006  G11:  ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057076 006160 000004      ROL      4(R0)
(1) 057102 006160 000002      ROL      2(R0)
(1) 057106 006110          ROL      (R0)
(1) 057110 004737 060300      JSR      PC, @GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                     ;BUFFER.
(1) 057114 077330          SOB      R3,G7
(1)
(1) 15650          ;TEST ACCUMULATOR 1 WITH FLOATING ZERO
(1) 057116 012700 060360      MOV      @GPAT10,R0
(1) 057122 012701 060410      MOV      @GDAT00,R1
(1) 057126 004737 060222      JSR      PC, @GSETUP      ;LOAD TEST PATTERN.
(1) 057132 012703 000102      MOV      #102,R3
(1) 057136          G12:  LDD      (R0),AC0
(1) 057136 172410          STD      AC0,AC1
(1) 057140 174001          LDD      AC1,AC0          ;STORE THE TEST PATTERN.
(1) 057142 172401          STD      AC0,(R1)
(1) 057144 174011          JSR      PC, @GCMP
(1) 057146 004737 060320          ;COMPARE THE DATA READ WITH
(1)                                     ;THAT WHICH WAS WRITTEN.
(1) 057152 005737 060344      TST      @GFLAG1
(1) 057156 001004          BNE      G13
(1) 057160 005137 060344      COM      @GFLAG1
(1) 057164 000241          CLC
(1) 057166 000401          BR       G14
(1) 057170 000261          G13:  SEC
(1) 057172 006160 000006  G14:  ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057176 006160 000004      ROL      4(R0)
(1) 057202 006160 000002      ROL      2(R0)
(1) 057206 006110          ROL      (R0)
(1) 057210 004737 060300      JSR      PC, @GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                     ;BUFFER.
(1) 057214 077330          SOB      R3,G12
(1)
(1) 15651          ;TEST ACCUMULATOR 2 WITH FLOATING ONE
(1) 057216 012700 060350      MOV      @GPAT00,R0
(1) 057222 012701 060410      MOV      @GDAT00,R1
(1) 057226 004737 060222      JSR      PC, @GSETUP      ;LOAD TEST PATTERN.
(1) 057232 012703 000102      MOV      #102,R3
(1) 057236          G15:  LDD      (R0),AC0
(1) 057236 172410          STD      AC0,AC2
(1) 057240 174002          LDD      AC2,AC0          ;STORE THE TEST PATTERN.
(1) 057242 172402          STD      AC0,(R1)
(1) 057244 174011          JSR      PC, @GCMP
(1) 057246 004737 060320          ;COMPARE THE DATA READ WITH
(1)                                     ;THAT WHICH WAS WRITTEN.
(1) 057252 005737 060344      TST      @GFLAG1
(1) 057256 001004          BNE      G16
(1) 057260 005137 060344      COM      @GFLAG1
(1) 057264 000261          SEC
(1) 057266 000401          BR       G17
(1) 057270 000241          G16:  CLC
(1) 057272 006160 000006  G17:  ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057276 006160 000004      ROL      4(R0)
(1) 057302 006160 000002      ROL      2(R0)

```

```

(1) 057306 006110          ROL      (R0)
(1) 057310 004737 060300  JSR      PC, @GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                     ;BUFFER.
(1) 057314 077330          SOB      R3, G15
(1)                                     ;TEST ACCUMULATOR 2 WITH FLOATING ZERO
15652                                     MOV      @GPAT10, R0
(1) 057316 012700 060360  MOV      @GDAT00, R1
(1) 057322 012701 060410  JSR      PC, @GSETUP      ;LOAD TEST PATTERN.
(1) 057326 004737 060222  MOV      #102, R3
(1) 057332 012703 000102  G20:    LDD      (R0), AC0
(1) 057336 172410          STD      AC0, AC2
(1) 057340 174002          LDD      AC2, AC0          ;STORE THE TEST PATTERN.
(1) 057342 172402          STD      AC0, (R1)
(1) 057344 174011          JSR      PC, @GCMP      ;COMPARE THE DATA READ WITH
(1) 057346 004737 060320  TST      @GFLAG1          ;THAT WHICH WAS WRITTEN.
(1) 057352 005737 060344  BNE      G21
(1) 057356 001004          COM      @GFLAG1
(1) 057360 005137 060344  CLC
(1) 057364 000241          BR       G22
(1) 057366 000401          G21:    SEC
(1) 057370 000261          G22:    ROL      6(R0)          ;GENERATE THE NEXT TEST PATTERN.
(1) 057372 006160 000006  ROL      4(R0)
(1) 057376 006160 000004  ROL      2(R0)
(1) 057402 006160 000002  ROL      (R0)
(1) 057406 006110          JSR      PC, @GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1) 057410 004737 060300  SOB      R3, G20          ;BUFFER.
(1) 057414 077330
(1)                                     ;TEST ACCUMULATOR 3 WITH FLOATING ONE
15653                                     MOV      @GPAT00, R0
(1) 057416 012700 060350  MOV      @GDAT00, R1
(1) 057422 012701 060410  JSR      PC, @GSETUP      ;LOAD TEST PATTERN.
(1) 057426 004737 060222  MOV      #102, R3
(1) 057432 012703 000102  G23:    LDD      (R0), AC0
(1) 057436 172410          STD      AC0, AC3
(1) 057440 174003          LDD      AC3, AC0          ;STORE THE TEST PATTERN.
(1) 057442 172403          STD      AC0, (R1)
(1) 057444 174011          JSR      PC, @GCMP      ;COMPARE THE DATA READ WITH
(1) 057446 004737 060320  TST      @GFLAG1          ;THAT WHICH WAS WRITTEN.
(1) 057452 005737 060344  BNE      G24
(1) 057456 001004          COM      @GFLAG1
(1) 057460 005137 060344  CLC
(1) 057464 000261          BR       G25
(1) 057466 000401          G24:    CLC
(1) 057470 000241          G25:    ROL      6(R0)          ;GENERATE THE NEXT TEST PATTERN.
(1) 057472 006160 000006  ROL      4(R0)
(1) 057476 006160 000004  ROL      2(R0)
(1) 057502 006160 000002  ROL      (R0)
(1) 057506 006110          JSR      PC, @GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1) 057510 004737 060300  SOB      R3, G23          ;BUFFER.
(1) 057514 077330

```



```

(1)
15654      ;TEST ACCUMULATOR 3 WITH FLOATING ZERO
(1) 057516 012700 060360      MOV      #GPAT10,R0
(1) 057522 012701 060410      MOV      #GDAT00,R1
(1) 057526 004737 060222      JSR      PC,#GSETUP      ;LOAD TEST PATTERN.
(1) 057532 012703 000102      MOV      #102,R3
(1) 057536      G26:
(1) 057536 172410      LDD      (R0),AC0
(1) 057540 174003      STD      AC0,AC3
(1) 057542 172403      LDD      AC3,AC0      ;STORE THE TEST PATTERN.
(1) 057544 174011      STD      AC0,(R1)
(1) 057546 004737 060320      JSR      PC,#GCMP      ;COMPARE THE DATA READ WITH
(1)                                ;THAT WHICH WAS WRITTEN.
(1) 057552 005737 060344      TST      @GFLAG1
(1) 057556 001004      BNE      G27
(1) 057560 005137 060344      COM      @GFLAG1
(1) 057564 000241      CLC
(1) 057566 000401      BR       G30
(1) 057570 000261      G27: SEC
(1) 057572 006160 000006      G30: ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057576 006160 000004      ROL      4(R0)
(1) 057602 006160 000002      ROL      2(R0)
(1) 057606 006110      ROL      (R0)
(1) 057610 004737 060300      JSR      PC,#GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                ;BUFFER.
(1) 057614 077330      SOB      R3,G26
(1)
(1)
15655      ;TEST ACCUMULATOR 4 WITH FLOATING ONE
(1) 057616 012700 060350      MOV      #GPAT00,R0
(1) 057622 012701 060410      MOV      #GDAT00,R1
(1) 057626 004737 060222      JSR      PC,#GSETUP      ;LOAD TEST PATTERN.
(1) 057632 012703 000102      MOV      #102,R3
(1) 057636      G31:
(1) 057636 172410      LDD      (R0),AC0
(1) 057640 174004      STD      AC0,AC4
(1) 057642 172404      LDD      AC4,AC0      ;STORE THE TEST PATTERN.
(1) 057644 174011      STD      AC0,(R1)
(1) 057646 004737 060320      JSR      PC,#GCMP      ;COMPARE THE DATA READ WITH
(1)                                ;THAT WHICH WAS WRITTEN.
(1) 057652 005737 060344      TST      @GFLAG1
(1) 057656 001004      BNE      G32
(1) 057660 005137 060344      COM      @GFLAG1
(1) 057664 000261      SEC
(1) 057666 000401      BR       G33
(1) 057670 000241      G32: CLC
(1) 057672 006160 000006      G33: ROL      6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057676 006160 000004      ROL      4(R0)
(1) 057702 006160 000002      ROL      2(R0)
(1) 057706 006110      ROL      (R0)
(1) 057710 004737 060300      JSR      PC,#GRESET      ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                ;BUFFER.
(1) 057714 077330      SOB      R3,G31
(1)
(1)
15656      ;TEST ACCUMULATOR 4 WITH FLOATING ZERO
(1) 057716 012700 060360      MOV      #GPAT10,R0
(1) 057722 012701 060410      MOV      #GDAT00,R1

```

J4

```
(1) 057726 004737 060222      JSR    PC,@GSETUP      ;LOAD TEST PATTERN.
(1) 057732 012703 000102      MOV    @102,R3
(1) 057736                    G34:   LDD    (R0),AC0
(1) 057736 172410                STD    AC0,AC4
(1) 057740 174004                LDD    AC4,AC0      ;STORE THE TEST PATTERN.
(1) 057742 172404                STD    AC0,(R1)
(1) 057744 174011                JSR    PC,@GCMP      ;COMPARE THE DATA READ WITH
(1) 057746 004737 060320                ;THAT WHICH WAS WRITTEN.
(1) 057752 005737 060344                TST    @GFLAG1
(1) 057756 001004                BNE    G35
(1) 057760 005137 060344                COM    @GFLAG1
(1) 057764 000241                CLC
(1) 057766 000401                BR    G36
(1) 057770 000261                G35:   SEC
(1) 057772 006160 000006                G36:   ROL    6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 057776 006160 000004                ROL    4(R0)
(1) 060002 006160 000002                ROL    2(R0)
(1) 060006 006110                ROL    (R0)
(1) 060010 004737 060300                JSR    PC,@GRESET    ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                ;BUFFER.
(1) 060014 077330                SOB    R3,G34
(1)                                ;TEST ACCUMULATOR 5 WITH FLOATING ONE
15657 (1) 060016 012700 060350                MOV    @GPAT00,R0
(1) 060022 012701 060410                MOV    @GDAT00,R1
(1) 060026 004737 060222                JSR    PC,@GSETUP    ;LOAD TEST PATTERN.
(1) 060032 012703 000102                MOV    @102,R3
(1) 060036                    G37:   LDD    (R0),AC0
(1) 060036 172410                STD    AC0,AC5
(1) 060040 174005                LDD    AC5,AC0      ;STORE THE TEST PATTERN.
(1) 060042 172405                STD    AC0,(R1)
(1) 060044 174011                JSR    PC,@GCMP      ;COMPARE THE DATA READ WITH
(1) 060046 004737 060320                ;THAT WHICH WAS WRITTEN.
(1) 060052 005737 060344                TST    @GFLAG1
(1) 060056 001004                BNE    G40
(1) 060060 005137 060344                COM    @GFLAG1
(1) 060064 000261                SEC
(1) 060066 000401                BR    G41
(1) 060070 000241                G40:   CLC
(1) 060072 006160 000006                G41:   ROL    6(R0)      ;GENERATE THE NEXT TEST PATTERN.
(1) 060076 006160 000004                ROL    4(R0)
(1) 060102 006160 000002                ROL    2(R0)
(1) 060106 006110                ROL    (R0)
(1) 060110 004737 060300                JSR    PC,@GRESET    ;RESET DEFAULT PATTERN IN OUTPUT
(1)                                ;BUFFER.
(1) 060114 077330                SOB    R3,G37
(1)                                ;TEST ACCUMULATOR 5 WITH FLOATING ZERO
15658 (1) 060116 012700 060360                MOV    @GPAT10,R0
(1) 060122 012701 060410                MOV    @GDAT00,R1
(1) 060126 004737 060222                JSR    PC,@GSETUP    ;LOAD TEST PATTERN.
(1) 060132 012703 000102                MOV    @102,R3
(1) 060136                    G42:   LDD    (R0),AC0
```

```

(1) 060140 174005 STD AC0,AC5
(1) 060142 172405 LDD AC5,AC0 ;STORE THE TEST PATTERN.
(1) 060144 174011 STD AC0,(R1)
(1) 060146 004737 060320 JSR PC,@GCMF ;COMPARE THE DATA READ WITH
;THAT WHICH WAS WRITTEN.
(1) 060152 005737 060344 TST @GFLAG1
(1) 060156 001004 BNE G43
(1) 060160 005137 060344 COM @GFLAG1
(1) 060164 000241 CLC
(1) 060166 000401 BR G44
(1) 060170 000261 G43: SEC
(1) 060172 006160 000006 G44: ROL 6(R0) ;GENERATE THE NEXT TEST PATTERN.
(1) 060176 006160 000004 ROL 4(R0)
(1) 060202 006160 000002 ROL 2(R0)
(1) 060206 006110 ROL (R0)
(1) 060210 004737 060300 JSR PC,@GRESET ;RESET DEFAULT PATTERN IN OUTPUT
;BUFFER.
(1) 060214 077330 SOB R3,G42
(1) 15659 JMP @GDONE
15660 060216 000137 060420
15661
15662 ;USE THIS ROUTINE TO INITIALIZE ALL THE DATA BUFFERS.
15663 060222 012705 060344 GSETUP: MOV @GFLAG1,R5
15664 060226 012704 000026 MOV @26,R4
15665 060232 005025 1$: CLR (R5)+
15666 060234 077402 SOB R4,1$
15667
15668 060236 012705 060360 MOV @GPAT10,R5
15669 060242 012704 000010 MOV @10,R4
15670 060246 005125 2$: COM (R5)+
15671 060250 077402 SOB R4,2$
15672
15673 060252 020067 000072 GS1: CMP R0,GPAT00
15674 060256 001401 BEQ 3$
15675 060260 000207 RTS PC
15676
15677 060262 012705 060410 3$: MOV @GDAT00,R5
15678 060266 012704 000004 MOV @4,R4
15679 060272 005125 4$: COM (R5)+
15680 060274 077402 SOB R4,4$
15681 060276 000207 RTS PC
15682
15683 060300 012705 060410 GRESET: MOV @GDAT00,R5
15684 060304 012704 000004 MOV @4,R4
15685 060310 005025 1$: CLR (R5)+
15686 060312 077402 SOB R4,1$
15687 060314 000137 060252 JMP @GS1
15688
15689 ;SEE IF THE DATA WRITTEN MATCHES THE DATA READ
15690 060320 012705 060410 GCMP: MOV @GDAT00,R5
15691 060324 012704 000004 MOV @4,R4
15692 060330 010002 MOV R0,R2
15693 060332 022225 1$: CMP (R2)+,(R5)+
15694 060334 001401 BEQ 2$
(2) 060336 104000 EMT ;

```

15695 060340 077404
 15696 060342 000207
 15697
 15698
 15699
 15700
 15701 060344 000000
 15702 060346 000000
 15703
 15704 060350 000000
 15705 060352 000000
 15706 060354 000000
 15707 060356 000000
 15708
 15709 060360 177777
 15710 060362 177777
 15711 060364 177777
 15712 060366 177777
 15713
 15714 060370 177777
 15715 060372 177777
 15716 060374 177777
 15717 060376 177777
 15718
 15719 060400 000000
 15720 060402 000000
 15721 060404 000000
 15722 060406 000000
 15723
 15724 060410 000000
 15725 060412 000000
 15726 060414 000000
 15727 060416 000000
 15728
 15729
 15730 060420
 (1) 060420 004767 044126
 (1)
 (1)
 (1)
 (1)
 (1)
 15731
 15732
 15739
 (2)
 (3)
 (2) 060424
 15740 060424 005037 061114
 15741 060430 012700 061116
 15742 060434 012701 061236
 15743 060440 012703 000024
 15744 060444 012120
 15745 060446 077302
 15746
 15747 060450 004767 000420
 15748

24: SOB R4,16
 RTS PC

GFLAG1: 0
 GFLAG2: 0
 GPAT00: 0
 GPAT01: 0
 GPAT02: 0
 GPAT03: 0
 GPAT10: -1
 GPAT11: -1
 GPAT12: -1
 GPAT13: -1
 GAND0: -1
 GAND1: -1
 GAND2: -1
 GAND3: -1
 GOR0: 0
 GOR1: 0
 GOR2: 0
 GOR3: 0
 GDAT00: 0
 GDAT01: 0
 GDAT02: 0
 GDAT03: 0

GDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 443 FPP ACCUMULATORS DUAL ADDRESS TEST
 ;*****

TS443:
 H1: CLR B0HFLAG ;INITIALIZE THE LOAD BUFFER DATA.
 MOV #HA1W,R0
 MOV #HDAT1,R1
 MOV #24,R3
 H2: MOV (R1)+,(R0)+
 SOB R3,H2
 JSR PC,HCLR ;CLEAR THE OUTPUT DATA BUFFER.

15749 060454 170011
 15750
 (1) 060456 012700 061116
 (1) 060462 172410
 (1) 060464 174001
 15751
 (1) 060466 012700 061126
 (1) 060472 172410
 (1) 060474 174002
 15752
 (1) 060476 012700 061136
 (1) 060502 172410
 (1) 060504 174003
 15753
 (1) 060506 012700 061146
 (1) 060512 172410
 (1) 060514 174004
 15754
 (1) 060516 012700 061156
 (1) 060522 172410
 (1) 060524 174005
 15755
 15756 060526 004737 060762
 15757
 15758 060532 004737 061040
 15759
 15760
 (1)
 (1)
 (1) 060536 012700 061116
 (1) 060542 012702 000004
 (1) 060546 010001
 (1) 060550 005121
 (1) 060552 172410
 (1) 060554 174001
 (1) 060556 004737 060762
 (1) 060562 004737 061040
 (1) 060566 077210
 (1)
 15761
 (1)
 (1)
 (1) 060570 012700 061126
 (1) 060574 012702 000004
 (1) 060600 010001
 (1) 060602 005121
 (1) 060604 172410
 (1) 060606 174002
 (1) 060610 004737 060762
 (1) 060614 004737 061040
 (1) 060620 077210
 (1)
 15762
 (1)
 (1)
 (1) 060622 012700 061136

```

H3:  SETD
      ;LOAD ACCUMULATOR 1
      MOV  #HA1W,RO
      LDD  (RO),ACO
      STD  ACO,AC1
      ;LOAD ACCUMULATOR 2
      MOV  #HA2W,RO
      LDD  (RO),ACO
      STD  ACO,AC2
      ;LOAD ACCUMULATOR 3
      MOV  #HA3W,RO
      LDD  (RO),ACO
      STD  ACO,AC3
      ;LOAD ACCUMULATOR 4
      MOV  #HA4W,RO
      LDD  (RO),ACO
      STD  ACO,AC4
      ;LOAD ACCUMULATOR 5
      MOV  #HA5W,RO
      LDD  (RO),ACO
      STD  ACO,AC5

H4:  JSR  PC,@HSTD          ;GO READ ALL ACCUMULATORS BACK.
      JSR  PC,@HCMP        ;SEE IF DATA IS CORRECT.

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 1,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA1W,RO
      MOV  #4,R2
      MOV  RO,R1
H5:  COM  (R1)+
      LDD  (RO),ACO
      STD  ACO,AC1
      JSR  PC,@HSTD          ;READ ALL THE ACCUMULATORS BACK.
      JSR  PC,@HCMP        ;CHECK THE DATA.
      SOB  R2,H5

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 2,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA2W,RO
      MOV  #4,R2
      MOV  RO,R1
H6:  COM  (R1)+
      LDD  (RO),AC
      STD  ACO,AC2
      JSR  PC,@HSTD          ;READ ALL THE ACCUMULATORS BACK.
      JSR  PC,@HCMP        ;CHECK THE DATA.
      SOB  R2,H6

      ;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 3,
      ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
      ;THE DATA.
      MOV  #HA3W,RO
  
```

(1) 060626 012702 000004
 (1) 060632 010001
 (1) 060634 005121
 (1) 060636 172410
 (1) 060640 174003
 (1) 060642 004737 060762
 (1) 060646 004737 061040
 (1) 060652 077210

MOV #4,R2
 MOV R0,R1
 H7: COM (R1)+
 LDD (R0),AC0
 STD AC0,AC3
 JSR PC,@HSTD ;READ ALL THE ACCUMULATORS BACK.
 JSR PC,@HCMP ;CHECK THE DATA.
 SOB R2,H7

15763
 (1)
 (1)
 (1) 060654 012700 061146
 (1) 060660 012702 000004
 (1) 060664 010001
 (1) 060666 005121
 (1) 060670 172410
 (1) 060672 174004
 (1) 060674 004737 060762
 (1) 060700 004737 061040
 (1) 060704 077210

;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 4,
 ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
 ;THE DATA.
 MOV #HA4W,R0
 MOV #4,R2
 MOV R0,R1
 H10: COM (R1)+
 LDD (R0),AC0
 STD AC0,AC4
 JSR PC,@HSTD ;READ ALL THE ACCUMULATORS BACK.
 JSR PC,@HCMP ;CHECK THE DATA.
 SOB R2,H10

15764
 (1)
 (1)
 (1) 060706 012700 061156
 (1) 060712 012702 000004
 (1) 060716 010001
 (1) 060720 005121
 (1) 060722 172410
 (1) 060724 174005
 (1) 060726 004737 060762
 (1) 060732 004737 061040
 (1) 060736 077210

;COMPLIMENT EACH WORD OF THE DATA STORED IN ACCUMULATOR 5,
 ;RELOAD THAT ACCUMULATOR, READ ALL THE ACCUMULATORS BACK AND CHECK
 ;THE DATA.
 MOV #HA5W,R0
 MOV #4,R2
 MOV R0,R1
 H11: COM (R1)+
 LDD (R0),AC0
 STD AC0,AC5
 JSR PC,@HSTD ;READ ALL THE ACCUMULATORS BACK.
 JSR PC,@HCMP ;CHECK THE DATA.
 SOB R2,H11

15765
 15766 060740 005737 061114
 15767 060744 001402
 15768 060746 000137 061306
 15769
 15770 060752 005137 061114
 15771 060756 000137 060454

TST @HFLAG
 BEQ H12
 JMP @HDONE
 H12: COM @HFLAG
 JMP @H3

15772
 15773
 15774 060762 004737 061074
 15775
 (1) 060766 012704 061166
 (1) 060772 172401
 (1) 060774 174014
 15776
 (1) 060776 012704 061176
 (1) 061002 172402
 (1) 061004 174014
 15777
 (1) 061006 012704 061206
 (1) 061012 172403

;STORE ALL ACCUMULATORS IN THE OUTPUT BUFFERS.
 HSTD: JSR PC,@HCLR ;CLEAR ALL OUTPUT BUFFERS.
 ;STORE ACCUMULATOR 1
 MOV #HA1R,R4
 LDD AC1,AC0
 STD AC0,(R4)
 ;STORE ACCUMULATOR 2
 MOV #HA2R,R4
 LDD AC2,AC0
 STD AC0,(R4)
 ;STORE ACCUMULATOR 3
 MOV #HA3R,R4
 LDD AC3,AC0

CJKL560 LCP 5 CPU CLSTR DIAG
CJKLSB P11 07-JAN-85 09:05

MACY11 3C(1046) 07-JAN-85 09:28 PAGE 23 11
T443 FPP ACCUMULATORS DUAL ADDRESS TEST

SEQ 0260

(1)	061014	174014					STD	AC0.(R4)		
15778							;STORE	ACCUMULATOR 4		
(1)	061016	012704	061216				MOV	#HA4R,R4		
(1)	061022	172404					LDC	AC4,AC0		
(1)	061024	174014					STD	AC0.(R4)		
15779							;STORE	ACCUMULATOR 5		
(1)	061026	012704	061226				MOV	#HA5R,R4		
(1)	061032	172405					LDD	AC5,AC0		
(1)	061034	174014					STD	AC0.(R4)		
15780	061036	000207					RTS	PC		
15781										
15782							;COMPARE	DATA LOADED WITH DATA READ.		
15783	061040	012637	061112				HCMP:	MOV	(SP),#HADR	;SAVE RETURN ADDRESS
15784	061044	012703	061116					MOV	#HA1W,R3	
15785	061050	012704	061166					MOV	#HA1R,R4	
15786	061054	012705	000024					MOV	#24,R5	
15787	061060	022324					HCMP1:	CMF	(R3), (R4)	
15788	061062	001401						BEQ	HCMP2	
(2)	061064	104000						EMT		
15789	061066	077504					HCMP2:	SOB	R5,HCMP1	
15790	061070	000177	000016					JMP	HADR	
15791										
15792							;CLEAR	THE DATA OUTPUT BUFFER.		
15793	061074	012704	061166				HCLR:	MOV	#HA1R,R4	
15794	061100	012705	000024					MOV	#24,R5	
15795	061104	005024					HCLR1:	CLR	(R4)	
15796	061106	077502						SOB	R5,HCLR1	
15797	061110	000207						RTS	PC	
15798										
15799	061112	000000					HADR:	0		
15800	061114	000000					HFLAG:	0		
15801										
15802	061116	000000	000000	000000			HA1W:	.WORD	0,0,0,0	
	061124	000000								
15803	061126	000000	000000	000000			HA2W:	.WORD	0,0,0,0	
	061134	000000								
15804	061136	000000	000000	000000			HA3W:	.WORD	0,0,0,0	
	061144	000000								
15805	061146	000000	000000	000000			HA4W:	.WORD	0,0,0,0	
	061154	000000								
15806	061156	000000	000000	000000			HA5W:	.WORD	0,0,0,0	
	061164	000000								
15807										
15808	061166	000000	000000	000000			HA1R:	.WORD	0,0,0,0	
	061174	000000								
15809	061176	000000	000000	000000			HA2R:	.WORD	0,0,0,0	
	061204	000000								
15810	061206	000000	000000	000000			HA3R:	.WORD	0,0,0,0	
	061214	000000								
15811	061216	000000	000000	000000			HA4R:	.WORD	0,0,0,0	
	061224	000000								
15812	061226	000000	000000	000000			HA5R:	.WORD	0,0,0,0	
	061234	000000								
15813										
15814	061236	073567	073567	073567			HDAT1:	.WORD	73567,73567,73567,73567	
	061244	073567								

15815 061246 063146 063146 063146 HDAT2: .WORD 63146,63146,63146,63146
 061254 063146
 15816 061256 010421 010421 010421 HDAT3: .WORD 10421,10421,10421,10421
 061264 010421
 15817 061266 031463 031463 031463 HDAT4: .WORD 31463,31463,31463,31463
 061274 031463
 15818 061276 042104 042104 042104 HDAT5: .WORD 42104,42104,42104,42104
 061304 042104

15819
 15820 061306 HDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 (1) 061306 004767 043240 ;SEE IF THE USER HAS EXPRESSED
 (1) ;THE DESIRE TO CHANGE THE SOFTWARE
 (1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 (1) ;THE USER TYPED CONTROL G?).

15821
 15822
 15830 (2) ;*****
 (3) ;TEST 444 FSRC MODE 0 WITH ILLEGAL ACCUMULATOR TEST
 (2) ;*****
 TS444:

15831 061312 170011 SETD ;SET FD
 15832 061314 012700 061574 MOV #SPAT10,R0 ;LOAD ACO
 15833 061320 172410 LDD (R0),ACO
 15834
 15835 061322 012737 061500 000244 MOV #SERR0,#FPVECT ;USE OF THE NON-EXISTENT AC
 15836 ;CUMULATOR SHOULD RESULT IN
 15837 ;A TRAP TO 244.
 15838 061330 012700 000001 MOV #1,R0 ;A FAILURE IN THE FSRC FLOWS
 15839 ;WILL RESULT IN AN ODD ADDRESS
 15840 061334 012737 061406 000004 MOV #SERR1,#ERRVECT ;TRAP TO 4.
 15841 061342 005003 CLR R3
 15842
 15843 061344 172407 SX2: LDD AC7,ACO
 15844 061346 170000 SX3: CFCC
 15845 061350 005203 INC R3
 15846 061352 005203 SX4: INC R3
 15847
 15848 061354 012701 061604 MOV #SDAT00,R1 ;NO TRAP OCCURRED!!
 15849 061360 174011 STD ACO,(R1) ;SEE IF ACO WAS MODIFIED.
 15850

15851 061362 012701 061604 MOV #SDAT00,R1
 15852 061366 012702 061574 MOV #SPAT10,R2
 15853 061372 012703 000004 MOV #4,R3
 15854 061376 022122 SX5: CMP (R1),.(R2).
 15855 061400 001401 BEQ SX6
 (2) 061402 104000 EMT ;
 15856 061404 077304 SX6: SOB R3,SX5
 15857 061406 SERR1: EMT ;
 (2) 061406 104000

15858
 15859 ;NOW TEST AC6.
 15860 061410 170011 SX7: SETD
 15861 061412 012700 061574 MOV #SPAT10,R0 ;LOAD ACO
 15862 061416 172410 LDD (R0),ACO
 15863 061420 012737 061550 000244 MOV #SERR4,#FPVECT


```

15864 061426 012700 000001      MOV    #1,R0
15865 061432 005003      CLR    R3
15866
15867 061434 172406      SX8:   LDD    AC6,AC0
15868 061436 170000      SX9:   CFCC
15869 061447 005203      INC    R3
15870 061442 005203      SX10:  INC    R3
15871
15872 061444 012701 061604      MOV    #SDAT00,R1
15873 061450 174011      STD    AC0,(R1)      ;NO TRAP! GET ACO.
15874
15875 061452 012701 061604      MOV    #SDAT00,R1      ;WAS ACO MODIFIED.
15876 061456 012702 061574      MOV    #SPAT10,R2
15877 061462 012703 000004      MOV    #4,R3
15878 061466 022122      SX11:  CMP    (R1)+,(R2)+
15879 061470 001401      BEG    SX12
(2) 061472 104000      EMT
15880 061474 077304      SX12:  SOB    R3,SX11
15881 061476 104000      EMT
15882
15883      ;TRAPPED TO 244.
15884 061500 021627 061346      SERR0: CMP    (SP),#SX3      ;PC OF TRAP CORRECT?
15885 061504 001401      BEQ    14
(2) 061506 104000      EMT
15886 061510 012737 061410 061570 14:      MOV    #SX7,#SADR
15887 061516 022626      SERR10: CMP    (SP)+,(SP)+
15888 061520 005004      CLR    R4
15889 061522 170204      STFPS  R4      ;IS FPS CORRECT?
15890 061524 022704 100200      CMP    #100200,R4
15891 061530 001326      BNE    SERR1
15892
15893 061532 005004      CLR    R4
15894 061534 170304      STST   R4      ;IS FEC CORRECT?
15895 061536 022704 000002      CMP    #2,R4
15896 061542 001321      BNE    SERR1
15897 061544 000177 000020      JMP    #SADR
15898
15899 061550 021627 061436      SERR4: CMP    (SP),#SX9
15900 061554 001401      BEQ    14
(2) 061556 104000      EMT
15901 061560 012737 061614 061570 14:      MOV    #SDONE,#SADR
15902 061566 000753      BR     SERR10
15903
15904 061570 000000      SADR:   0
15905 061572 177777      -1
15906 061574 010421      SPAT10: 10421
15907 061576 021042      SPAT11: 21042
15908 061600 031463      SPAT12: 31463
15909 061602 042104      SPAT13: 42104
15910
15911 061604 000000      SDAT00: 0
15912 061606 000000      SDAT01: 0
15913 061610 000000      SDAT02: 0
15914 061612 000000      SDAT03: 0
15915
15916 061614      SDONE:

```

```

(1) 061614 004767 042732 JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
15917
15924

```

```

(2) ;*****
(3) ;TEST 445 FSRC MODE 2 TEST
(2) ;*****
15925 TS445:
15926 J1:
15927

```

```

15928 061620 170011 SETD ;SET DOUBLE MODE
15929 061622 012700 061750 MOV #JDAT0,R0
15930 061626 172410 LDD (R0),ACO ;LOAD ACO=ALL 1
15931
15932 061630 012700 061730 MOV #JDATIO,R0
15933 061634 065003 CLR R3

```

```

15934 061636 172420 J2: LDD (R0)+,ACO ;TEST INSTRUCTION
15935 061640 005203 J3: INC R3
15936 061642 005203 J4: INC R3
15937

```

```

15938 061644 012701 061740 MOV #JDAT00,R1
15939 061650 174011 STD ACO,(R1) ;PICK UP RESULTS
15940
15941 061652 020027 061720 CMP R0,#JBUFO ;WAS AN AUTO
15942 061656 001001 BNE 18

```

```

(2) 061660 104000 EMT ;
15943 061662 012702 061730 18: MOV #JDATIO,R2 ;IS DATA CORRECT?
15944 061666 012703 061740 MOV #JDAT00,R3
15945 061672 012704 000004 MOV #4,R4
15946 061676 022223 J5: CMP (R2)+,(R3)+
15947 061700 001401 BEQ J6
(2) 061702 104000 EMT ;
15948 061704 077404 J6: SOB R4,J5
15949

```

```

15950 061706 022700 061740 CMP #JDATIO+10,R0 ;WAS R0 INCREM.
15951 061712 001401 BEQ J7
(2) 061714 104000 EMT ;
15952 061716 000420 J7: BR JDONE
15953

```

```

15954 061720 010421 JBUF0: .WORD 010421
15955 061722 021042 JBUF1: 021042
15956 061724 042104 JBUF2: 042104
15957 061726 031463 JBUF3: 031463
15958
15959 061730 052525 JDATIO: 052525
15960 061732 114631 JDATIO1: 114631
15961 061734 063146 JDATIO2: 063146
15962 061736 073567 JDATIO3: 073567
15963

```

```

15964 061740 000000 JDAT00: 0
15965 061742 000000 JDAT01: 0
15966 061744 000000 JDAT02: 0
15967 061746 000000 JDAT03: 0

```

15968
 15969 061750 177777
 15970 061752 177777
 15971 061754 177777
 15972 061756 177777
 15973
 15974
 15975 061760
 (1) 061760 004767 042566
 (1)
 (1)
 (1)
 (1)
 15976
 15983
 15984
 (2)
 (3)
 (2) 061764
 15985 061764 170011
 15986
 15987 061766 012700 062114
 15988 061772 172410
 15989
 15990 061774 012700 062074
 15991 062000 005003
 15992 062002 172440
 15993 062004 005203
 15994 062006 005203
 15995
 15996 062010 012701 062104
 15997 062014 174011
 15998
 15999 062016 020027 062104
 16000 062022 001001
 (2) 062024 104000
 16001 062026 012702 062064
 16002 062032 012703 062104
 16003 062036 012704 000004
 16004 062042 022223
 16005 062044 001401
 (2) 062046 104000
 16006 062050 077404
 16007
 16008 062052 022700 062064
 16009 062056 001401
 (2) 062060 104000
 16010 062062 000420
 16011
 16012 062064 052525
 16013 062066 114631
 16014 062070 063140
 16015 062072 073567
 16016
 16017 062074 010421
 16018 062076 031463

JDAT0: 1
 JDAT1: -1
 JDAT2: 1
 JDAT3: -1

JDONE: JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 446 FSRC MODE 4 TEST
 ;*****
 TS446:

SETD ;SET DOUBLE MODE

MOV #KPAT0,R0
 LDD (R0),ACO ;LOAD A DEFAULT
 ;PATTERN INTO ACO

MOV #KBUF0,R0
 CLR R3
 LDD -(R0),ACO ;TEST INSTRUCTION

KX2: INC R3
 KX3: INC R3
 KX4: INC R3

MOV #KDAT00,R1
 STD ACO,(R1) ;PICK UP THE RESULT

CMP R0,#KBUF0+10 ;WAS AN AUTO
 BNE 14
 EMT ;
 14: MOV #KDAT10,R2 ;IS DATA CORRECT?
 MOV #KDAT00,R3
 MOV #4,R4
 KX5: CMP (R2)+,(R3)+
 BEQ KX6
 EMT ;
 KX6: SOB R4,KX5

CMP #KBUF0-10,R0 ;WAS R0 DECREMENTED
 BEQ KX7
 EMT ;
 KX7: BR KDONE

KDAT10: .WORD 052525
 KDAT11: 114631
 KDAT12: 063140
 KDAT13: 073567

KBUF0: 010421
 KBUF1: 031463

16019 062100 042104
 16020 062102 021042
 16021
 16022 062104 000000
 16023 062106 000000
 16024 062110 000000
 16025 062112 000000
 16026
 16027 062114 177777
 16028 062116 177777
 16029 062120 177777
 16030 062122 177777
 16031
 16032 062124
 (1) 062124 004767 042422
 (1)
 (1)
 (1)
 (1)
 16033
 16040
 16041
 (2)
 (3)
 (2) 062130
 16042 062130
 16043 062130 170011
 16044
 16045 062132 012700 062256
 16046 062136 172410
 16047
 16048 062140 012700 062300
 16049 062144 012701 062266
 16050 062150 012702 000004
 16051
 16052 062154 012120
 16053 062156 077202
 16054
 16055 062160 012700 062300
 16056 062164 005003
 16057 062166 170001
 16058
 16059 062170 172420
 16060 062172 005203
 16061
 16062 062174
 16063 062174 170011
 16064
 16065 062176 012701 062312
 16066 062202 174011
 16067
 16068 062204 020027 062304
 16069 062210 001401
 (2) 062212 104000
 16070 062214 012737 177777 062304
 16071 062222 012737 177777 062306

KBUF2: 042104
 KBUF3: 021042
 KDAT00: 0
 KDAT01: 0
 KDAT02: 0
 KDAT03: 0
 KPAT0: 1
 KPAT1: 1
 KPAT2: 1
 DPAT3: 1
 KDONE:
 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 447 FSRC MODE 2, WITH FD=0, TEST
 ;*****
 TS447:
 L1: SETD ;SET DOUBLE MODE
 MOV #LPAT10,R0
 LDD (R0),ACO ;LOAD ACO
 MOV #LDAT10,R0 ;SET UP THE INPUT
 MOV #LPAT20,R1 ;DATA
 MOV #4,R2
 14: MOV (R1)+,(R0)+
 SOB R2,14
 MOV #LDAT10,R0
 CLR R3
 SETF ;CLEAR FD.
 L2: LDF (R0)+,ACO
 L3: INC R3
 L4: SETD ;SET FD
 MOV #LDAT00,R1
 STD ACO,(R1) ;PICK UP RESULTS
 CMP R0,#LDAT12 ;WAS R0 INCREMENTED
 BEQ 14
 EMT
 14: MOV #-1,#LDAT12
 MOV #-1,#LDAT13

16072 062230 012702 062300
 16073 062234 012703 062312
 16074 062240 012704 000004
 16075
 16076 062244 022223
 16077 062246 001401
 (2) 062250 104000
 16078 062252 077404
 16079 062254 000422
 16080
 16081 062256 177777
 16082 062260 177777
 16083 062262 177777
 16084 062264 177777
 16085
 16086 062266 052525
 16087 062270 114631
 16088 062272 063142
 16089 062274 073567
 16090 062276 000001
 16091 062300 000000
 16092 062302 000000
 16093 062304 000000
 16094 062306 000000
 16095 062310 000001
 16096 062312 000000
 16097 062314 000000
 16098 062316 000000
 16099 062320 000000
 16100
 16101 062322
 (1) 062322 004767 042224
 (1)
 (1)
 (1)
 (1)
 16102
 16110
 16111
 (2)
 (3)
 (2) 062326
 16112
 16113 062326
 16114 062326 170011
 16115
 16116 062330 012700 062430
 16117 062334 172410
 16118
 16119 062336 005004
 16120 062340 012737 062366 000004
 16121
 16122 062346 172427 000000
 16123 062350
 16124 062350 005204
 16125 062352 005204

MOV #LDAT10,R2 ;IS DATA CORRECT
 MOV #LDAT00,R3
 MOV #4,R4
 L5: CMP (R2), (R3)
 BEG L6
 EMT ;
 L6: SOB R4, L5
 BR LDONE
 LPAT10: .WORD -1
 LPAT11: -1
 LPAT12: 1
 LPAT13: 1
 LPAT20: 052525
 LPAT21: 114631
 LPAT22: 063142
 LPAT23: 073567
 .WORD 000001
 LDAT10: 0
 LDAT11: 0
 LDAT12: 0
 LDAT13: 0
 .WORD 00001
 LDAT00: 0
 LDAT01: 0
 LDAT02: 0
 LDAT03: 0
 LDONE: JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 450 FSRC MODE 2 WITH GR7, IMMEDIATE MODE, TEST

 TS450:

M1: SETD
 MOV #MPAT10,R0
 LDD (R0),AC0 ;LOAD BACKGROUND
 ;PATTERN INTO AC0.
 CLF R4
 MOV #MERR3, #MERRVECT
 M15: LDD #0,AC0 ;TEST INSTRUCTION
 .=-2 ;EFFECTIVELY; 05204 IS PUT IN THE FIRST
 .WORD 5204 ;16 BIT WORD, OR THE "EXP-FRACTION" WORD.
 M2: INC R4 ;NOTE THAT

16126 062354 005204
 16127 062356 005204
 16128 062360 020427 000003
 16129 062364 001401
 (1) 062366
 (2) 062366 104000
 16130 062370 012700 062450
 16131 062374 174010
 16132
 16133 062376 012700 062450
 16134 062402 022720 005204
 16135 062406 001401
 (2) 062410 104000
 16136 062412 012701 000003
 16137 062416 005720
 16138 062420 001401
 (2) 062422 104000
 16139 062424 077104
 16140 062426 000414
 16141
 16142 062430 177777
 16143 062432 177777
 16144 062434 177777
 16145 062436 177777
 16146
 16147 062440 005204
 16148 062442 005204
 16149 062444 005204
 16150 062446 005204
 16151
 16152 062450 000000
 16153 062452 000000
 16154 062454 000000
 16155 062456 000000
 16156
 16157 062460
 (1) 062460 004767 042066
 (1)
 (1)
 (1)
 (1)
 16158
 16165
 16166
 (2)
 (3)
 (2) 062464
 16167
 16168 062464
 16169 062464 170011
 16170
 16171 062466 012700 062614
 16172 062472 172410
 16173
 16174 062474 012700 062602
 16175 062500 005003

M3: INC R4 ;005204=INC R4
 M4: INC R4
 CMP R4,#3 ;SEE IF THE PC
 BEQ M8
 MERR3:
 EMT ;
 M8: MOV #MDAT00,R0 ;
 STD ACO,(R0) ;GET THE DATA
 MOV #MDAT00,R0
 CMP #5204,(R0) ;IS THE DATA CORRECT?
 BEQ M5
 EMT ;
 M5: MOV #3,R1
 M6: TST (R0) ;
 BEQ M7
 EMT ;
 M7: SOB R1,M6
 BR MDONE
 MPAT10: -1
 MPAT11: -1
 MPAT12: -1
 MPAT13: -1
 MPAT20: 5204
 MPAT21: 5204
 MPAT22: 5204
 MPAT23: 5204
 MDAT00: 0
 MDAT01: 0
 MDAT02: 0
 MDAT03: 0
 MDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 451 FSRC MODE 3 TEST
 ;*****
 TS451:

N1: SETD ;SET FD MODE
 MOV #MPAT10,R0
 LDD (R0),ACO ;LOAD ACO WITH A DEFAULT
 ;PATTERN
 MOV #MPAT20,R0
 CLR R3

```

16176 062502 012737 062554 000004      MOV      #NERR0,#ERRVECT      ;IF A FAILURE OCCURS
16177                                     ;IN THE FSRC FLOWS AN
16178                                     ;ODD TRAP TO 4 COULD OCCUR
16179 062510 172430      N2:      LDD      @R0)+,AC0      ;TEST INSTRUCTION.
16180 062512 005203      N3:      INC      R3
16181 062514 005203      N4:      INC      R3
16182
16183 062516 012701 062562      MOV      #NDAT00,R1
16184 062522 174011      STD      AC0,(R1)      ;GET THE DATA
16185
16186 062524 020027 062604      CMP      R0,#NPAT20+2      ;WAS R0 INCREMENTED
16187 062530 001401      BEQ      N12
16188 (2) 062532 104000      EMT
16189 062534 012702 062562      N12:     MOV      #NDAT00,R2      ;DATA CORRECT
16190 062540 012703 062624      MOV      #NDATIO,R3
16191 062544 012704 000004      MOV      #4,R4
16192 062550 022223      N13:     CMP      (R2)+,(R3)+
16193 (1) 062554 001401      BEQ      N14
16194 (2) 062554 104000      NERR0:   EMT
16195 062556 077404      N14:     SOB      R4,N13
16196 062560 000425      BR
16197 NDAT00: .WORD 0
16198 NDAT01: .WORD 0
16199 NDAT02: .WORD 0
16200 NDAT03: .WORD 0
16201 062572 052525 052525 052525      .WORD 52525,52525,52525,52525
16202 062600 052525
16203 062602 062624      NPAT20: .WORD NDATIO
16204 062604 070707      NPAT21: .WORD 070707
16205 062606 070707      NPAT22: .WORD 070707
16206 062610 070707      NPAT23: .WORD 070707
16207 062612 000001      .WORD 1
16208 062614 177777      NPAT10: .WORD -1
16209 062616 177777      NPAT11: .WORD -1
16210 062620 177777      NPAT12: .WORD -1
16211 062622 177777      NPAT13: .WORD -1
16212 062624 010421      NDATIO: .WORD 010421
16213 062626 021042      NDATI1: .WORD 021042
16214 062630 031463      NDATI2: .WORD 031463
16215 062632 042104      NDATI3: .WORD 042104
16216
16217 062634      NDONE:
(1) 062634 004767 041712      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
(1)                                     ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
16218
16225
16226      ;*****
(2)      ;TEST 452      FSRC MODE 5 TEST
(3)      ;*****

```

```

(2) 062640
16227
16228 062640
16229 062640 170011
16230
16231 062642 012700 062770
16232 062646 172410
16233
16234 062650 012700 062756
16235 062654 005003
16236 062656 012737 062706 000004
16237
16238
16239
16240 062664 172450
16241 062666 005203
16242 062670 005203
16243
16244 062672 012701 062736
16245 062676 174011
16246
16247 062700 020027 062754
16248 062704 001401
(1) 062706
(2) 062706 104000
16249 062710 012702 062736
16250 062714 012703 063000
16251 062720 012704 000004
16252 062724 022223
16253 062726 001401
(2) 062730 104000
16254 062732 077404
16255 062734 000425
16256
16257 062736 000000
16258 062740 000000
16259 062742 000000
16260 062744 000000
16261
16262 062746 052525 052525 052525
16263 062754 063000
16264 062756 070707
16265 062760 070707
16266 062762 070707
16267 062764 070707
16268 062766 000001
16269 062770 177777
16270 062772 177777
16271 062774 177777
16272 062776 177777
16273
16274 063000 073567
16275 063002 004210
16276 063004 114631
16277 063006 125252
16278

```

```

TS452:
01:
    SETD                ;SET FD MODE
    MOV    #OPAT10,R0
    LDD    (R0),ACO      ;LOAD ACO WITH A
                        ;DEFAULT PATTERN.
    MOV    #OPAT21,R0
    CLR    R3
    MOV    #OERR0,#ERRVEC ;IF A FAILURE
                        ;OCCURS IN THE FSRC
                        ;FLOWS AN ODD ADDR.
                        ;TRAP TO 4 MAY OCCUR.
                        ;TEST INSTRUCTION
02:    LDD    @-(R0),ACO
03:    INC    R3
04:    INC    R3
    MOV    #ODAT00,R1
    STD    ACO,(R1)      ;GET THE DATA
    CMP    R0,#OPAT20    ;WAS R0 DECREMENTED
    BEQ    012
OERR0:
    EMT
012:   MOV    #ODAT00,R2 ;DATA CORRECT?
    MOV    #ODATIO,R3
    MOV    #4,R4
013:   CMP    (R2)+,(R3)+
    BEQ    014
    EMT
014:   SOB    R4,013
    BR     00ONE
ODAT00: .WORD 0
ODAT01: .WORD 0
ODAT02: .WORD 0
ODAT03: .WORD 0
OPAT20: .WORD 52525,52525,52525
OPAT21: .WORD 00ATIO
OPAT22: .WORD 070707
OPAT23: .WORD 070707
OPAT24: .WORD 070707
OPAT10: .WORD 1
OPAT11: .WORD -1
OPAT12: .WORD -1
OPAT13: .WORD -1
ODATIO: .WORD 73567
ODATIO1: .WORD 004210
ODATIO2: .WORD 114631
ODATIO3: .WORD 125252

```


16279 063010
 (1) 063010 004767 041536
 (1)
 (1)
 (1)
 (1)
 (1)
 16280
 16286
 16287
 (2)
 (3)
 (2) 063014
 16288
 16289 063014
 16290 063014 170011
 16291
 16292 063016 012700 063102
 16293 063022 172410
 16294
 16295 063024 012700 062651
 16296
 16297 063030 172460 000241
 16298 063032
 16299
 16300 063034 012701 063122
 16301 063040 174011
 16302 063042 012703 000004
 16303 063046 012702 063112
 16304 063052 012701 063122
 16305 063056 022221
 16306 063060 001401
 (2) 063062 104000
 16307 063064 077304
 16308 063066 022700 062651
 16309 063072 001401
 (2) 063074 104000
 16310 063076 000137 063132
 16311 063102 177777
 16312 063104 177777
 16313 063106 177777
 16314 063110 177777
 16315
 16316 063112 010421
 16317 063114 031463
 16318 063116 052525
 16319 063120 073567
 16320
 16321 063122 000000
 16322 063124 000000
 16323 063126 000000
 16324 063130 000000
 16325
 16326 063132
 (1) 063132 004767 041414
 (1)
 (1)

ODONE: JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

;*****
 ;TEST 453 FSRC MODE 6 TEST
 ;*****
 TS453:

P1: SETD ;SET FD MODE
 MOV #PPAT10,R0
 LDD (R0),ACO ;LOAD A DEFAULT PATTERN
 ;INTO ACO
 ;COULD OCCUR.
 MOV #PDATIO-241,R0

P2: LDD 241(R0),ACO
 P3=P2+2

P4: MOV #PDAT00,R1
 STD ACO,(R1) ;GET THE DATA
 MOV #4,R3
 MOV #PDATIO,R2
 MOV #PDAT00,R1

P5: CMP (R2)+,(R1)+ ;CHECK THE DATA
 BEQ 24
 EMT ;

24: SOB R3,P5 ;RO CORRECT?
 CMP #PDATIO-241,R0
 BEQ 14
 EMT ;

14: JMP B@PDONE
 PPAT10: .WORD -1
 PPAT11: -1
 PPAT12: -1
 PPAT13: -1

PDATIO: .WORD 010421
 PDATI1: 031463
 PDATI2: 052525
 PDATI3: 073567

PDAT00: .WORD 0
 PDAT01: 0
 PDAT02: 0
 PDAT03: 0

PDONE: JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE

;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

(1)
(1)
16327
16334
16335
(2)
(3)
(2) 063136
16336
16337 063136
16338 063136 170011
16339
16340 063140 012700 063224
16341 063144 172410
16342
16343 063146 012700 062773
16344
16345 063152 172470 000241
16346 063154
16347
16348 063156 012701 063244
16349 063162 174011
16350
16351 063164 012703 000004
16352 063170 012704 063244
16353 063174 012705 063254
16354 063200 022425
16355 063202 001401
(2) 063204 104000
16356 063206 077304
16357
16358 063210 022700 062773
16359 063214 001401
(2) 063216 104000
16360 063220 000137 063264
16361
16362 063224 177777
16363 063226 177777
16364 063230 177777
16365 063232 177777
16366
16367 063234 063254
16368 063236 052525
16369 063240 052525
16370 063242 052525
16371
16372 063244 000000
16373 063246 000000
16374 063250 000000
16375 063252 000000
16376
16377 063254 073567
16378 063256 052525
16379 063260 031463
16380 063262 010421
16381

.....
;TEST 454 FSRC MODE 7 TEST
;.....
TS454:

Q1: SETD

MOV #QPAT10,R0
LDD (R0),ACO ;LOAD A DEFAULT
;PATTERN INTO ACO

MOV #QPAT20-241,R0

Q2: LDD @241(R0),ACO
Q3=Q2*2

Q4: MOV #QDAT00,R1
STD ACO,(R1) ;GET THE DATA

MOV #4,R3
MOV #QDAT00,R4
MOV #QDAT10,R5
Q5: CMP (R4)*,(R5)* ;CHECK THE DATA
BEQ 2+
EMT ;
2+: SOB R3,Q5 ;

CMP #QPAT20-241,R0 ;CHECK R0.
BEQ 1+
EMT ;
1+: JMP @QDONE

QPAT10: .WORD -1
QPAT11: -1
QPAT12: -1
QPAT13: -1

QPAT20: .WORD QDAT10
QPAT21: 52525
QPAT22: 52525
QPAT23: 52525

QDAT00: .WORD 0
QDAT01: 0
QDAT02: 0
QDAT03: 0

QDAT10: .WORD 073567
QDAT11: .WORD 052525
QDAT12: .WORD 031463
QDAT13: .WORD 010421

```

16382 063264
(1) 063264 004767 041262
(1)
(1)
(1)
(1)
16383
16398
(2)
(3)
(2) 063270
16399 063270 005037 064042
16400 063274 012700 063772
16401 063300 012701 000004
16402 063304 012720 177777
16403 063310 077103
16404
16405 063312 012737 000033 064044
16406 063320 012737 000023 064046
16407 063326 012737 063412 030244
16408 063334 012700 000200
16409 063340 170100
16410 063342 012700 063772
16411 063346 172410
16412 063350 013737 064044 064050
16413 063356 012737 000001 064052
16414 063364 012737 000254 064054
16415
16416 063372 012700 064002
16417 063376 172410
16418 063400 012704 000204
16419 063404 170205
16420
16421 063406 020405
16422 063410 001401
(1) 063412
(2) 063412 104000
16423 063414 012700 000200
16424 063420 170100
16425
16426 063422 012700 063772
16427 063426 172410
16428 063430 013737 064046 064050
16429 063436 012737 000003 064052
16430 063444 012737 000054 064054
16431
16432 063452 012700 064012
16433
16434 063456 172410
16435 063460 012704 000200
16436 063464 170205
16437
16438 063466 020405
16439 063470 001401
(2) 063472 104000
16440 063474 012700 000200

```

```

QDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

```

```

;*****
;TEST 455 (BUT EZBT Y8),(BUT ENBT) AND (BUT FIUV) TEST
;*****
TS455:

```

```

CLR @UFLAG
MOV @UPAT00,R0 ;SET UP ACO0 DATA.
MOV #4,R1
U0: MOV #-1,(R0)+
SOB R1,U0
MOV #033,@UTMP1
MOV #023,@UTMP2
U1: MOV @UERR0,@FPVECT ;IN CASE (BUT FIUV FAILS)
MOV #200,R0
LDFPS R0
MOV @UPAT00,R0 ;LOAD ACO
LDD (R0),ACO
MOV @UTMP1,@UROM1
MOV #001,@UROM2
MOV #254,@UROM3
U2: MOV @UPAT10,R0 ;LOAD 0 INTO ACO
LDD (R0),ACO
MOV #204,R4 ;SEE IF FPS IS CORRECT
STFPS R5
CMP R4,R5
BEQ U3
UERR0:
U3: EMT
MOV #200,R0
LDFPS R0
MOV @UPAT00,R0 ;LOAD ACO
LDD (R0),ACO
MOV @UTMP2,@UROM1
MOV #003,@UROM2
MOV #054,@UROM3
U4: MOV @UPAT20,R0 ;LOAD A POSITIVE NUMBER
;INTO ACO
LDD (R0),ACO
MOV #200,R4 ;FPS CORRECT?
STFPS R5
CMP R4,R5
BEQ U5
(2) EMT
U5: MOV #200,R0

```

```

16441 063500 170100 LDFPS R0
16442 063502 012700 063772 MOV @UPAT00,R0 ;LOAD ACO
16443 063506 172410 LDD (R0),ACO
16444 063510 013737 064046 064050 MOV @OUTMP2,@UROM1
16445 063516 012737 000403 064052 MOV @403,@UROM2
16446 063524 012737 000056 064054 MOV @056,@UROM3
16447 063532 012700 064022 MOV @UPAT30,R0 ;LOAD A NEGATIVE
16448 ;NUMBER INTO ACO
16449 063536 172410 U6: LDD (R0),ACO
16450 063540 012704 000210 MOV @210,R4 ;FPS CORRECT
16451 063544 170205 STFPS R5
16452 063546 020405 CMP R4,R5
16453 063550 001401 BEQ U7
16454 (2) 063552 104000 EMT ;
16455 063554 012700 000200 U7: MOV @200,R0
16456 063560 170100 LDFPS R0
16457 063562 012700 063772 MOV @UPAT00,R0 ;LOAD ACO
16458 063570 013737 064044 064050 LDD (R0),ACO
16459 063576 012737 000401 064052 MOV @OUTMP1,@UROM1
16460 063604 012737 000256 064054 MOV @256,@UROM3
16461 063612 012700 064032 MOV @UPAT40,R0 ;LOAD -0 INTO ACO
16462 063616 172410 U10: LDD (R0),ACO
16463 063620 000240 U11: NOP ;TRAP FROM HERE IF
16464 063622 012704 000214 MOV @214,R4 ;SEE IF FPS IS CORRECT.
16465 063626 170205 STFPS R5
16466 063630 020405 CMP R4,R5
16467 063632 001401 BEQ U12
16468 (2) 063634 104000 EMT ;
16469 063636 005737 064042 U12: TST @UFLAG ;SEE IF ALL THE PATTERNS
16470 063642 001021 BNE U14 ;HAVE BEEN TEST WITH
;BOTH AC NOT EQUAL TO 0 AND AC=0
16471 063644 012700 063772 MOV @UPAT00,R0 ;IF NOT GO BACK AND
16472 063650 012701 000004 MOV @4,R1 ;CHECK THEM WITH AC=0
16473 063654 005020 U13: CLR (R0)
16474 063656 077102 SOB R1,U13
16475 063660 012737 177777 064042 MOV @-1,@UFLAG
16476 063666 012737 000233 064044 MOV @233,@OUTMP1
16477 063674 012737 000223 064046 MOV @223,@OUTMP2
16478 063702 000137 063326 JMP @U1
16479 ;NOW SEE IF A TRAP CAN BE FORCED BY SETTING FIUV AND LOADING -0
16480 063706 012737 063744 000244 U14: MOV @UERR3,@FPVECT
16481 063714 012700 004200 MOV @4200,R0 ;SET FD AND FIUV
16482 063720 170100 LDFPS R0
16483 063722 012700 063772 MOV @UPAT00,R0 ;SET UP ACO
16484 063726 172410 LDD (R0),ACO
16485 063730 012700 064032 MOV @UPAT40,R0 ;LOAD -0
16486 063734 172410 U15: LDD (R0),ACO ;SHOULD TRAP TO 244
16487 063736 170000 U16: CFCC
16488 063740 000240 NOP
16489 063742 104000 EMT ;
16490 ;
16491 ;INTERRUPT HERE WHEN FIUV SET AND ATTEMPTED TO LOAD 0
16492 063744 021627 063736 UERR3: CMP (SP),@U16
16493 063750 001401 BEQ 14
16494 (2) 063752 104000 EMT ;

```

16494 063754 022626
 16495 063756 005000
 16496 063760 170300
 16497 063762 022700 000014
 16498 063766 001433
 (2) 063770 104000
 16499 063772 000000
 16500 063774 000000
 16501 063776 000000
 16502 064000 000000
 16503
 16504 064002 000000
 16505 064004 000000
 16506 064006 000000
 16507 064010 000000
 16508
 16509 064012 010421
 16510 064014 114631
 16511 064016 125252
 16512 064020 177777
 16513
 16514 064022 114631
 16515 064024 135673
 16516 064026 146314
 16517 064030 167356
 16518
 16519 064032 100000
 16520 064034 000000
 16521 064036 000000
 16522 064040 000000
 16523
 16524 064042 000000
 16525 064044 000000
 16526 064046 000000
 16527 064050 000000
 16528 064052 000000
 16529 064054 000000
 16530 064056
 16531
 16532
 16540
 (2)
 (3)
 (2) 064056
 16541 064056 012700 000200
 16542 064062 170100
 16543 064064 012700 064406
 16544 064070 172410
 16545 064072 012700 064406
 16546 064076 172010
 16547 064100 170205

18: CMP (SP), (SP).
 CLR R0
 STST R0 ;GET FEC.
 CMP #14, R0 ;CORRECT
 BEQ UDONE
 EMT ;
 UPAT00: .WORD 0
 UPAT01: 0
 UPAT02: 0
 UPAT03: 0
 UPAT10: .WORD 0 ;0
 UPAT11: 0
 UPAT12: 0
 UPAT13: 0
 UPAT20: .WORD 010421 ;POS NUM
 UPAT21: 114631
 UPAT22: 125252
 UPAT23: 177777
 UPAT30: 114631 ;NEG NUM
 UPAT31: 135673
 UPAT32: 146314
 UPAT33: 167356
 UPAT40: 100000 ;NEG ZERO
 UPAT41: 0
 UPAT42: 0
 UPAT43: 0
 UFLAG: .WORD 0
 UTHP1: 0
 UTHP2: 0
 UROM1: 0
 UROM2: 0
 UROM3: 0
 UDONE:

 ;TEST 456 ADDF, ADD, SUBF AND SUBD WITH FSRC=AC=0 TEST

 TS456:
 MOV #200, R0
 LDFPS R0 ;SET DOUBLE MODE
 MOV @MPAT00, R0 ;LOAD AC0=
 LDD (R0), AC0
 MOV @MPAT00, R0
 W2: ADD (R0), AC0 ;TEST INSTRUCTION. ADD ITSELF
 STFPS R5 ;GET FPS

16549	064102	170011		SETD		;SET DOUBLE MODE
16550	064104	012700	064406	MOV	#MPAT00,R0	
16551	064110	174010		STD	ACO,(R0)	;GET THE RESULT
16552	064112	012701	064406	MOV	#MPAT00,R1	
16553	064116	012702	000004	MOV	#4,R2	
16554	064122	022021		W3: CMP	(R0)+,(R1)+	;IS RESULT CORRECT
16555	064124	001401		BEQ	W4	
(2)	064126	104000		EMT		
16556	064130	077204		W4: SOB	R2,W3	
16557	064132	022705	000204	CMP	#204,R5	;IS FPS CORRECT
16558	064136	001401		BEQ	W5	
(2)	064140	104000		EMT		
16559	064142	012700	000200	W5: MOV	#200,R0	
16560	064146	170100		LDFPS	R0	;SET DOUBLE MODE
16561	064150	012700	064406	MOV	#MPAT00,R0	;LOAD ACO=0
16562	064154	172410		LDD	(R0),ACO	
16563	064156	005000		CLR	R0	
16564	064160	170100		LDFPS	R0	;GO TO FLOATING MODE
16565	064162	012700	064406	MOV	#MPAT00,R0	
16566	064166	172010		W6: ADDF	(R0),ACO	;TEST INSTRUCTION
16567	064170	170205		STFPS	R5	;GET FPS
16568	064172	170011		SETD		;RESET TO DOUBLE MODE
16569	064174	012700	064406	MOV	#MPAT00,R0	
16570	064200	174010		STD	ACO,(R0)	;GET THE RESULT
16571	064202	012701	064406	MOV	#MPAT00,R1	
16572	064206	012702	000004	MOV	#4,R2	
16573	064212	022021		W7: CMP	(R0)+,(R1)+	;WAS THE RESULT
16574	064214	001401		BEQ	W10	
(2)	064216	104000		EMT		
16575	064220	077204		W10: SOB	R2,W7	
16576	064222	022705	000004	CMP	#4,R5	;WAS FPS CORRECT
16577	064226	001401		BEQ	W11	
(2)	064230	104000		EMT		
16578	064232	012700	000200	W11: MOV	#200,R0	
16579	064236	170100		LDFPS	R0	;SET DOUBLE MODE
16580	064240	012700	064406	MOV	#MPAT00,R0	;LOAD ACO=0
16581	064244	172410		LDD	(R0),ACO	
16582	064246	012700	064406	MOV	#MPAT00,R0	
16583	064252	173010		W12: SUBD	(R0),ACO	;TEST INSTRUCTION
16584	064254	170205		STFPS	R5	;GET FPS
16585	064256	170011		SETD		;SET DOUBLE MODE
16586	064260	012700	064406	MOV	#MPAT00,R0	
16587	064264	174010		STD	ACO,(R0)	;GET THE RESULT
16588	064266	012701	064406	MOV	#MPAT00,R1	
16589	064272	012702	000004	MOV	#4,R2	
16590	064276	022021		W13: CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
16591	064300	001401		BEQ	W14	
(2)	064302	104000		EMT		
16592	064304	077204		W14: SOB	R2,W13	
16593	064306	022705	000204	CMP	#204,R5	;IS FPS CORRECT?
16594	064312	001401		BEQ	W15	
(2)	064314	104000		EMT		
16595	064316	012700	000200	W15: MOV	#200,R0	
16596	064322	170100		LDFPS	R0	;SET DOUBLE MODE
16597	064324	012700	064406	MOV	#MPAT00,R0	;LOAD ACO=0
16598	064330	172410		LDD	(R0),ACO	

```

16599 064332 005000          CLR      R0
16600 064334 170100          LDFPS   R0          ;ENTER FLOATING MODE.
16601 064336 012700 064406    MOV      @WPAT00,R0
16602 064342 173010          SUBF    (R0),AC0    ;TEST INSTRUCTION.
16603 064344 170205          STFPS   R5          ;GET FPS
16604 064346 170011          SETD    ;RESET TO DOUBLE MODE
16605 064350 012700 064406    MOV      @WPAT00,R0 ;GET THE RESULT.
16606 064354 174010          STD     AC0,(R0)
16607 064356 012701 064406    MOV      @WPAT00,R1
16608 064362 012702 000004    MOV      @4,R2
16609 064366 022021          W17:    CMP      (R0)+,(R1)+ ;IS RESULT CORRECT?
16610 064370 001401          BEQ     W20
16611 064372 104000          EMT
16612 064374 077204          W20:    SOB     R2,W17
16613 064376 022705 000004    CMP      @4,R5          ;IS FPS CORRECT?
16614 064402 001411          BEQ     WDONE
16615 064404 104000          EMT
16614 064406 000000          WPAT00: .WORD 0
16616 064410 000000          WPAT01:      0
16617 064412 000000          WPAT02:      0
16618 064414 000000          WPAT03:      0
16619
16620 064416 000000          WDAT00: .WORD 0
16621 064420 000000          WDAT01:      0
16622 064422 000000          WDAT02:      0
16623 064424 000000          WDAT03:      0
16624
16625 064426          WDONE:
16626 064426 004767 040120    JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
16627                                     ;SEE IF THE USER HAS EXPRESSED
16628                                     ;THE DESIRE TO CHANGE THE SOFTWARE
16629                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
16630                                     ;THE USER TYPED CONTROL G?).
16631
16632                                     ;*****
16633                                     ;TEST 457      ADDD AND SUB WITH FSRC=0
16634                                     ;*****
16635 (2) 064432          TS457:
16636 064432 012700 000200    MOV      @200,R0
16637 064436 170100          LDFPS   R0          ;SET DOUBLE MODE
16638 064440 012700 064772    MOV      @XPAT00,R0 ;SET AC0 TO POSITIVE
16639 064444 172410          LDD     (R0),AC0
16640 064446 012700 065002    MOV      @XPAT10,R0 ;FSRC=0
16641 064452 172010          X2:    ADDD   (R0),AC0    ;TEST INSTRUCTION
16642 064454 170205          STFPS   R5
16643 064456 170011          SETD    ;GET RESULT.
16644 064460 012700 064762    MOV      @XDAT00,R0
16645 064464 174010          STD     AC0,(R0)
16646 064466 012701 064772    MOV      @XPAT00,R1
16647 064472 012702 000004    MOV      @4,R2
16648 064476 022021          X3:    CMP      (R0)+,(R1)+ ;IS RESULT CORRECT?
16649 064500 001401          BEQ     X4
16650 (2) 064502 104000          EMT
16651 064504 077204          X4:    SOB     R2,X3

```

16650	064506	012704	000200		MOV	#200,R4	
16651	064512	020405			CMP	R4,R5	;IS FPS CORRECT?
16652	064514	001401			BEQ	X5	
(2)	064516	104000			EMT		
16653	064520	012700	000200	X5:	MOV	#200,R0	
16654	064524	170100			LDFPS	R0	;SET DOUBLE MODE
16655	064526	012700	065012		MOV	#XPAT20,R0	;SET ACO TO
16656	064532	172410			LDD	(R0),ACO	
16657	064534	012700	065002		MOV	#XPAT10,R0	;FSRC=0
16658	064540	172010		X6:	ADD	(R0),ACO	;TEST INSTRUCTION
16659	064542	170205			STFPS	R5	
16660	064544	170011			SETD		
16661	064545	012700	064762		MOV	#XDAT00,R0	;GET RESULT
16662	064552	174010			STD	ACO,(R0)	
16663	064554	012701	065012		MOV	#XPAT20,R1	
16664	064560	012702	000004		MOV	#4,R2	
16665	064564	022021		X7:	CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
16666	064566	001401			BEQ	X10	
(2)	064570	104000			EMT		
16667	064572	077204		X10:	SOB	R2,X7	
16668	064574	012704	000210		MOV	#210,R4	
16669	064600	020405			CMP	R4,R5	;IS FPS CORRECT?
16670	064602	001401			BEQ	X11	
(2)	064604	104000			EMT		
16671	064606	012700	000200	X11:	MOV	#200,R0	
16672	064612	170100			LDFPS	R0	;SET DOUBLE MODE
16673	064614	012700	064772		MOV	#XPAT00,R0	;SET ACO TO NON-ZERO
16674	064620	172410			LDD	(R0),ACO	
16675	064622	012700	065002		MOV	#XPAT10,R0	;FSRC=0
16676	064626	173010		X12:	SUBD	(R0),ACO	;TEST INSTRUCTION
16677	064630	170205			STFPS	R5	
16678	064632	170011			SETD		
16679	064634	012700	064762		MOV	#XDAT00,R0	;GET RESULT
16680	064640	174010			STD	ACO,(R0)	
16681	064642	012701	064772		MOV	#XPAT00,R1	
16682	064646	012702	000004		MOV	#4,R2	
16683	064652	022021		X13:	CMP	(R0)+,(R1)+	;IS RESULT CORRECT?
16684	064654	001401			BEQ	X14	
(2)	064656	104000			EMT		
16685	064660	077204		X14:	SOB	R2,X13	
16686	064662	012704	000200		MOV	#200,R4	;IS FPS CORRECT?
16687	064666	020405			CMP	R4,R5	
16688	064670	001401			BEQ	X15	
(2)	064672	104000			EMT		
16689	064674	012700	000200	X15:	MOV	#200,R0	
16690	064700	170100			LDFPS	R0	;SET DOUBLE MODE
16691	064702	012700	065012		MOV	#XPAT20,R0	;SET ACO=A NEGATIVE
16692	064706	172410			LDD	(R0),ACO	
16693	064710	012700	065002		MOV	#XPAT10,R0	;FSRC=0
16694	064714	173010		X16:	SUBD	(R0),ACO	;TEST INSTRUCTION.
16695	064716	170205			STFPS	R5	
16696	064720	170011			SETD		
16697	064722	012700	064762		MOV	#XDAT00,R0	;GET RESULT
16698	064726	174010			STD	ACO,(R0)	
16699	064730	012701	065012		MOV	#XPAT20,R1	
16700	064734	012702	000004		MOV	#4,R2	

66

16701 064740 022021
 16702 064742 001401
 (2) 064744 104000
 16703 064746 077204
 16704 064750 012704 000210
 16705 064754 020405
 16706 064756 001421
 (2) 064760 104000
 16707 064762 000000
 16708 064764 000000
 16709 064766 000000
 16710 064770 000000
 16711
 16712 064772 010421
 16713 064774 021042
 16714 064776 031463
 16715 065000 042104
 16716
 16717 065002 000000
 16718 065004 000000
 16719 065006 000000
 16720 065010 000000
 16721 065012 104210
 16722 065014 114631
 16723 065016 125252
 16724 065020 135673
 16725
 16726 065022
 (1) 065022 004767 037524
 (1)
 (1)
 (1)
 (1)
 (1)
 16727
 16735
 (2)
 (3)
 (2) 065026
 16736 065026 005037 065202
 16737 065032 012737 065222 065204
 16738 065040 012737 065232 065206
 16739 065046 012737 000210 065210
 16740 065054 012700 000200
 16741 065060 170100
 16742 065062 012700 065242
 16743 065066 172410
 16744 065070 013700 065204
 16745 065074 173010
 16746 065076 170205
 16747 065100 170011
 16748 065102 012700 065212
 16749 065106 174010
 16750 065110 012702 000004
 16751 065114 013701 065206
 16752 065120 022021
 16753 065122 001401

X17: CMP (R0), (R1) ; IS RESULT CORRECT?
 BEQ X20
 EMT ;
 X20: SOB R2, X17
 MOV #210, R4 ; IS FPS CORRECT?
 CMP R4, R5
 BEQ XDONE
 EMT ;
 XDAT00: .WORD 0
 XDAT01: 0
 XDAT02: 0
 XDAT03: 0
 XPAT00: .WORD 010421
 XPAT01: 021042
 XPAT02: 031463
 XPAT03: 042104
 XPAT10: .WORD 0
 XPAT11: 0
 XPAT12: 0
 XPAT13: 0
 XPAT20: .WORD 104210
 XPAT21: 114631
 XPAT22: 125252
 XPAT23: 135673
 XDONE: JSR PC, RSET ; GO INITIALIZE THE FPS AND STACK; AND
 ; SEE IF THE USER HAS EXPRESSED
 ; THE DESIRE TO CHANGE THE SOFTWARE
 ; VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ; THE USER TYPED CONTROL G?).
 ;*****
 ;TEST 460 SUBD WITH AC=0 TEST
 ;*****
 TS460:
 CLR @YFLAG
 MOV @YPAT00, @YTMP1 ;P
 MOV @YPAT10, @YTMP2 ;N
 MOV #210, @YTMP3
 Y1: MOV #200, R0
 LDFPS R0 ; SET DOUBLE MODE
 MOV @YPAT20, R0 ; SET ACO=0
 LDD (R0), ACO
 MOV @YTMP1, R0
 Y2: SUBD (R0), ACO ; TEST INSTRUCTION
 STFPS R5
 SETD
 MOV @YDAT00, R0 ; GET RESULT
 STD ACO, (R0)
 MOV #4, R2
 MOV @YTMP2, R1 ; CHECK RESULT.
 Y3: CMP (R0), (R1)
 BEQ 14

```

(2) 065124 104000
16754 065126 077204
16755 065130 023705 065210
16756 065134 001401
(2) 065136 104000
16757 065140 005737 065202
16758 065144 001015
16759 065146 012737 177777 065202
16760 065154 012737 065232 065204
16761 065162 012737 065222 065206
16762 065170 012737 000200 065210
16763 065176 000726
16764 065200 000424
16765
16766 065202 000000
16767 065204 000000
16768 065206 000000
16769 065210 000000
16770
16771 065212 000000
16772 065214 000000
16773 065216 000000
16774 065220 000000
16775
16776 065222 063146
16777 065224 052525
16778 065226 042104
16779 065230 167356
16780
16781 065232 163146
16782 065234 052525
16783 065236 042104
16784 065240 167356
16785
16786 065242 000000
16787 065244 000000
16788 065246 000000
16789 065250 000000
16790
16791 065252
(1) 065252 004767 037274
(1)
(1)
(1)
(1)
16800
(2)
(3)
(2) 065256
16801 065256 005067 000134
16802 065262 012737 065434 065420
16803 065270 012737 000200 065422
16804 065276 012700 000200
16805 065302 170100
16806 065304 012700 065454
16807 065310 172410

EMT
SOB R2,Y3
CMP @YTMP3,R5 ;FPS CORRECT?
BEQ Y4
EMT
TST @YFLAG ;FINISHED TEST?
BNE Y5
MOV @-1,@YFLAG
MOV @YPAT10,@YTMP1
MOV @YPAT00,@YTMP2
MOV @200,@YTMP3
BR Y1
BR YDONE

Y5:
YFLAG: .WORD 0
YTMP1: 0
YTMP2: 0
YTMP3: 0

YDAT00: .WORD 0
YDAT01: 0
YDAT02: 0
YDAT03: 0

YPAT00: 063146
YPAT01: 052525
YPAT02: 042104
YPAT03: 167356

YPAT10: 163146
YPAT11: 052525
YPAT12: 042104
YPAT13: 167356

YPAT20: 0
YPAT21: 0
YPAT22: 0
YPAT23: 0

YDONE:
JSR PC,.RSET ;GO INITIALIZE THE FF AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 461 ADDD WITH AC=0 TEST
;*****
TS461:
CLR ZFLAG
MOV @ZPAT00,@ZTMP1 ;P
MOV @200,@ZTMP2
Z1: MOV @200,R0
LDFPS R0 ;SET DOUBLE MODE
MOV @ZPAT20,R0 ;SET ACO=0
LDD (R0),ACO

```

```
16808 065312 013700 065420      MOV    @ZTMP1,R0
16809 065316 172010      Z2:   ADDO  (R0),AC0      ;TEST INSTRUCTION
16810 065320 170205      STFPS R5
16811 065322 170011      SETO
16812 065324 012700 065424      MOV    @ZDAT00,R0      ;GET RESULT
16813 065330 174010      STD   AC0,(R0)
16814 065332 012702 000004      MOV    #4,R2
16815 065336 013701 065420      MOV    @ZTMP1,R1      ;RESULT CORRECT?
16816 065342 022021      Z3:   CMP   (R0)*,(R1)*
16817 065344 001401      BEQ   Z4
      (2) 065346 104000      EMT
16818 065350 077204      Z4:   SOB  R2,Z3
16819 065352 023705 065422      CMP   @ZTMP2,R5      ;FPS CORRECT?
16820 065356 001401      BEQ   Z5
      (2) 065360 104000      EMT
16821 065362 005737 065416      Z5:   TST   @ZFLAG      ;FINISHED TEST?
16822 065366 001012      BNE   Z6
16823 065370 012737 177777 065416      MOV    #-1,@ZFLAG
16824 065376 012737 065444 065420      MOV    @ZPAT10,@ZTMP1
16825 065404 012737 000210 065422      MOV    @ZPAT10,@ZTMP2
16826 065412 000731      BR    Z1
16827 065414 000423      Z6:   BR    ZDONE
16828
16829 065416 000000      ZFLAG: .WORD 0
16830 065420 000000      ZTMP1: 0
16831 065422 000000      ZTMP2: 0
16832
16833 065424 000000      ZDAT00: .WORD 0
16834 065426 000000      ZDAT01: 0
16835 065430 000000      ZDAT02: 0
16836 065432 000000      ZDAT03: 0
16837
16838 065434 031463      ZPAT00: 031463
16839 065436 010421      ZPAT01: 010421
16840 065440 146314      ZPAT02: 146314
16841 065442 156735      ZPAT03: 156735
16842
16843 065444 156735      ZPAT10: 156735
16844 065446 167356      ZPAT11: 167356
16845 065450 135673      ZPAT12: 135673
16846 065452 146314      ZPAT13: 146314
16847
16848 065454 000000      ZPAT20: 0
16849 065456 000000      ZPAT21: 0
16850 065460 000000      ZPAT22: 0
16851 065462 000000      ZPAT23: 0
16852
16853 065464      ZDONE:
      (1) 065464 004767 037062      JSR   PC,,RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                                     ;SEE IF THE USER HAS EXPRESSED
      (1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                                     ;THE USER TYPED CONTROL G?).
16854
16855
16863
```

;TEST 462 ADDF AND ADD WITH E(AC)=E(FSRC) TEST AND (BUT FT) TEST
.....

TS462:

(2)
(3)
(2) 065470
16864 065470 012700 003240
16865 065474 170100
16866 065476 012700 065700
16867
16868 065502 172410
16869 065504 012700 065710
16870 065510 172010
16871
16872 065512 012700 065670
16873 065516 174010
16874 065520 012701 065720
16875 065524 012702 000004
16876 065530 022021
16877 065532 001401
(2) 065534 104000
16878 065536 077204
16879
16880
16881
16882
16883 065540 012700 003200
16884 065544 170100
16885 065546 012700 065700
16886 065552 172410
16887 065554 012700 065710
16888 065560 172010
16889
16890 065562 012700 065670
16891 065566 174010
16892 065570 012701 065730
16893 065574 012702 000004
16894 065600 022021
16895 065602 001401
(2) 065604 104000
16896 065606 077204
16897
16898
16899
16900 065610 012700 003200
16901 065614 170100
16902 065616 012700 065700
16903 065622 172410
16904 065624 170001
16905 065626 012700 065750
16906 065632 172010
16907
16908 065634
16909 065634 170011
16910
16911 065636 012700 065670
16912 065642 174010
16913 065644 012701 065760
16914 065650 012702 000002

MOV #3240,R0
LDFPS RO ;SET FIU FIV FD AND FT
MOV #AAPATO,R0 ;FLOWS IN TRAP WILL
;OCCUR
LDD (RO),ACO ;SET UP ACO
MOV #AAPAT1,R0
AA2: ADDD (RO),ACO ;TEST INSTRUCTION
;SHOULD TRUNCATE
AA3: MOV #AADATO,R0
STD ACO,(RO) ;GET THE RESULT
MOV #AAPAT2,R1
MOV #4,R2
AA4: CMP (RO)+,(R1)+ ;CORRECT?
BEQ AA7
EMT
AA7: SOB R2,AA4
;NOW TEST DOUBLE FLOATING ROUND MODE.
;A 1 SHOULD BE ADDED TO THE LSB ON ROUND MODE.
MOV #3200,R0 ;SET FD FIV FIV. FT=0
LDFPS RO
MOV #AAPATO,R0
LDD (RO),ACO ;SET UP ACO OPERAND
MOV #AAPAT1,R0
AA11: ADDD (RO),ACO ;TEST INSTRUCTION
;SHOULD ROUND
AA12: MOV #AADATO,R0
STD ACO,(RO) ;GET THE RESULT
MOV #AAPAT3,R1
MOV #4,R2
AA13: CMP (RO)+,(R1)+ ;CORRECT?
BEQ AA20
EMT
AA20: SOB R2,AA13
;NOW TEST ADDF WITH FT=0. ROUND MODE
MOV #3200,R0 ;FIV=1. FIV=1. FT=0
LDFPS RO
MOV #AAPATO,R0 ;LOAD ACO OFERAND
LDD (RO),ACO
SETF ;ENTER FLOATING MODE
MOV #AAPAT5,R0
AA22: ADDF (RO),ACO ;TEST INSTRUCTION
;SHOULD ROUND
AA23: SETD ;RESET TO DOUBLE
;MODE
MOV #AADATO,R0 ;GET THE RESULT
STD ACO,(RO)
MOV #AAPAT6,R1 ;CORRECT?
MOV #2,R2

166

16915 065654 022021
 16916 065656 001401
 (2) 065660 104000
 16917 065662 077204
 16918 065664 000137 065770
 16919 065670 000000
 16920 065672 000000
 16921 065674 000000
 16922 065676 000000
 16923 065700 000200
 16924 065702 000000
 16925 065704 000000
 16926 065706 000000
 16927 065710 000200
 16928 065712 000000
 16929 065714 000000
 16930 065716 000001
 16931 065720 000400
 16932 065722 000000
 16933 065724 000000
 16934 065726 000000
 16935 065730 000400
 16936 065732 000000
 16937 065734 000000
 16938 065736 000001
 16939 065740 000400
 16940 065742 000000
 16941 065744 100000
 16942 065746 000000
 16943 065750 000200
 16944 065752 000001
 16945 065754 000000
 16946 065756 000000
 16947 065760 000400
 16948 065762 000001
 16949 065764 000000
 16950 065766 000000
 16951 065770
 (1) 065770 004767 036556
 (1)
 (1)
 (1)
 (1)
 16962
 (2)
 (3)
 (2) 065774
 16963
 16964 065774 012704 003200
 16965 066000 170104
 16966 066002 012700 066450
 16967 066006 172410
 16968 066010 012700 066470
 16969 066014 172010
 16970 066016 170205
 16971 066020 012700 066440

AA24: CMP (R0), (R1)
 BEQ AA27
 EMT
 AA27: SOB R2, AA24
 JMP @AADONE
 AADAT0: 0
 0
 0
 0
 AAPAT0: 200
 0
 0
 0
 AAPAT1: 200
 0
 0
 1
 AAPAT2: 400
 0
 0
 0
 AAPAT3: 400
 0
 0
 1
 AAPAT4: 400
 0
 100000
 0
 AAPAT5: 200
 1
 0
 0
 AAPAT6: 400
 1
 0
 0
 AADONE: JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 463 ADDF AND ADD WITH E(AC) LESS THAN E(FSRC) TEST

 TS463:
 ;EXPONENT DIFFERENCE=57-71 (OCT) FD=1
 MOV #3200, R4 ;SET FI' FIV. AND FD
 LDFPS R4
 MOV #CCP0, R0 ;SET ACO OPERAND
 LDD (R0), ACO ;ACO
 MOV #CCP2, R0
 CCX2: ADDD (R0), ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #CCDAT0, R0 ;GET THE RESULT

16972 066024 174010
 16973 066026 012701 066470
 16974 066032 012702 000004
 16975 066036 022021
 16976 066040 001401
 (2) 066042 104000
 16977 066044 077204
 16978 066046 020405
 16979 066050 001401
 (2) 066052 104000
 16980
 16981 066054 012704 003200
 16982 066060 170104
 16983 066062 012700 066450
 16984 066066 172410
 16985 066070 012700 066460
 16986 066074 172010
 16987 066076 170205
 16988 066100 012700 066440
 16989 066104 174010
 16990 066106 012701 066540
 16991 066112 012702 000004
 16992 066116 022021
 16993 066120 001401
 (2) 066122 104000
 16994 066124 077204
 16995 066126 020405
 16996 066130 001401
 (2) 066132 104000
 16997
 16998 066134 012700 066450
 16999 066140 172410
 17000 066142 012704 003000
 17001 066146 170104
 17002 066150 012700 066530
 17003 066154 172010
 17004 066156 170205
 17005 066160 170011
 17006 066162 012700 066440
 17007 066166 174010
 17008 066170 012701 066530
 17009 066174 012702 000002
 17010 066200 022021
 17011 066202 001401
 (2) 066204 104000
 17012 066206 077204
 17013 066210 020405
 17014 066212 001401
 (2) 066214 104000
 17015
 17016 066216 012700 066500
 17017 066222 172410
 17018 066224 012704 003000
 17019 066230 170104
 17020 066232 012700 066520
 17021 066236 172010

STD ACO,(R0)
 MOV #CCP2,R1 ;IS IT CORRECT
 MOV #4,R2
 CCX3: CMP (R0)+,(R1)+
 BEQ CCX6
 EMT ;
 CCX6: SOB R2,CCX3
 CMP R4,R5 ;FPS CORRECT?
 BEQ CCX7
 EMT ;
 ;EXPONENT DIFFERENCE=56-70 (OCT) FD=1
 CCX7: MOV #3200,R4 ;SET FIV,FIV. AND FD
 LDFPS R4
 MOV #CCP0,R0 ;SET ACO OPERAND
 LDD (R0),ACO
 MOV #CCP1,R0 ;FSRC
 CCX8: ADDF (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #CCDATG,R0 ;GET THE RESULT
 STD ACO,(R0)
 MOV #CCP7,R1 ;IS IT CORRECT
 MOV #4,R2
 CCX9: CMP (R0)+,(R1)+
 BEQ CCX12
 EMT ;
 CCX12: SOB R2,CCX9
 CMP R4,R5 ;FPS CORRECT?
 BEQ CCX13
 EMT ;
 ;EXPONENT DIFFERENCE=25-31 (OCT) FD=0
 CCX13: MOV #CCP0,R0 ;SET UP ACO OPERAND.
 LDD (R0),ACO
 MOV #3000,R4 ;SET FIV,FIV. CLEAR FD.
 LDFPS R4
 MOV #CCP6,R0 ;FSRC
 CCX14: ADDF (R0),ACO ;TEST INSTRUCTION
 STFPS R5
 SETD ;REENTER DOUBLE MOVE
 MOV #CCDATG,R0 ;GET THE RESULT
 STD ACO,(R0)
 MOV #CCP6,R1 ;IS THE RESULT CORRECT?
 MOV #2,R2
 CCX15: CMP (R0)+,(R1)+
 BEQ CCX18
 EMT ;
 CCX18: SOB R2,CCX15
 CMP R4,R5
 BEQ CCX19
 EMT ;
 ;EXPONENT DIFFERENCE=24-30 (OCT) FD=0
 CCX19: MOV #CCP3,R0 ;SET UP ACO OPERAND.
 LDD (R0),ACO
 MOV #3000,R4 ;SET FIV,FIV. CLEAR FD.
 LDFPS R4
 MOV #CCP5,R0 ;FSRC
 CCX20: ADDF (R0),ACO ;TEST INSTRUCTION

17022	066240	170205		STFPS	R5	
17023	066242	170011		SETD		;REENTER DOUBLE MOVE
17024	066244	012700	066440	MOV	#CCDAT0,R0	;GET THE RESULT
17025	066250	174010		STD	AC0,(R0)	
17026	066252	012701	066550	MOV	#CCP10,R1	;IS THE RESLT CORRECT?
17027	066256	012702	000002	MOV	#2,R2	
17028	066262	022021		CCX21: CMP	(R0)+,(R1)+	
17029	066264	001401		BEQ	CCX24	
(2)	066266	104000		EMT		;
17030	066270	077204		CCX24: SOB	R2,CCX21	
17031	066272	020405		CMP	R4,R5	
17032	066274	001401		BEQ	CCX25	
(2)	066276	104000		EMT		;
17033					;EXPONENT DIFFERENCE=1 FD=1	
17034	066300	012704	003200	CCX25: MOV	#3200,R4	;SET FIV,FIV, AND FD
17035	066304	170104		LDFPS	R4	
17036	066306	012700	066450	MOV	#CCP0,R0	;SET ACO OPERANC
17037	066312	172410		LDD	(R0),AC0	
17038	066314	012700	066500	MOV	#CCP3,R0	;FSRC
17039	066320	172010		CCX26: ADDD	(R0),AC0	;TEST INSTRUCTION
17040	066322	170205		STFPS	R5	;GET FPS
17041	066324	012700	066440	MOV	#CCDAT0,R0	;GET THE RESULT
17042	066330	174010		STD	AC0,(R0)	
17043	066332	012701	066560	MOV	#CCP11,R1	;IS IT CORRECT
17044	066336	012702	000004	MOV	#4,R2	
17045	066342	022021		CCX27: CMP	(R0)+,(R1)+	
17046	066344	001401		BEQ	CCX30	
(2)	066346	104000		EMT		;
17047	066350	077204		CCX30: SOB	R2,CCX27	
17048	066352	020405		CMP	R4,R5	;FPS CORRECT?
17049	066354	001401		BEQ	CCX31	
(2)	066356	104000		EMT		;
17050					;EXPONENT DIFFERENCE=100-144 (OCT) FD=1	
17051	066360	012704	003200	CCX31: MOV	#3200,R4	;SET FIV,FIV, AND FD
17052	066364	170104		LDFPS	R4	
17053	066366	012700	066450	MOV	#CCP0,R0	;SET ACO OPERAND
17054	066372	172410		LDD	(R0),AC0	
17055	066374	012700	066510	MOV	#CCP4,R0	;FSRC
17056	066400	172010		CCX32: ADDD	(R0),AC0	;TEST INSTRUCTION
17057	066402	170205		STFPS	R5	;GET FPS
17058	066404	012700	066440	MOV	#CCDAT0,R0	;GET THE RESULT
17059	066410	174010		STD	AC0,(R0)	
17060	066412	012701	066510	MOV	#CCP4,R1	;IS IT CORRECT
17061	066416	012702	000004	MOV	#4,R2	
17062	066422	022021		CCX33: CMP	(R0)+,(R1)+	
17063	066424	001401		BEQ	CCX36	
(2)	066426	104000		EMT		;
17064	066430	077204		CCX36: SOB	R2,CCX33	
17065	066432	020405		CMP	R4,R5	;FPS CORRECT?
17066	066434	001461		BEQ	CCXDONE	
(2)	066436	104000		EMT		;
17067	066440	000000		CCDAT0: 0		
17068	066442	000000		0		
17069	066444	000000		0		
17070	066446	000000		0		
17071	066450	000200		CCP0: 200		;E(AC)=1

```

17072 066452 000000 0
17073 066454 000000 0
17074 066456 000000 0
17075 066460 016200 CCP1: 16200 ;E(FSRC)=E(AC)+56=57
17076 066462 000000 0 ; ; =71(OCT)
17077 066464 000000 0
17078 066466 000000 0
17079 066470 016400 CCP2: 16400 ;E(FSRC)=E(AC)+57=58
17080 066472 000000 0 ; ; =72(OCT)
17081 066474 000000 0
17082 066476 000000 0
17083 066500 000400 CCP3: 400 ;E(FSRC)=E(AC)+1=2
17084 066502 000000 0
17085 066504 000000 0
17086 066506 000000 0
17087 066510 031200 CCP4: 31200 ;E(FSRC)=E(AC)+100=101=145(OCT)
17088 066512 000000 0
17089 066514 000000 0
17090 066516 000000 0
17091 066520 006200 CCP5: 6200 ;E(FSRC)=E(AC)+24=25=31(OCT)
17092 066522 000000 0
17093 066524 000000 0
17094 066526 000000 0
17095 066530 006400 CCP6: 6400 ;E(FSRC)=E(AC)+25=26=32(OCT)
17096 066532 000000 0
17097 066534 000000 0
17098 066536 000000 0
17099 066540 016200 CCP7: 16200 ;CCP1 RES
17100 066542 000000 0
17101 066544 000000 0
17102 066546 000001 1
17103 066550 006200 CCP10: 6200 ;CCP5 RES
17104 066552 000001 1
17105 066554 000000 0
17106 066556 000000 0
17107 066560 000500 CCP11: 500 ;CCP3 RES
17108 066562 000000 0
17109 066564 000000 0
17110 066566 000000 0
17111 066570 000200 CCP12: 200 ;BAD CONSTANT
17112 066572 000000 0 ;RES CCP2,CCP4
17113 066574 000000 0
17114 066576 000000 0
17115
17116 066600 CCXDONE:
(1) 066600 004767 035746 JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
17117
17128 (2) ;*****
;TEST 464 ADDF AND ADDD WITH E(AC) GREATER THAN E(FSRC) TEST
(3) ;*****
(2) 066604 TS464:
17129 ;EXPONENT DIFFERENCE=57=71 (OCT) FD=1

```


17130	066604	012704	003200		MOV	#3200,R4	;SET FIV FIV. AND FD
17131	066610	170104			LDFPS	R4	
17132	066612	012700	067300		MOV	##BPAT2,R0	;SET ACO OPERAND.
17133	066616	172410			LDD	(R0),ACO	
17134	066620	012700	067270		MOV	##BPAT1,R0	;FSRC
17135	066624	172010		BB2:	ADDF	(R0),ACO	;TEST INSTRUCTION
17136	066626	170205			STFPS	R5	
17137	066630	012700	067250	BB3:	MOV	##BDATO,R0	;GET THE RESULT
17138	066634	174010			STD	ACO,(R0)	
17139	066636	012701	067300		MOV	##BPAT2,R1	;RESULT CORRECT?
17140	066642	012702	000004		MOV	#4,R2	
17141	066646	022021		BB4:	CHP	(R0).,(R1).	
17142	066650	001401			BEQ	BB5	
(2)	066652	104000			EMT		
17143	066654	077204		BB5:	SOB	R2,BB4	
17144							;WAS FPS CORRECT?
17145	066656	020405			CHP	R4,R5	
17146	066660	001401			BEQ	BB6	
(2)	066662	104000			EMT		
17147							;EXPONENT DIFFERENCE=56-70 (OCT) FD=1
17148	066664	012704	003200	BB6:	MOV	#3200,R4	;SET FIV,FIV. AND FD
17149	066670	170104			LDFPS	R4	
17150	066672	012700	067320		MOV	##BPAT4,R0	;SET ACO OPERAND
17151	066676	172410			LDD	(R0),ACO	
17152	066700	012700	067270		MOV	##BPAT1,R0	;FSRC
17153	066704	172010		BB7:	ADDF	(R0),ACO	;TEST INSTRUCTION
17154	066706	170205			STFPS	R5	;GET FPS
17155	066710	012700	067250		MOV	##BDATO,R0	;GET THE RESULT
17156	066714	174010			STD	ACO,(R0)	
17157	066716	012701	067360		MOV	##BP10,R1	;IS IT CORRECT
17158	066722	012702	000004		MOV	#4,R2	
17159	066726	022021		BB10:	CHP	(R0).,(R1).	
17160	066730	001401			BEQ	BB13	
(2)	066732	104000			EMT		
17161	066734	077204		BB13:	SOB	R2,BB10	
17162	066736	020405			CHP	R4,R5	;FPS CORRECT?
17163	066740	001401			BEQ	BB14	
(2)	066742	104000			EMT		
17164							;EXPONENT DIFFERENCE=25-31 (OCT) FD=0
17165	066744	012700	067260	BB14:	MOV	##BPATO,R0	;SET UP ACO OPERAND
17166	066750	172410			LDD	(R0),ACO	
17167	066752	012704	003000		MOV	#3000,R4	;SET FIV AND FIV
17168							;CLEAR FD
17169	066756	170104			LDFPS	R4	
17170	066760	012700	067270		MOV	##BPAT1,R0	;FSRC
17171	066764	172010		BB15:	ADDF	(R0),ACO	;TEST INSTRUCTION
17172	066766	170205			STFPS	R5	
17173	066770	170011			SETD		;REENTERED DOUBLE MODE.
17174	066772	012700	067250		MOV	##BDATO,R0	;GET THE RESULT
17175	066776	174010			STD	ACO,(R0)	
17176	067000	012701	067260		MOV	##BPATO,R1	;IS THE RESULT
17177	067004	012702	000002		MOV	#2,R2	;CORRECT?
17178	067010	022021		BB16:	CHP	(R0).,(R1).	
17179	067012	001401			BEQ	BB17	
(2)	067014	104000			EMT		
17180	067016	077204		BB17:	SOB	R2,BB16	

17181	067020	020405			CMP	R4,R5		;IS FPS CORRECT?
17182	067022	001401			BEQ	BB20		
(2)	067024	104000			EMT			
17183					;EXPONENT DIFFERENCE=24-30 (OCT)			
17184	067026	012700	067310	BB20:	MOV	##BPAT3,R0		;SET UP ACO OPERAND.
17185	067032	172410			LDD	(R0),ACO		
17186	067034	012704	003000		MOV	#3000,R4		;SET FIU,FIV. CLEAR FD
17187	067040	170104			LDFPS	R4		
17188	067042	012700	067270		MOV	##BPAT1,R0		;FSRC
17189	067046	172010		BB21:	ADDF	(R0),ACO		;TEST INSTRUCTION
17190	067050	170205			STFPS	R5		
17191	067052	170011			SETD			;REENTER DOUBLE MODE
17192	067054	012700	067250		MOV	##BDATO,R0		;GET THE RESULT
17193	067060	174010			STD	ACO,(R0)		
17194	067062	012701	067350		MOV	##BP7,R1		;IS THE RESULT CORRECT
17195	067066	012702	000002		MOV	#2,R2		
17196	067072	022021		BB22:	CMP	(R0)+,(R1)+		
17197	067074	001401			BEQ	BB25		
(2)	067076	104000			EMT			
17198	067100	077204		BB25:	SQB	R2,BB22		
17199	067102	020405			CMP	R4,R5		
17200	067104	001401			BEQ	BB26		
(2)	067106	104000			EMT			
17201					;EXPONENT DIFFERENCE=1			
17202	067110	012704	003200	BB26:	MOV	#3200,R4		
17203	067114	170104			LDFPS	R4		;SET UP ACO OPERAND
17204	067116	012700	067330		MOV	##BPAT5,R0		
17205	067122	172410			LDD	(R0),ACO		
17206	067124	012700	067270		MOV	##BPAT1,R0		;FSRC
17207	067130	172010		BB27:	ADDF	(R0),ACO		;TEST INSTRUCTION
17208	067132	170205			STFPS	R5		
17209	067134	012700	067250		MOV	##BDATO,R0		;GET THE RESULT.
17210	067140	174010			STD	ACO,(R0)		
17211	067142	012701	067370		MOV	##BP11,R1		;IS IT CORRECT?
17212	067146	012702	000004		MOV	#4,R2		
17213	067152	022021		BB30:	CMP	(R0)+,(R1)+		
17214	067154	001401			BEQ	BB31		
(2)	067156	104000			EMT			
17215	067160	077204		BB31:	SQB	R2,BB30		
17216	067162	020405			CMP	R4,R5		;IS FPS CORRECT
17217	067164	001401			BEQ	BB32		
(2)	067166	104000			EMT			
17218					;EXPONENT DIFFERENCE=100-144 (OCT)			
17219	067170	012704	003200	BB32:	MOV	#3200,R4		
17220	067174	170104			LDFPS	R4		;SET FIV,FIV AND FD
17221	067176	012700	067340		MOV	##BPAT6,R0		;SET UP ACO OPERAND.
17222	067202	172410			LDD	(R0),ACO		
17223	067204	012700	067270		MOV	##BPAT1,R0		;FSRC
17224	067210	172010		BB33:	ADDF	(R0),ACO		;TEST INSTRUCTION
17225	067212	170205			STFPS	R5		
17226	067214	012700	067250		MOV	##BDATO,R0		;GET THE RESULT
17227	067220	174010			STD	ACO,(R0)		
17228	067222	012701	067340		MOV	##BPAT6,R1		;IS IT CORRECT
17229	067226	012702	000004		MOV	#4,R2		
17230	067232	022021		BB34:	CMP	(R0)+,(R1)+		
17231	067234	001401			BEQ	BB35		

(2) 067236 104000
 17232 067240 077204
 17233 067242 020405
 17234 067244 001455
 (2) 067246 104000
 17235 067250 000000
 17236 067252 000000
 17237 067254 000000
 17238 067256 000000
 17239 067260 006400
 17240 067262 000000
 17241 067264 000000
 17242 067266 000000
 17243 067270 000200
 17244 067272 000000
 17245 067274 000000
 17246 067276 000000
 17247 067300 016400
 17248 067302 000000
 17249 067304 000000
 17250 067306 000000
 17251 067310 006200
 17252 067312 000000
 17253 067314 000000
 17254 067316 000000
 17255 067320 016200
 17256 067322 000000
 17257 067324 000000
 17258 067326 000000
 17259 067330 000400
 17260 067332 000000
 17261 067334 000000
 17262 067336 000000
 17263 067340 031200
 17264 067342 000000
 17265 067344 000000
 17266 067346 000000
 17267 067350 006200
 17268 067352 000001
 17269 067354 000000
 17270 067356 000000
 17271 067360 016200
 17272 067362 000000
 17273 067364 000000
 17274 067366 000001
 17275 067370 000500
 17276 067372 000000
 17277 067374 000000
 17278 067376 000000
 17279 067400
 (1) 067400 004767 035146
 (1)
 (1)
 (1)
 (1)
 17287

```

EMT
BB35: SOB R2, BB34
      CMP R4, R5 ; IS FPS CORRECT
      BEQ BB DONE
      EMT
BB DATO: 0
        0
        0
BB PATO: 6400 ; F(AC)=E(FSRC)*25=26
        0 ; =32(OCT)
        0
BB PAT1: 200 ; E(FSRC)=1
        0
        0
BB PAT2: 16400 ; E(AC)=E(FSRC)*57=58
        0 ; =72(OCT)
        0
BB PAT3: 6200 ; E(AC)=E(FSRC)*24=25
        0 ; =31(OCT)
        0
BB PAT4: 16200 ; E(AC)=E(FSRC)*56=57
        0 ; =71(OCT)
        0
BB PAT5: 400 ; E(AC)=E(FSRC)*1=2
        0
        0
BB PAT6: 31200 ; E(AC)=E(FSRC)*100=101
        0 ; =145(OCT)
        0
BB P7: 6200 ; BB PAT3 RES
        1
        0
BB P10: 16200 ; BB PAT4 RES
        0
        0
BB P11: 500 ; BB PAT5 RES
        1
        0
        0
BB DONE: JSR PC, .RSET ; GO INITIALIZE THE FPS AND STACK; AND
        ; SEE IF THE USER HAS EXPRESSED
        ; THE DESIRE TO CHANGE THE SOFTWARE
        ; VIRTUAL CONSOLE SWITCH REGISTER (HAS
        ; THE USER TYPED CONTROL G?).
;*****

```

;TEST 465 ADDD WITH NEGATIVE OPRANDS TEST
;.....

TS465:
;BOTH OPERANDS NEGATIVE

DD2: MOV #3200,R4 ;SET FIO, FIV, AND FD
LDFPS R4
MOV #DDP1,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #DDP1,R0 ;FSRC
ADD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #DDDAT0,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #DDP9,R1 ;IS IT CORRECT
MOV #4,R2
DD3: CMP (R0)+,(R1)+
BEQ DD6
EMT ;
DD6: SOB R2,DD3
BIS #10,R4
CMP R4,R5 ;FPS CORRECT?
BEQ DD7
EMT

;AC POS FSRC NEG AC=-FSRC ;

DD7: MOV #3200,R4 ;SET FIO, FIV, AND FD
LDFPS R4
MOV #DDP2,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #DDP1,R0 ;FSRC
DD8: ADD (R0),ACC ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #DDDAT0,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #DDP0,R1 ;IS IT CORRECT
MOV #4,R2
DD10: CMP (R0)+,(R1)+
BEQ DD11
EMT ;
DD11: SOB R2,DD10
BIS #4,R4
CMP R4,R5 ;FPS CORRECT?
BEQ DD12
EMT

;AC NEG FSRC POS AC=-FSRC ;

DD12: MOV #3200,R4 ;SET FIO, FIV, AND FD
LDFPS R4
MOV #DDP1,R0 ;SET ACO OPERAND
LDD (R0),ACO
MOV #DDP2,R0 ;FSRC
DD13: ADD (R0),ACC ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #DDDAT0,R0 ;GET THE RESULT
STD ACO,(R0)
MOV #DDP0,R1 ;IS IT CORRECT
MOV #4,R2
DD14: CMP (R0)+,(R1)+

(2)
(3)
(2) 067404
17288
17289 067404 012704 003200
17290 067410 170104
17291 067412 012700 070170
17292 067416 172410
17293 067420 012700 070170
17294 067424 172010
17295 067426 170205
17296 067430 012700 070150
17297 067434 174010
17298 067436 012701 070270
17299 067442 012702 000004
17300 067446 022021
17301 067450 001401
(2) 067452 104000
17302 067454 077204
17303 067456 052704 000010
17304 067462 020405
17305 067464 001401
(2) 067466 104000
17306
17307 067470 012704 003200
17308 067474 170104
17309 067476 012700 070200
17310 067502 172410
17311 067504 012700 070170
17312 067510 172010
17313 067512 170205
17314 067514 012700 070150
17315 067520 174010
17316 067522 012701 070160
17317 067526 012702 000004
17318 067532 022021
17319 067534 001401
(2) 067536 104000
17320 067540 077204
17321 067542 052704 000004
17322 067546 020405
17323 067550 001401
(2) 067552 104000
17324
17325 067554 012704 003200
17326 067556 170104
17327 067562 012700 070170
17328 067566 172410
17329 067570 012700 070200
17330 067574 172010
17331 067576 170205
17332 067600 012700 070150
17333 067604 174010
17334 067606 012701 070160
17335 067612 012702 000004
17336 067616 022021

17337	067620	001401		BEQ	DD15		
(2)	067622	104000		EMT			
17338	067624	077204		DD15:	SOB	R2,DD14	
17339	067626	052704	000004		BIS	#4,R4	
17340	067632	020405			CMP	R4,R5	;EPS CORRECT?
17341	067634	001401			BEQ	DD16	
(2)	067636	104000			EMT		
17342				;AC0 POC	FSRC	NEG	/AC/ > /FSRC/
17343	067640	012704	003200	DD16:	MOV	#3200,R4	;SET FIV, FIV AND FD
17344	067644	170104			LDFPS	R4	
17345	067646	012700	070210		MOV	#DDP3,R0	;SET ACO OPERAND
17346	067652	172410			LDD	(R0),ACO	
17347	067654	012700	070240		MOV	#DDP6,R0	;ESPC
17348	067660	172010		DD17:	ADD	(R0),ACO	;TEST INSTRUCTION
17349	067662	170205			STFPS	R5	;GET FPS
17350	067664	012700	070150		MOV	#DDDAT0,R0	;GET THE RESULT
17351	067670	174010			STD	ACO,(R0)	
17352	067672	012701	070250		MOV	#DDP7,R1	;IS IT CORRECT
17353	067676	012702	000004		MOV	#4,R2	
17354	067702	022021		DD18:	CMP	(R0)*,(R1)*	
17355	067704	001401			BEQ	DD21	
(2)	067706	104000			EMT		
17356	067710	077204		DD21:	SOB	R2,DD18	
17357	067712	020405			CMP	R4,R5	;EPS CORRECT?
17358	067714	001401			BEQ	DD22	
(2)	067716	104000			EMT		
17359				;AC NEG	FSRC	POS	/FSRC/ > /AC/
17360	067720	012704	003200	DD22:	MOV	#3200,R4	;SET FIO,FIV, AND FD
17361	067724	170104			LDFPS	R4	
17362	067726	012700	070240		MOV	#DDP6,R0	;SET ACO OPERAND
17363	067732	172410			LDD	(R0),ACO	
17364	067734	012700	070210		MOV	#DDP3,R0	;FSPC
17365	067740	172010		DD23:	ADD	(R0),ACO	;TEST INSTRUCTION
17366	067742	170205			STFPS	R5	;GET FPS
17367	067744	012700	070150		MOV	#DDDAT0,R0	;GET THE RESULT
17368	067750	174010			STD	ACO,(R0)	
17369	067752	012701	070250		MOV	#DDP7,R1	;IS IT CORRECT?
17370	067756	012702	000004		MOV	#4,R2	
17371	067762	022021		DD24:	CMP	(R0)*,(R1)*	
17372	067764	001401			BEQ	DD27	
(2)	067766	104000			EMT		
17373	067770	077204		DD27:	SOB	R2,DD24	
17374	067772	020405			CMP	R4,R5	;FPS CORRECT?
17375	067774	001401			BEQ	DD30	
(2)	067776	104000			EMT		
17376				;AC0 POS	FSRC	NEG	/AC/ < /FSRC/
17377	070000	012704	003200	DD30:	MOV	#3200,R4	;SET FIO,FIV,AND FD
17378	070004	170104			LDFPS	R4	
17379	070006	012700	070220		MOV	#DDP4,R0	;SET ACO OPERAND
17380	070012	172410			LDD	(R0),ACO	
17381	070014	012700	070230		MOV	#DDP5,R0	;FSPC
17382	070020	172010		DD31:	ADD	(R0),ACO	;TEST INSTRUCTION
17383	070022	170205			STFPS	R5	;GET FPS
17384	070024	012700	070150		MOV	#DDDAT0,R0	;GET THE RESULT
17385	070030	174010			STD	ACO,(R0)	
17386	070032	012701	070260		MOV	#DDP8,R1	;IS IT CORRECT

17387	070036	012702	000004		MOV	#4,R2		
17388	070042	022021		DD32:	CMF	(R0)..(R1).		
17389	070044	001401			BEQ	DD35		
(2)	070046	104000			EMT			
17390	070050	077204		DD35:	SQB	R2,DD32		
17391	070052	052704	000010		BIS	#10,R4		
17392	070056	020405			CMF	R4,R5		;FPS CORRECT?
17393	070060	001401			BEQ	DD36		
(2)	070062	104000			EMT			
17394								
17395	070064	012704	003200		;ACO NEG	FSRC POS		/FSRC/ << /AC/
17396	070070	170104		DD36:	MOV	#3200,R4		;SET F10, F1V, AND FD
17397	070072	012700	070230		LDFPS	R4		
17398	070076	172410			MOV	#DDP5,R0		;SET ACO OPERAND
17399	070100	012700	070220		LDD	(R0),ACO		
17400	070104	172010			MOV	#DDP4,R0		;FSPC
17401	070106	170205		DD37:	ADD	(R0),ACO		;TEST INSTRUCTION
17402	070110	012700	070150		STFPS	R5		;GET FPS
17403	070114	174010			MOV	#DDATO,R0		;GET THE RESULT
17404	070116	012701	070260		STD	ACO,(R0)		
17405	070122	012702	000004		MOV	#DDP8,R1		;IS IT CORRECT
17406	070126	022021			MOV	#4,R2		
17407	070130	001401		DD38:	CMF	(R0)..(R1).		
(2)	070132	104000			BEQ	DD41		
17408	070134	077204			EMT			
17409	070136	052704	000010	DD41:	SQB	R2,DD38		
17410	070142	020405			BIS	#10,R4		
17411	070144	001455			CMF	R4,R5		;FPS CORRECT?
(2)	070146	104000			BEQ	DDDONE		
17412	070150	000000			EMT			
17413	070152	000000		DDATO:	0			
17414	070154	000000			0			
17415	070156	000000			0			
17416	070160	000000		DDP0:	0			
17417	070162	000000			0			
17418	070164	000000			0			
17419	070166	000000			0			
17420	070170	100200		DDP1:	100200			; -DDP2
17421	070172	000000			0			
17422	070174	000000			0			
17423	070176	000000			0			
17424	070200	000200		DDP2:	200			; -DDP1
17425	070202	000000			0			
17426	070204	000000			0			
17427	070206	000000			0			
17428	070210	001100		DDP3:	1100			;EXP=4
17429	070212	000000			0			;FRAC=...110...
17430	070214	000000			0			
17431	070216	000000			0			
17432	070220	000600		DDP4:	600			;EXP=3
17433	070222	000000			0			;FRAC=...100...
17434	070224	000000			0			
17435	070226	000000			0			
17436	070230	101100		DDP5:	101100			; -DDP3
17437	070232	000000			0			
17438	070234	000000			0			

17439 070236 000000
 17440 070240 100600
 17441 070242 000000
 17442 070244 000000
 17443 070246 000000
 17444 070250 001000
 17445 070252 000000
 17446 070254 000000
 17447 070256 000000
 17448 070260 101000
 17449 070262 000000
 17450 070264 000000
 17451 070266 000000
 17452 070270 100400
 17453 070272 000000
 17454 070274 000000
 17455 070276 000000
 17456 070300
 (1) 070300 004767 034246
 (1)
 (1)
 (1)
 (1)
 17464
 (2)
 (3)
 (2) 070304
 17465
 17466 070304 012704 003200
 17467 070310 170104
 17468 070312 012700 070476
 17469 070316 172410
 17470 070320 012700 070476
 17471 070324 173010
 17472 070326 170205
 17473 070330 012700 070454
 17474 070334 174010
 17475 070336 012701 070464
 17476 070342 012702 000004
 17477 070346 022021
 17478 070350 001401
 (2) 070352 104000
 17479 070354 077204
 17480 070356 052704 000004
 17481 070362 020405
 17482 070364 001401
 (2) 070366 104000
 17483
 17484 070370 012704 003200
 17485 070374 170104
 17486 070376 012700 070516
 17487 070402 172410
 17488 070404 012700 070516
 17489 070410 173010
 17490 070412 170205
 17491 070414 012700 070454

0
 DDP6: 100600 ; -DDP4
 0
 0
 0
 DDP7: 1000 ; DDP3+DDP6
 0
 0
 0
 DDP8: 101000 ; DDP5+DDP4
 0
 0
 0
 DDP9: 100400 ; DDP1+DDP1
 0
 0
 0
 DDDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).
 ;*****
 ;TEST 466 SUBD TEST
 ;*****
 TS466:
 ; USE POSITIVE OPERANDS
 MOV #3200,R4 ;SET FIU, FIV, AND FD
 LDFPS R4
 MOV #EEP1,R0 ;SET ACO OPERAND
 LDD (R0),ACO
 MOV #EEP1,R0 ;FSPC
 EE2: SUBD (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #EEDATO,R0 ;GET THE RESULT
 STD ACO,(R0)
 MOV #EEP0,R1 ;IS IT CORRECT?
 MOV #4,R2
 EE3: CMP (R0),.(R1).
 BEQ EE6
 EMT ;
 EE6: SOB R2,EE3
 BIS #4,R4
 CMP R4,R5 ;FPS CORRECT?
 BEQ EE7
 EMT ;
 ; USE NEGATIVE OPERANDS
 EE7: MOV #3200,R4 ;SET FIO, FIV, AND FD
 LDFPS R4
 MOV #EEP3,R0 ;SET ACO OPERAND
 LDD (R0),ACO
 MOV #EEP3,R0 ;FSPC
 EE8: SUBD (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #EEDATO,R0 ;GET THE RESULT

17492 070420 174010
17493 070422 012701 070464
17494 070426 012702 000004
17495 070432 022021
17496 070434 001401
(2) 070436 104000
17497 070440 077204
17498 070442 052704 000004
17499 070446 020405
17500 070450 001432
(2) 070452 104000
17501 070454 000000
17502 070456 000000
17503 070460 000000
17504 070462 000000
17505 070464 000000
17506 070466 000000
17507 070470 000000
17508 070472 000000
17509 070474 000000
17510 070476 000200
17511 070500 000000
17512 070502 000000
17513 070504 000000
17514 070506 000400
17515 070510 000000
17516 070512 000000
17517 070514 000000
17518 070516 100200
17519 070520 000000
17520 070522 000000
17521 070524 000000
17522 070526 100400
17523 070530 000000
17524 070532 000000
17525 070534 000000
17526 070536
(1) 070536 004767 034010
(1)
(1)
(1)
(1)
17536
(2)
(3)
(2) 070542
17537
17538 070542 012704 003200
17539 070546 170104
17540 070550 012700 070732
17541 070554 172410
17542 070556 012700 070742
17543 070562 172010
17544 070564 170205
17545 070566 012700 070702
17546 070572 174010

```
STD ACO,(R0)
MOV #EEP0,R1 ;IS IT CORRECT?
MOV #4,R2
EE9: CMP (R0),R1
BEQ EE12
EMT ;
EE12: SOB R2,EE9
BIS #4,R4
CMP R4,R5 ;FPS CORRECT?
BEQ EEDONE
EMT ;
EEDATO: 0
0
0
0
EEPO: 0
0
00000
0
0
EEP1: 200
0
0
0
EEP2: 400
0
0
0
EEP3: 100200
0
0
0
EEP4: 100400
0
0
0
EEDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 467 NORMALIZE ALGORITHM TEST
;*****
TS467:
;USE DATA PATTERNS THAT REQUIRE ONLY ONE LEFT SHIFT TO NORMALIZE
MOV #3200,R4 ;SET FIO, FIV, AND FD
LDFPS R4
MOV #FFP2,R0 ;SET ACO OP_LAND
LDD (R0),ACO
MOV #FFP3,R0 ;FSPC
FF2: ADD (R0),ACO ;TEST INSTRUCTION
STFPS R5 ;GET FPS
MOV #FFDATO,R0 ;GET THE RESULT
STD ACO,(R0)
```


CJKLS80 LCP-5 CPU CLSTR DIAG
CJKLSB.P11 07 JAN-85 09:05

J
MACY11 30(1046) 07-JAN-85 09:28 PAGE 24 19
T467 NORMALIZE ALGORITHM TEST

SEQ 0294

17547 070574 012701 070752

MOV @FFP4, R1

;IS IT CORRECT

17549 070600 012702 000004
 17550 070604 022021
 17551 070606 001401
 (2) 070610 104000
 17552 070612 077204
 17553 070614 020405
 17554 070616 001401
 (2) 070620 104000
 17555
 17556
 17557 070622 012704 003200
 17558 070626 170104
 17559 070630 012700 070712
 17560 070634 172410
 17561 070636 012700 070722
 17562 070642 172010
 17563 070644 170205
 17564 070646 012700 070702
 17565 070652 174010
 17566 070654 012701 070752
 17567 070660 012702 000004
 17568 070664 022021
 17569 070666 001401
 (2) 070670 104000
 17570 070672 077204
 17571 070674 020405
 17572 070676 001431
 (2) 070700 104000
 17573
 17574
 17575 070702 000000
 17576 070704 000000
 17577 070706 000000
 17578 070710 000000
 17579
 17580 070712 016000
 17581 070714 000000
 17582 070716 000000
 17583 070720 000001
 17584 070722 116000
 17585 070724 000000
 17586 070726 000000
 17587 070730 000000
 17588 070732 000500
 17589 070734 000000
 17590 070736 000000
 17591 070740 000000
 17592 070742 100400
 17593 070744 000000
 17594 070746 000000
 17595 070750 000000
 17596 070752 000200
 17597 070754 000000
 17598 070756 000000
 17599 070760 000000
 17600

```

MOV #4,R2
FF3:  CMP (R0)+,(R1)+
      BEQ FF4
      EMT ;
FF4:  SOB R2,FF3
      CMP R4,R5 ;FPS CORRECT?
      BEQ FF5
      EMT ;
;USE DATA PATTERNS WHICH REQUIRE 56 LEFT SHIFTS TO NORMALIZE
;THE RESULT
FF5:  MOV #3200,R4 ;SET FIU, FIV, AND FD
      LDFPS R4
      MOV #FFP0,R0 ;SET ACO OPERAND
      LDD (R0),ACO
      MOV #FFP1,R0 ;FSRC
FF6:  ADDD (R0),ACO ;TEST INSTRUCTION
      STFPS R5 ;GET FPS
      MOV #FFDAT0,R0 ;GET THE RESULT
      STD ACO,(R0)
      MOV #FFP4,R1 ;IS IT CORRECT
      MOV #4,R2
FF7:  CMP (R0)+,(R1)+
      BEQ FF10
      EMT ;
FF10: SOB R2,FF7
      CMP R4,R5 ;FPS CORRECT?
      BEQ FFDONE
      EMT ;

```

```

FFDAT0: 0
        0
        0
        0
FFP0:   16000
        0
        0
        1
FFP1:   116000
        0
        0
        0
FFP2:   500
        0
        0
        0
FFP3:   100400
        0
        0
        0
FFP4:   200 ;FFP4=FFP0+FFP1
          ; =FFP3+FFP4
        0
        0
        0

```

17601 070762
 (1) 070762 004767 033564
 (1)
 (1)
 (1)
 (1)
 17602
 17603
 17604
 17605
 17606
 17607
 17608
 17609
 17610
 17611
 17622
 (2)
 (3)
 (2) 070766
 17623
 17624
 17625
 17626 070766 012704 003200
 17627 070772 170104
 17628 070774 012700 071376
 17629 071000 172410
 17630 071002 012700 071406
 17631 071006 172010
 17632 071010 170205
 17633 071012 012700 071366
 17634 071016 174010
 17635 071020 012701 071416
 17636 071024 012702 000004
 17637 071030 022021
 17638 071032 001401
 (2) 071034 104000
 17639 071036 077204
 17640 071040 020405
 17641 071042 001401
 (2) 071044 104000
 17642
 17643
 17644
 17645
 17646
 17647 071046 012704 043200
 17648
 17649 071052 170104
 17650 071054 012700 071446
 17651 071060 172410
 17652 071062 012700 071456
 17653 071066 172010
 17654 071070 170205
 17655 071072 012700 071366
 17656 071076 174010

FFDONE: JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

;*****
 ;*****
 ;FLOATING POINT SECOND PART
 ;*****
 ;*****

;*****
 ;TEST 470 ROUND\TRUNK TEST
 ;*****
 TS470:

;ROUND AND NORMALIZE TEST
 MOV #3200,R4 ;SET FIU, FIV, AND FD
 LDFPS R4
 MOV #MHP0,R0 ;SET ACO OPERAND
 LDD (R0),ACO
 MOV #MHP1,R0 ;FSPC
 HM2: ADDD (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #MHDATO,R0 ;GET THE RESULT
 STD ACO,(R0)
 MOV #MHP2,R1 ;IS IT CORRECT
 MOV #4,R2
 HM3: CMP (R0),.(R1).
 BEQ HM6
 EMT ;
 HM6: SOB R2,HM3
 CMP R4,R5 ;FPS CORREC.?
 BEQ HM7
 EMT ;

;THIS IS A TEST OF THE ABILITY
 ;OF NORMALIZE TO PRODUCE A ZERO EXP. AND
 ;OF THE R\T ALGORITHM TO PROPERLY SET THE FPS

HM7: MOV #043200,R4 ;SET FIU,FIV,AND FD
 ;FID
 LDFPS R4
 MOV #MHP5,R0 ;SET ACO OPERAND
 LDD (R0),ACO
 MOV #MHP6,R0 ;FSPC
 HM8: ADDD (R0),ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV #MHDATO,R0 ;GET THE RESULT
 STD ACO,(R0)

17657	071100	012701	071436		MOV	#RMP4,R1		;IS IT CORRECT
17658	071104	012702	000004		MOV	#4,R2		
17659	071110	022021		HH9:	CMP	(R0)+,(R1)+		
17660	071112	001401			BEQ	HH10		
(2)	071114	104000			EMT			
17661	071116	077204		HH10:	SQB	R2,HH9		
17662	071120	052704	100004		BIS	#100004,R4		
17663	071124	020405			CMP	R4,R5		
17664	071126	001401			BEQ	HH11		
(2)	071130	104000			EMT			
17665								
17666								
17667								
17668	071132	012704	043200	HH11:	MOV	#043200,R4		;SET FIV, FIV, AND FD
17669								
17670	071136	170104			LDFPS	R4		
17671	071140	012700	071476		MOV	#RMP8,R0		;SET ACO OPERAND
17672	071144	172410			LDD	(R0),ACO		
17673	071146	012700	071506		MOV	#RMP9,R0		;FSPC
17674	071152	172010		HH12:	ADD	(R0),ACO		;TEST INSTRUCTION
17675	071154	170205			STFPS	R5		;GET FPS
17676	071156	012700	071366		MOV	#HDATA0,R0		;GET THE RESULT
17677	071162	174010			STD	ACO,(R0)		
17678	071164	012701	071466		MOV	#RMP7,R1		;IS IT CORRECT
17679	071170	012702	000004		MOV	#4,R2		
17680	071174	022021		HH13:	CMP	(R0)+,(R1)+		
17681	071176	001401			BEQ	HH16		
(2)	071200	104000			EMT			
17682	071202	077204		HH16:	SQB	R2,HH13		
17683	071204	052704	100014		BIS	#100014,R4		;FPS CORRECT?
17684	071210	020405			CMP	R4,R5		
17685	071212	001401			BEQ	HH17		
(2)	071214	104000			EMT			
17686								
17687	071216	012704	000200	HH17:	MOV	#00200,R4		;SET FIV, FIV, AND FD
17688	071222	170104			LDFPS	R4		
17689	071224	012700	071476		MOV	#RMP8,R0		;SET ACO OPERAND
17690	071230	172410			LDD	(R0),ACO		
17691	071232	012700	071476		MOV	#RMP8,R0		;FSPC
17692	071236	172010		HH18:	ADD	(R0),ACO		;TEST INSTRUCTION
17693	071240	170205			STFPS	R5		;GET FPS
17694	071242	012700	071366		MOV	#HDATA0,R0		;GET THE RESULT
17695	071246	174010			STD	ACO,(R0)		
17696	071250	012701	071516		MOV	#RMP10,R1		;IS IT CORRECT
17697	071254	012702	000004		MOV	#4,R2		
17698	071260	022021		HH19:	CMP	(R0)+,(R1)+		
17699	071262	001401			BEQ	HH20		
(2)	071264	104000			EMT			
17700	071266	077204		HH20:	SQB	R2,HH19		
17701	071270	052704	000000		BIS	#000000,R4		;FPS CORRECT?
17702	071274	020405			CMP	R4,R5		
17703	071276	001401			BEQ	HH21		
(2)	071300	104000			EMT			
17704								
17705	071302	012704	003200	HH21:	MOV	#3200,R4		;SET FIV, FIV, AND FD
17706	071306	170104			LDFPS	R4		

```

17707 071310 012700 071446      MOV      #HMP5,R0      ;SET ACO OPERAND
17708 071314 172410      LDD      (R0),ACO
17709 071316 012700 071446      MOV      #HMP5,R0      ;FSPC
17710 071322 172010      HM22:   ADDD     (R0),ACO ;TEST INSTRUCTION
17711 071324 170205      STFPS    R5           ;GET FPS
17712 071326 012700 071366      MOV      #HMDDATO,R0  ;GET THE RESULT
17713 071332 174010      STD      ACO,(R0)
17714 071334 012701 071526      MOV      #HMP11,R1    ;IS IT CORRECT
17715 071340 012702 000004      MOV      #4,R2
17716 071344 022021      HM23:   CMP      (R0),.(R1), ;
17717 071346 001401      BEQ      HM24
(2) 071350 104000      EMT
17718 071352 077204      HM24:   SOB      R2,HM23 ;
17719 071354 052704 000010      BIS      #10,R4
17720 071360 020405      CMP      R4,R5        ;FPS CORRECT?
17721 071362 001465      BEQ      HMDONE
(2) 071364 104000      EMT ;
17722 071366 000000      HMDDATO: 0
17723 071370 000000      0
17724 071372 000000      0
17725 071374 000000      0
17726 071376 000452      HMP0:   452
17727 071400 125252      125252
17728 071402 125252      125252
17729 071404 125253      125253
17730 071406 000252      HMP1:   252
17731 071410 125252      125252
17732 071412 125252      125252
17733 071414 125252      125252
17734 071416 000600      HMP2:   600          ;HMP0 + HMP1 WITH
17735 071420 000000      0              ;PROPER NORMALIZATION
17736 071422 000000      0
17737 071424 000000      0
17738 071426 000400      HMP3:   400          ;HMP0 + HMP1 WITH
17739 071430 000000      0              ;BAD NORMALIZATION
17740 071432 000000      0
17741 071434 000000      0
17742 071436 000000      HMP4:   0
17743 071440 000000      0
17744 071442 000000      0
17745 071444 000000      0
17746 071446 100200      HMP5:   100200
17747 071450 000000      0
17748 071452 000000      0
17749 071454 000000      0
17750 071456 000300      HMP6:   300
17751 071460 000000      0
17752 071462 000000      0
17753 071464 000000      0
17754 071466 100000      HMP7:   100000     ;HMP7 = HMP8 + HMP9
17755 071470 000000      0              ; = HMP5 + HMP6
17756 071472 000000      0
17757 071474 000000      0
17758 071476 000200      HMP8:   200
17759 071500 000000      0
17760 071502 000000      0

```

17761 071504 000000
 17762 071506 100300
 17763 071510 000000
 17764 071512 000000
 17765 071514 000000
 17766 071516 000400
 17767 071520 000000
 17768 071522 000000
 17769 071524 000000
 17770 071526 100400
 17771 071530 000000
 17772 071532 000000
 17773 071534 000000
 17774 071536
 (1) 071536 004767 033010
 (1)
 (1)
 (1)
 (1)
 17775
 17789
 17790
 (2)
 (3)
 (2) 071542
 17791
 17792
 17793 071542 012704 000200
 17794 071546 170104
 17795 071550 012737 071666 000244
 17796 071556 012700 072470
 17797 071562 172410
 17798 071564 012700 072470
 17799 071570 172010
 17800 071572 170205
 17801 071574 012700 072420
 17802 071600 174010
 17803 071602 012701 072500
 17804 071606 012702 000004
 17805 071612 022021
 17806 071614 001401
 (2) 071616 104000
 17807 071620 077204
 17808 071622 052704 000006
 17809 071626 020405
 17810 071630 001401
 (2) 071632 104000
 17811
 17812
 17813 071634 012704 001200
 17814 071640 170104
 17815 071642 012737 071670 000244
 17816 071650 012700 072470
 17817 071654 172410
 17818 071656 012700 072470
 17819 071662 172010

0
 HMP9: 100300
 0
 0
 0
 HMP10: 400 ;HMP10 = HMP8 * HMP8
 0
 0
 0
 HMP11: 100400 ;HMP11 = HMP5 * HMP5
 0
 0
 0
 HMDONE: JSR PC, RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

.....
 ;TEST 471 OVER\UNDER TEST
 ;.....
 TS471:
 ;TEST OVERFLOW CONDITION WITH TRAP DISABLER FIV=0
 MOV #200,R4 ;CLEAR FIU, FIV, AND SET FD
 LDFPS R4
 MOV @GGERO, @FFPVECT
 MOV @GGP5, R0 ;SET ACO OPERAND
 LDD (R0), ACO
 MOV @GGP5, R0 ;FSRC
 GG2: ADD (R0), ACO ;TEST INSTRUCTION
 STFPS R5 ;GET FPS
 MOV @GGDATO, R0 ;GET THE RESULT
 STD ACO, (R0)
 MOV @GGP6, R1 ;IS IT CORRECT
 GG3: CMP (R0), (R1)
 BEQ GG4
 EMT ;
 GG4: SOB R2, GG3
 BIS #6, R4 ;FPS CORRECT?
 CMP R4, R5
 BEQ GG5
 EMT ;
 ;TEST OVERFLOW WITH TRAPS ENABLED
 ;FIV = 1
 GG5: MOV #1200, R4 ;CLEAR FIU, SET FIV, AND FD
 LDFPS R4
 MOV @GG7, @FFPVECT
 MOV @GGP5, R0 ;SET ACO OPERAND
 LDD (R0), ACO
 MOV @GGP5, R0 ;FSPC
 GG6: ADD (R0), ACO ;TEST INSTRUCTION

```

17820 071664 170000          CFCC          ;NO OVERFLOW TRAP OCCURED
17821 071666          GGERO:
(2) 071666 104000          EMT          ;
17822 071670 012703 071664  GG7:  MOV      @GG6+2,R3
17823 071674 020316          CMP      R3,(SP)      ;CHECK STACK DATA
17824 071676 001401          BEQ      18
(2) 071700 104000          EMT          ;
17825 071702 022626          18:  CMP      (SP),.(SP).
17826 071704 170205          STFPS   R5
17827 071706 012700 072420  MOV      @GGDAT0,R0      ;GET THE RESULT
17828 071712 174010          STD     ACO,(R0)
17829 071714 012701 072500  MOV      @GCP6,R1      ;IS IT CORRECT
17830 071720 012702 000004  MOV      #4,R2
17831 071724 022021  GG8:  CMP      (R0),.(R1).
17832 071726 001401          BEQ      GG9
(2) 071730 104000          EMT          ;
17833 071732 077204          GG9:  SOB     R2,GG8
17834 071734 052704 100006  BIS     #100006,R4      ;EXACT ZERO RESULTED IF OVERFLOW
17835 071740 020405          CMP     R4,R5          ;FPS CORRECT?, CHECK FER, FZ, FV
17836 071742 001401          BEQ     18
(2) 071744 104000          EMT          ;
17837 071746 012704 000010  18:  MOV     #10,R4
17838          ;CHECK FEC
17839 071752 170305          STST   R5
17840 071754 020405          CMP    R4,R5
17841 071756 001401          BEQ    GG10
(2) 071760 104000          EMT          ;
17842          ;CHECK UNDER FLOW CONDITION WITH
17843          ;TRAPS DISABLED (FIU = 0)
17844 071762 012704 000200  GG10: MOV     #0200,R4      ;SET FIU, FIV, AND FD
17845 071766 170104          LDFPS  R4
17846 071770 012737 071666 000244  MOV     @GGERO,@FFPVECT
17847 071776 012700 072440  MOV     @GGP2,R0      ;SET ACO OPERAND
17848 072002 172410          LDD    (R0),ACO      ;FSAC
17849 072004 012700 072450  MOV     @GGP3,R0
17850 072010 172010          GG11: ADD    (R0),ACO      ;TEST INSTRUCTION
17851 072012 170205          STFPS  R5          ;GET FPS
17852 072014 012700 072420  MOV     @GGDAT0,R0      ;GET THE RESULT
17853 072020 174010          STD    ACO,(R0)
17854 072022 012701 072500  MOV     @GGP6,R1      ;IS IT CORRECT
17855 072026 012702 000004  MOV     #4,R2
17856 072032 022021  GG12: CMP     (R0),.(R1).
17857 072034 001401          BEQ    GG13
(2) 072036 104000          EMT          ;
17858 072040 077204          GG13: SOB     R2,GG12
17859 072042 052704 000004  BIS     #4,R4          ;FPS CORRECT?
17860 072046 020405          CMP    R4,R5
17861 072050 001401          BEQ    GG14
(2) 072052 104000          EMT          ;
17862          ;CHECK UNDERFLOW CONDITION WITH
17863          ;TRAP ENABLED (FIU = 1)
17864 072054 012704 002200  GG14: MOV     #2200,R4      ;SET FIU, FIV, AND FD
17865 072060 170104          LDFPS  R4
17866 072062 012737 072106 000244  MOV     @GG16,@FFPVECT
17867 072070 012700 072440  MOV     @GGP2,R0      ;SET ACO OPERAND
17868 072074 172410          LDD    (R0),ACO      ;FSPC

```

17869	072076	012700	072450			MOV	@GGP3,R0	
17870	072102	172010			GG15:	ADD	(R0),ACO	;TEST INSTRUCTION
17871	072104	170000				CFCC		
17872	072106	012703	072104		GG16:	MOV	@GG15+2,R3	
17873	072112	021603				CMP	(SP),R3	
17874	072114	001401				BEQ	14	
(2)	072116	104000				EMT		
17875	072120	022626			14:	CMP	(SP)..(SP).	
17876	072122	170205				STFPS	R5	;GET FPS
17877	072124	012700	072420			MOV	@GGDAT0,R0	;GET THE RESULT
17878	072130	174010				STD	ACO,(R0)	
17879	072132	012701	072510			MOV	@GGP7,R1	;IS IT CORRECT
17880	072136	012702	000004			MOV	#4,R2	
17881	072142	022021			GG17:	CMP	(R0)..(R1).	
17882	072144	001401				BEQ	GG18	
(2)	072146	104000				EMT		
17883	072150	077204			GG18:	SQB	R2,GG17	
17884	072152	052704	100000			BIS	#100000,R4	
17885	072156	020405				CMP	R4,R5	;FPS CORRECT?
17886	072160	001401				BEQ	14	
(2)	072162	104000				EMT		
17887	072164	012704	000012		14:	MOV	#12,R4	
17888						;CHECK FEC		
17889	072170	170305				STST	R5	
17890	072172	020405				CMP	R4,R5	
17891	072174	001401				BEQ	GG19	
(2)	072176	104000				EMT		
17892						;CHECK UNDERFLOW CONDITION WITH TRAPS		
17893						;DISABLED (FIU = 0)		
17894	072200	012704	000200		GG19:	MOV	#0200,R4	;SET FIU, FIV, AND FD
17895	072204	170104				LDFPS	R4	
17896	072206	012737	072324	000244		MOV	@GGER14,B#FPVECT	
17897	072214	012700	072440			MOV	@GGP2,R0	;SET ACO OPERAND
17898	072220	172410				LDD	(R0),ACO	
17899	072222	012700	072520		GG20:	MOV	@GGP8,R0	;FSPC
17900	072226	172010				ADD	(R0),ACO	;TEST INSTRUCTION
17901	072230	170205				STFPS	R5	;GET FPS
17902	072232	012700	072420			MOV	@GGDAT0,R0	;GET THE RESULT
17903	072236	174010				STD	ACO,(R0)	
17904	072240	012701	072500			MOV	@GGP6,R1	;IS IT CORRECT
17905	072244	012702	000004			MOV	#4,R2	
17906	072250	022021			GG21:	CMP	(R0)..(R1).	
17907	072252	001401				BEQ	GG22	
(2)	072254	104000				EMT		
17908	072256	077204			GG22:	SQB	R2,GG21	
17909	072260	052704	000004			BIS	#4,R4	;FPS CORRECT?
17910	072264	020405				CMP	R4,R5	
17911	072266	001401				BEQ	GG23	
(2)	072270	104000				EMT		
17912						;CHECK UNDERFLOW CONDITION WITH TRAP		
17913						;ENABLED (FIU = 1)		
17914	072272	012704	002200		GG23:	MOV	#2200,R4	;SET FIU, FIV, AND FD
17915	072276	170104				LDFPS	R4	
17916	072300	012737	072326	000244		MOV	@GG25,B#FPVECT	
17917	072306	012700	072440			MOV	@GGP2,R0	;SET ACO OPERAND
17918	072312	172410				LDD	(R0),ACO	

17919	072314	012700	072520		MOV	@GGP8,R0		;FSRC
17920	072320	172010		GG24:	ADDD	(R0),ACO		;TEST INSTRUCTION
17921	072322	170000			CFCC			
17922	072324			GGER14:	EMT			
(2)	072324	104000						
17923	072326	012703	072322	GG25:	MOV	@GG24+2,R3		
17924	072332	020316			CMF	R3,(SP)		
17925	072334	001401			BEQ	18		
(2)	072336	104000			EMT			
17926	072340	022626		18:	CMF	(SP)+,(SP)+		
17927	072342	170205			STFPS	R5		;GET FPS
17928	072344	012700	072420		MOV	@GGDAT0,R0		;GET THE RESULT
17929	072350	174010			STD	ACO,(R0)		
17930	072352	012701	072530		MOV	@GGP9,R1		;IS IT CORRECT
17931	072356	012702	000004		MOV	#4,R2		
17932	072362	022021		GG26:	CMF	(R0)+,(R1)+		
17933	072364	001401			BEQ	GG27		
(2)	072366	104000			EMT			
17934	072370	077204		GG27:	SQB	R2,GG26		
17935	072372	052704	100004		BIS	#100004,R4		
17936	072376	020405			CMF	R4,R5		;FPS CORRECT?
17937	072400	001401			BEQ	18		
(2)	072402	104000			EMT			
17938	072404	012704	000012	18:	MOV	#12,R4		
17939					;CHECK FEC			
17940	072410	170305			STST	R5		
17941	072412	020405			CMF	R4,R5		
17942	072414	001451			BEQ	GGDONE		
(2)	072416	104000			EMT			
17943	072420	000000		GGDAT0:	0			
17944	072422	000000			0			
17945	072424	000000			0			
17946	072426	000000			0			
17947								
17948	072430	000300		GGP1:	300			
17949	072432	000000			0			
17950	072434	000000			0			
17951	072436	000000			0			
17952	072440	100200		GGP2:	100200			
17953	072442	000000			0			
17954	072444	000000			0			
17955	072446	000000			0			
17956	072450	000200		GGP3:	200			
17957	072452	000000			0			
17958	072454	000000			0			
17959	072456	000001			1			
17960	072460	010200		GGP4:	10200			
17961	072462	000000			C			
17962	072464	000000			0			
17963	072466	000000			0			
17964	072470	077600		GGP5:	77600			;OVER FLOW = GGP5 * GGP5
17965	072472	000000			0			
17966	072474	000000			0			
17967	072476	000000			0			
17968	072500	000000		GGP6:	0			;OVERFLOW RESULT
17969	072502	000000			0			;UNDERFLOW RESULT

17970 072504 000000
17971 072506 000000
17972
17973 072510 062400
17974 072512 000000
17975 072514 000000
17976 072516 000000
17977 072520 000340
17978 072522 000000
17979 072524 000000
17980 072526 000000
17981 072530 000100
17982 072532 000000
17983 072534 000000
17984 072536 000000
17985 072540
(1) 072540 004767 032006
(1)
(1)
(1)
(1)
17986
17992
17993
(2)
(3)
(2) 072544
17994
17995 072544 012704 000200
17996 072550 170104
17997 072552 012700 073302
17998 072556 172410
17999 072560 012700 073312
18000 072564 177420
18001 072566 020027 073316
18002 072572 001401
(2) 072574 104000
18003 072576
18004 072576 170205
18005 072600 012700 073272
18006 072604 174010
18007 072606 012701 073362
18008 072612 012702 000004
18009 072616 022120
18010 072620 001401
(2) 072622 104000
18011 072624 077204
18012 072626 012704 000200
18013 072632 020405
18014 072634 001401
(2) 072636 104000
18015
18016 072640 012704 000200
18017 072644 170104
18018
18019 072646 012700 073302

0
0
GGP7: 62400
0
0
GGP8: 340
0
0
GGP9: 100
0
0
GGDONE:
JSR PC,.RSET
;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 472 LDCFD AND LDCDF TEST
;*****
TS472:
;TEST FOR CORRECT AUTO INCREMENT CONSTANT.
MOV #200,R4 ;SET LONG INTEGER MODE
LDFPS R4
MOV #HXP1,R0
LDD (R0),ACO
MOV #HXP2,R0
HX2: LDCFD (R0)+,ACO
CMP R0,#HXP2+4 ;IS R0 CORRECT
BEQ HX3
EMT ;
HX3: STFPS R5 ;GET FPS
MOV #HXDAT0,R0
STD ACO,(R0) ;GET ACO
MOV #HXP7,R1 ;SEE IF RESULT IS
MOV #4,R2 ;CORRECT
HX4: CMP (R1)+,(R0)+
BEQ HX7
EMT ;
HX7: SOB R2,HX4
MOV #200,R4 ;FPS CORRECT?
CMP R4,R5
BEQ HX8
EMT ;
;NOW TEST LDCDF
HX8: MOV #200,R4
LDFPS R4
MOV #HXP1,R0

18020	072652	172410			LDD	(R0),ACO	
18021							
18022	072654	012700	073312		MOV	#HXP2,R0	
18023	072660	170001			SETF		
18024	072662	177420		HX9:	LDCDF	(R0)+,ACO	;TEST INSTRUCTION
18025	072664	020027	073322		CHP	R0,#HXP2+10	;WAS A GOOD
18026	072670	001401			BEQ	HX10	
(2)	072672	104000			EMT		;
18027							
18028	072674			HX10:			
18029	072674	170205			STFPS	R5	
18030	072676	012700	073272		MOV	#HXDAT0,R0	
18031	072702	170011			SETD		
18032	072704	174010			STD	ACO,(R0)	;GET RESULT
18033	072706	012701	073372		MOV	#HXP8,R1	
18034	072712	012702	000004		MOV	#4,R2	
18035	072716	022120		HX11:	CHP	(R1)+,(R0)+	;IS IT CORRECT?
18036	072720	001401			BEQ	HX14	
(2)	072722	104000			EMT		;
18037	072724	077204		HX14:	SOB	R2,HX11	
18038							
18039	072726	012704	000000		MOV	#0,R4	;FPS CORRECT?
18040	072732	020405			CHP	R4,R5	
18041	072734	001401			BEQ	HX15	
(2)	072736	104000			EMT		;
18042							;TEST GR7 IMMEDIATE MODE CONSTANT
18043	072740	012704	000200	HX15:	MOV	#200,R4	
18044	072744	170104			LDFPS	R4	;SET FD
18045	072746	012737	072776	000004	MOV	#HXER9,#ERRVECT	
18046	072754	005001			CLR	R1	
18047	072756	177427	043243	HX16:	LDCFD	#5201,ACO	
18048	072762	005201		HX165:	INC	R1	
18049	072764	005201			INC	R1	
18050	072766	005201			INC	R1	
18051	072770	020127	000003		CHP	R1,#3	;SEE IF PC WAS
18052	072774	001401			BEQ	HX17	
(1)	072776			HXER9:			
(2)	072776	104000			EMT		;
18053	073000	012704	000200	HX17:	MOV	#200,R4	
18054	073004	170104			LDFPS	R4	
18055	073006	012700	073352		MOV	#HXP6,R0	
18056	073012	172410			LDD	(R0),ACO	
18057	073014	012700	073312		MOV	#HXP2,R0	
18058	073020	177410		HX18:	LDCFD	(R0),ACO	
18059							
18060	073022	012700	073272		MOV	#HXDAT0,R0	
18061	073026	174010			STD	ACO,(R0)	;GET RESULT.
18062	073030	012701	073362		MOV	#HXP7,R1	
18063	073034	012702	000004		MOV	#4,R2	
18064	073040	022021		HX19:	CHP	(R0)+,(R1)+	;IS RESULT CORRECT?
18065	073042	001401			BEQ	HX20	
(2)	073044	104000			EMT		;
18066	073046	077204		HX20:	SOB	R2,HX19	
18067							
18068							;TEST LDCFD WITH NEGATIVE OPERAND
18069	073050	012704	000200		MOV	#200,R4	

18070	073054	170104		LDFPS	R4	
18071	073056	012700	073352	MOV	#HXP6,R0	
18072	073062	172410		LDD	(R0),ACO	
18073	073064	012700	073332	MOV	#HXP4,R0	
18074	073070	177410		HX22: LDCFD	(R0),ACO	
18075						
18076	073072	012700	073272	MOV	#HXDAT0,R0	
18077	073076	174010		STD	ACO,(R0)	;GET RESULT
18078						
18079	073100	012701	073342	MOV	#HXP5,R1	
18080	073104	012702	000004	MOV	#4,R2	
18081	073110	022120		HX23: CMP	(R1),,(R0),	
18082	073112	001401		BEQ	HX26	
(2)	073114	104000		EMT		;
18083	073116	077204		HX26: SOB	R2,HX23	
18084						
18085				;TEST LDCFD	0	
18086						
18087	073120	012704	000200	MOV	#200,R4	
18088	073124	170104		LDFPS	R4	
18089						
18090	073126	012700	073302	MOV	#HXP1,R0	
18091	073132	172410		LDD	(R0),ACO	
18092	073134	172010		ADD	(R0),ACO	
18093						
18094	073136	012700	073302	MOV	#HXP1,R0	
18095	073142	177410		HX28: LDCFD	(R0),ACO	
18096						
18097	073144	170205		STFPS	R5	
18098						
18099	073146	012700	073272	MOV	#HXDAT0,R0	
18100	073152	174010		STD	ACO,(R0)	;GET RESULT
18101						
18102	073154	012701	073302	MOV	#HXP1,R1	
18103	073160	012702	000004	MOV	#4,R2	
18104	073164	022120		HX29: CMP	(R1),,(R0),	;IS IT 0?
18105	073166	001401		BEQ	HX30	
(2)	073170	104000		EMT		;
18106	073172	077204		HX30: SOB	R2,HX29	
18107						
18108	073174	012704	000204	MOV	#204,R4	;FPS CORRECT
18109	073200	020405		CMP	R4,R5	
18110	073202	001401		BEQ	HX31	
(2)	073204	104000		EMT		;
18111				;TEST LDCFD	0	
18112	073206	012704	000200	HX31: MOV	#200,R4	
18113	073212	170104		LDFPS	R4	
18114	073214	012700	073352	MOV	#HXP6,R0	
18115	073220	172410		LDD	(R0),ACO	
18116	073222	012700	073302	MOV	#HXP1,R0	
18117	073226	177410		HX32: LDCFD	(R0),ACO	
18118	073230	170205		STFPS	R5	
18119	073232	012700	073272	MOV	#HXDAT0,R0	
18120	073236	174010		STD	ACO,(R0)	;GET RESULT
18121	073240	012701	073302	MOV	#HXP1,R1	
18122	073244	012702	000004	MOV	#4,R2	

```

18123 073250 022120
18124 073252 001401
(2) 073254 104000
18125 073256 077204
18126
18127 073260 012704 000204
18128 073264 020405
18129 073266 001445
(2) 073270 104000
18130
18131 073272 000000
18132 073274 000000
18133 073276 000000
18134 073300 000000
18135
18136 073302 000000
18137 073304 000000
18138 073306 000000
18139 073310 000000
18140
18141 073312 000577
18142 073314 177776
18143 073316 177777
18144 073320 177776
18145 073322 005201
18146 073324 000000
18147 073326 000000
18148 073330 000000
18149 073332 100577
18150 073334 177776
18151 073336 177777
18152 073340 177776
18153 073342 100577
18154 073344 177776
18155 073346 000000
18156 073350 000000
18157 073352 000252
18158 073354 125252
18159 073356 125252
18160 073360 125252
18161
18162 073362 000577
18163 073364 177776
18164 073366 000000
18165 073370 000000
18166 073372 000577
18167 073374 177777
18168 073376 000000
18169 073400 000000
18170
18171 073402
(1) 073402 004767 031144
(1)
(1)
(1)
(1)

```

```

HX33:  CMP      (R1), (R0)      ; IS IT ZERO?
      BEQ      HX34
      EMT
HX34:  SOB      R2, HX33
      MOV      @204, R4
      CMP      R4, R5
      BEQ      HXDONE
      EMT
HXDAT0: 0
      0
      0
      0
HXP1:   0
      0
      0
      0
HXP2:   577
      177776
      177777
      177776
HXP3:   5201
      0
      0
      0
HXP4:   100577
      177776
      177777
      177776
HXP5:   100577
      177776
      0
      0
HXP6:   252
      125252
      125252
      125252
HXP7:   577
      177776
      0
      0
HXP8:   577
      177777
      0
      0
HXDONE:
      JSR      PC, .RSET
      ;GO INITIALIZE THE FPS AND STACK; AND
      ;SEE IF THE USER HAS EXPRESSED
      ;THE DESIRE TO CHANGE THE SOFTWARE
      ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      ;THE USER TYPED CONTROL G?).

```

```

18172
18173
18174
18182
18183
(2)
(3)
(2) 073406
18184
18185
18186 073406 004737 074102
18187 073412 000000 000000 000000
073420 000000
18188 073422 000000 000000 000000
073430 000000
18189 073432 000200
18190 073434 000204
18191
18192
18193
18194 073436 004737 074102
18195 073442 000000 000000 000000
073450 000000
18196 073452 025252
18197 073454 052525
18198 073456 125252
18199 073460 052525
18200 073462 000200
18201 073464 000200
18202
18203
18204 073466 004737 074102
18205 073472 000000 000000 000000
073500 000000
18206 073502 125252
18207 073504 125252
18208 073506 052525
18209 073510 125252
18210 073512 000200
18211 073514 000210
18212
18213
18214 073516 004737 074102
18215 073522 025252
18216 073524 052525
18217 073526 125252
18218 073530 052525
18219 073532 000000 000000 000000
073540 000000
18220 073542 000200
18221 073544 000210
18222
18223
18224
18225 073546 004737 074102
18226 073552 125252

```

```

;*****
;TEST 473      CMPD TEST
;*****
TS473:

```

```

;TEST THE CMPD INSTRUCTION WITH (FSRC=AC=0)
AAA1:  JSR      PC,B@CMPSUB
14:    .WORD    0,0,0,0          ;AC0
24:    .WORD    0,0,0,0          ;FSRC
34:    200
      204          ;FPS BEFORE EXECUTION
          ;FPS AFTER EXECUTION

```

```

;TEST CMPD WITH (AC=0) AND FSRC POSITIVE.
AAA2:  JSR      PC,B@CMPSUB
14:    .WORD    0,0,0,0          ;AC
24:    25252
      52525
      125252
      52525
34:    200
      200          ;FPS BEFORE EXECUTION
          ;FPS AFTER EXECUTION

```

```

;TEST CMPD WITH (AC=0) AND FSRC NEGATIVE
AAA3:  JSR      PC,B@CMPSUB
14:    .WORD    0,0,0,0          ;AC
24:    125252
      125252
      52525
      125252
34:    200
      210          ;FPS BEFORE EXECUTION
          ;FPS AFTER EXECUTION

```

```

;TEST CMPD WITH (FSRC=1) AND AC POSITIVE
AAA4:  JSR      PC,B@CMPSUB
14:    25252
      52525
      125252
      52525
24:    .WORD    0,0,0,0          ;FSRC
34:    200
      210          ;FPS BEFORE EXECUTION
          ;FPS AFTER EXECUTION

```

```

;TEST CMPD WITH (FSRC=0) AND AC NEGATIVE
AAA5:  JSR      PC,B@CMPSUB
14:    125252          ;AC

```

18227	073554	125252				125252		
18228	073556	052525				52525		
18229	073560	125252				125252		
18230	073562	000000	000000	000000	24:	.WORD 0.0.0.0		;FSRC
	073570	000000						
18231	073572	000200			34:	200		;FPS BEFORE EXECUTION
18232	073574	000200				200		;FPS AFTER EXECUTION
18233								
18234								
18235	073576	004737	074102					
18236	073602	052525						
18237	073604	125252						
18238	073606	052525						
18239	073610	125252						
18240	073612	125252			24:	125252		;FSRC
18241	073614	052525				52525		
18242	073616	125252				125252		
18243	073620	052525				52525		
18244	073622	000200			34:	200		;FPS BEFORE EXECUTION
18245	073624	000210				210		;FPS AFTER EXECUTION
18246								
18247								
18248								
18249	073626	004737	074102					
18250	073632	125252						
18251	073634	052525						
18252	073636	125252						
18253	073640	052525						
18254	073642	052525			24:	52525		;FSRC
18255	073644	125252				125252		
18256	073646	052525				52525		
18257	073650	125252				125252		
18258	073652	000200			34:	200		;FPS BEFORE EXECUTION
18259	073654	000200				200		;FPS AFTER EXECUTION
18260								
18261								
18262								
18263	073656	004737	074102					
18264	073662	012345						
18265	073664	067654						
18266	073666	032101						
18267	073670	023456						
18268	073672	023456			24:	23456		;FSRC
18269	073674	076543				76543		
18270	073676	021012				21012		
18271	073700	034567				34567		
18272	073702	000200			34:	200		;FPS BEFORE EXECUTION
18273	073704	000200				200		;FPS AFTER EXECUTION
18274								
18275								
18276								
18277	073706	004737	074102					
18278	073712	045676						
18279	073714	054321						
18280	073716	012345						
18281	073720	067654						

```

18282 073722 034567          24: 34567          ;FSRC
18283 073724 065432          65432
18284 073726 101234          101234
18285 073730 056765          56765
18286 073732 000200          34: 200          ;FPS BEFORE EXECUTION
18287 073734 000210          210          ;FPS AFTER EXECUTION
18288
18289          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE AND AC EQUAL TO FSRC
18290 073736 004737 074102 AAA10: JSR PC,@CMPSUB
18291 073742 012345          14: 12345          ;AC
18292 073744 067012          67012
18293 073746 034567          34567
18294 073750 012345          012345
18295 073752 012345          24: 12345          ;FSRC
18296 073754 067012          67012
18297 073756 034567          34567
18298 073760 012345          012345
18299 073762 000200          34: 200          ;FPS BEFORE EXECUTION
18300 073764 000204          204          ;FPS AFTER EXECUTION
18301
18302          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
18303          ;AND FSRC GREATER THAN AC.
18304 073766 004737 074102 AAA11: JSR PC,@CMPSUB
18305 073772 012345          14: 12345          ;AC
18306 073774 067012          67012
18307 073776 034567          34567
18308 074000 012345          012345
18309 074002 012345          24: 12345          ;FSRC
18310 074004 070123          70123
18311 074006 045670          45670
18312 074010 123456          123456
18313 074012 000200          34: 200          ;FPS BEFORE EXECUTION
18314 074014 000200          200          ;FPS AFTER EXECUTION
18315
18316          ;TEST CMPD WITH AC POSITIVE, FSRC POSITIVE, EAC EQUAL TO EFSRC,
18317          ;AND AC GREATER THAN FSRC.
18318 074016 004737 074102 AAA12: JSR PC,@CMPSUB
18319 074022 054321          14: 54321          ;AC
18320 074024 076543          76543
18321 074026 021076          21076
18322 074030 054321          54321
18323 074032 054321          24: 54321          ;FSRC
18324 074034 065432          65432
18325 074036 107654          107654
18326 074040 032107          32107
18327 074042 000200          34: 200          ;FPS BEFORE EXECUTION
18328 074044 000210          210          ;FPS AFTER EXECUTION
18329
18330          ;TEST CMPD WITH AC NEGATIVE, FSRC NEGATIVE, EAC EQUAL TO EFSRC,
18331          ;AND AC GREATER THAN FSRC
18332 074046 004737 074102 AAA13: JSR PC,@CMPSUB
18333 074052 112345          14: 112345          ;AC
18334 074054 043210          43210
18335 074056 076543          76543
18336 074060 021076          21076
18337 074062 112345          24: 112345          ;FSRC

```



```

18338 074064 054321          54321
18339 074066 007654          07654
18340 074070 032107          32107
18341 074072 000200          34: 200          ;FPS BEFORE EXECUTION
18342 074074 000210          210          ;FPS AFTER EXECUTION
18343
18344
18345 074076 000137 074206          JMP      @#AAADONE          ;FINISHED CMPD TEST.
18346
18347
18348          ;THIS SUBROUTINE, CMPSUB, IS CALLED TO SET UP, EXECUTE
18349          ;AND CHECK THE RESULTS OF A CMPD INSTRUCTION.
18350          ;IT IS CALLED THUS:
18351          ;
18352          ;          JSR      PC,@#CMPSUB
18353          ;          ACARG:  .WORD  X,X,X,X          ;AC OPERAND
18354          ;          FSRCARG: .WORD  X,X,X,X          ;FSRC OPERAND
18355          ;          FPSB:   .WORD  X          ;FPS BEFORE EXECUTION
18356          ;          FPSA:   .WORD  X          ;FPS AFTER EXECUTION
18357          ;          FPSE:   .WORD  X          ;ERROR FPS
18358          ;          ERR:    ERROR X          ;FPS ERROR
18359          ;          CONT:   ;RETURN ADDRESS
18360          ;
18361          ;THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
18362          ;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, CMPD, IS EXECUTED.
18363          ;AFTER THE EXECUTION THE FPS IS CHECKED AGAINST FPSA. IF IT IS A MATCH
18364          ;THEN THERE WAS NO ERROR AND CONTROL IS RETURNED TO CONT. IF
18365          ;THE FPS IS INCORRECT IT IS COMPARED WITH FPSE IN AN ATTEMPT TO ANALYSE
18366          ;THE FAILURE. IF THE FPS IS THE SAME AS FPSE THEN CONTROL IS
18367          ;RETURNED TO THE ERROR CALL AT LOCATION ERR. IF THE FPS WAS
18368          ;NOT CORRECT BUT DIDN'T MATCH FPSE A GENERAL ERROR IS REPORTED
18369          ;AND CONTROL IS PASSED TO CONT.
18370
18371 074102 012601          CMPSUB: MOV      (SP)+,R1          ;PICK UP A POINTER TO THE
18372          ;ARGUMENTS.
18373 074104 016100 000020          MOV      20(R1),R0          ;GET THE FPS BEFORE EXECUTION.
18374 074110 170100          LDFPS   R0          ;LOAD IT INTO THE FPS.
18375
18376 074112 010100          MOV      R1,R0          ;GET ADDRESS OF AC OPERAND.
18377 074114 172410          LDD     (R0),ACO          ;LOAD ACO OPERAND
18378
18379 074116 010100          MOV      R1,R0          ;COMPUTE FSRC OPERAND
18380 074120 062700 000010          ADD     #10,R0          ;ADDRESS
18381
18382 074124 000240          NOP          ;FOR SCOPING.
18383 074126 173410          14:    CMPD    (R0),ACO          ;EXECUTE THE TEST INSTRUCTION.
18384
18385 074130 170205          STFPS   R5          ;SAVE FPS AFTER INSTRUCTION.
18386
18387 074132 016104 000022          MOV      22(R1),R4          ;GET EXPECTED FPS.
18388 074136 020405          CMP     R4,R5          ;WAS FPS CORRECT?
18389 074140 020405          CMP     R4,R5          ;WAS FPS CORRECT?
18390 074142 001401          BEQ     34
18391 (2) 074144 104000          EMT
18391 074146 012700 074176          34:    MOV     @CMPTMP,R0          ;IF FPS WAS CORRECT MAKE SURE
18392 074152 174010          STD     ACO,(R0)          ;ACO WAS NOT AFFECTED BY CMPD.

```

```

18393 074154 010102          MOV      R1,R2
18394 074156 012703 000004    MOV      #4,R3
18395 074162 022220          4$:     CMP      (R2)*,(R0)*
18396 074164 001401          BEQ     5$
      (2) 074166 104000          EMT
18397 074170 077304          5$:     SOB     R3,4$
18398
18399 074172 000161 000024    JMP     24(R1)          ;RETURN
18400
18401 074176 000000 000000 000000  CMPTMP: .WORD 0,0,0,0
      074204 000000
18402
18403
18404
18405 074206          AAADONE:
      (1) 074206 004767 030340    JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                                     ;SEE IF THE USER HAS EXPRESSED
      (1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                                     ;THE USER TYPED CONTROL G?).
18406
18407
18408
18416
18417
      (2)
      (3)
      (2) 074212
18418
18419 074212 012704 040200    BBB0:   MOV     #40200,R4      ;SET UP FPS
18420                                     ;WITH INTERRUPTS
18421                                     ;DISABLED.
18422 074216 170104          LDFPS  R4
18423 074220 012737 074256 000244    MOV     @BBBER1,@FPVECT ;SET UP FOR ANY FP INTERRUPTS.
18424 074226 012700 074442    MOV     @BBBP1,R0          ;SET UP ACO = 0
18425 074232 172410          LDD    (R0),ACO
18426 074234 012701 074442    MOV     @BBBP1,R1          ;FSRC = 0
18427
18428 074240 174411          BBB1:   DIVD   (R1),ACO      ;TEST INSTRUCTION
18429
18430 074242 170205          STFPS  R5          ;GET FPS
18431 074244 170303          STST   R3          ;GET FEC
18432
18433 074246 012704 140204          MOV     #140204,R4        ;EXPECTED FPS.
18434 074252 020405          CMP    R4,R5          ;IS FPS CORRECT.
18435 074254 001401          BEQ    BBB7
      (1) 074256
18436 074256 104000          BBBER1: EMT
      (2) 074256 012702 000004    BBB7:   MOV     #4,R2          ;EXPECTED FEC.
18437 074264 020203          CMP    R2,R3          ;IS FEC CORRECT?
18438 074266 001401          BEQ    BBB2
      (2) 074270 104000          EMT
18439
18440
18441 074272 012704 040200          ;TEST DIVD WITH (FSRC=0) AND TRAPS DISABLED.
18442 074276 170104          BBB2:   MOV     #40200,R4        ;LOAD FPS WITH TRAPS DISABLED.
      LDFPS  R4

```

```

18443
18444 074300 012700 074452      MOV      @BBBP2,R0      ;SET UP ACO OPERAND (NON ZERO).
18445 074304 172410              LDD      (R0),ACO
18446 074306 012700 074442      MOV      @BBBP1,R0      ;FSRC=0
18447 074312 174410      BBB3:   DIVD      (R0),ACO
18448
18449 074314 170205              STFPS   R5              ;GET FPS.
18450 074316 170303              STST   R3              ;GET FEC.
18451
18452 074320 012704 140200      MOV      @140200,R4      ;EXPECTED FPS.
18453 074324 020405              CMP     R4,R5           ;IS FPS CORRECT?
18454 074326 001401              BEQ     11
(2) 074330 104000              EMT
18455 074332 012702 000004      11:    MOV      @4,R2       ;EXPECTED FEC.
18456 074336 020203              CMP     R2,R3           ;WAS FEC CORRECT?
18457 074340 001401              BEQ     BBB4
(2) 074342 104000              EMT
18458
18459              ;TEST DIVD WITH FSRC=0) AND TRAPS ENABLED.
18460 074344 012704 000200      BBB4:   MOV      @200,R4      ;SET UP FPS. TRAP ENABLED.
18461 074350 170104              LDFPS  R4
18462
18463 074352 012700 074452      MOV      @BBBP2,R0      ;SET UP ACO OPERAND (NON ZERO).
18464 074356 172410              LDD      (R0),ACO
18465
18466 074360 012737 074400 000244      MOV      @BBBP6,@FFPVECT ;SET UP FOR THE EXPECTED INTERRUPT.
18467 074366 012700 074442      MOV      @BBBP1,R0      ;FSRC=0
18468
18469 074372 174410      BBB5:   DIVD      (R0),ACO      ;TEST INSTRUCTION (SHOULD RESULT IN TRAP).
18470 074374 170000              CFCC
18471 074376 104000              EMT
18472 074400 022716 074374      BBB6:   CMP      @BBBP5+2,(SP) ;TRAP TO HERE WHEN THE DIVISION BY 0
18473              ;OCCURS. FIRST SEE IF THE ADDRESS OF
18474              ;THE TRAP IS 2*THE ADDRESS OF THE TEST
18475              ;DIVD INSTRUCTION.
18476 074404 001401              BEQ     11
(2) 074406 104000              EMT
18477 074410 170205      11:    STFPS   R5              ;GET FPS.
18478 074412 170303              STST   R3              ;GET FEC.
18479 074414 022626              CMP     (SP), (SP)      ;RESET THE STACK.
18480
18481 074416 012704 100200      MOV      @100200,R4      ;EXPECTED FPS.
18482 074422 020405              CMP     R4,R5           ;IS FPS CORRECT?
18483 074424 001401              BEQ     21
(2) 074426 104000              EMT
18484 074430 012702 000004      21:    MOV      @4,R2       ;EXPECTED FEC.
18485 074434 020203              CMP     R2,R3           ;IS FEC CORRECT?
18486 074436 001411              BEQ     BBBDONE
(2) 074440 104000              EMT
18487
18488 074442 000000 000000 000000 BBBP1:  .WORD  0,0,0,0
074450 000000
18489 074452 012345 054321 023456 BBBP2:  .WORD  12345,54321,23456,76543
074460 076543
18490
18491

```

18492
18493 074462
 (1) 074462 004767 030064
 (1)
 (1)
 (1)
 (1)
18494
18495
18503
18504
 (2)
 (3)
 (2) 074466
18505
18506
18507 074466 004767 000404
18508 074472 000000 000000
18509 074476 012345 067012
18510 074502 000000 000000
18511 074506 000000
18512 074510 000004
18513
18514
18515 074512 004737 075076
18516 074516 065652 125252
18517 074522 065600 000000
18518 074526 040252 125252
18519 074532 003000
18520 074534 003000
18521
18522
18523 074536 004767 000334
18524 074542 076400 000000
18525 074546 076400 000000
18526 074552 040200 000000
18527 074556 001000
18528 074560 001000
18529
18530 074562 004737 075076
18531 074566 056777 177777
18532 074572 054200 000000
18533 074576 042777 177777
18534 074602 000000
18535 074604 000000
18536
18537
18538 074606 004737 075076
18539 074612 012377 177777
18540 074616 012300 000000
18541 074622 040252 125252
18542 074626 000000
18543 074630 000000
18544
18545
18546 074632 004737 075076

BBBBDONE:

JSR PC, RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 475 DIVF TEST
;*****
TS475:

;CHECK DIVF WITH (AC=0).
CCC1: JSR PC, DIVFSUB
1#: .WORD 0,0 ;AC
2#: .WORD 12345,67012 ;FSRC
3#: .WORD 0,0 ;RES
4#: 0 ;FPS BEFORE EXECUTION
4 4 ;FPS AFTER EXECUTION

;TEST DIVF WITH AC POSITIVE, FSRC POSITIVE AND IN ROUND MODE.

CCC2: JSR PC, DIVFSUB
1#: .WORD 65652,125252 ;AC
2#: .WORD 65600,0 ;FSRC
3#: .WORD 40252,125252 ;RES
4#: 3000 ;FPS BEFORE EXECUTION.
3000 ;FPS AFTER EXECUTION.

;TEST DIVF WITH AC POSITIVE, FSRC POSITIVE.

CCC3: JSR PC, DIVFSUB
1#: .WORD 76400,0 ;AC
2#: .WORD 76400,0 ;FSRC
3#: .WORD 40200,0 ;RES
4#: 1000 ;FPS BEFORE EXECUTION.
1000 ;FPS AFTER EXECUTION.

;TEST DIVF WITH BOTH OPERANDS POSITIVE.

CCC4: JSR PC, DIVFSUB
1#: .WORD 56777,177777 ;AC
2#: .WORD 54200,0 ;FSRC
3#: .WORD 42777,177777 ;RES
4#: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

;TEST THE DIVF INSTRUCTION:

CCC5: JSR PC, DIVFSUB
1#: .WORD 12377,177777 ;AC
2#: .WORD 12300,0 ;FSRC
3#: .WORD 40252,125252 ;RES
4#: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION

;TEST DIVIDE ALGORITHM. TEST ROUND CONSTANT.

CCC6: JSR PC, DIVFSUB

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07 JAN 85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 25 19
T475 DIVF TEST

D9

SEQ 0314

18547 074636 064600 000001

18: .WORD 64600.1 ;AC

18549	074642	066600	000000	2#:	.WORD	66600.0		;FSRC
18550	074646	036200	000001	3#:	.WORD	36200.1		;RES
18551	074652	000000		4#:	0			;FPS BEFORE EXECUTION.
18552	074654	000000			0			;FPS AFTER EXECUTION.
18553								
18554								
18555	074656	004737	075076					
18556	074662	034577	177776	CCC7:	JSR	PC,8#DIVFSUB		
18557	074666	023400	000000	1#:	.WORD	34577.177776		;AC
18558	074672	051377	177776	2#:	.WORD	23400.0		;FSRC
18559	074676	000017		3#:	.WORD	51377.177776		;RES
18560	074700	000000		4#:	17			;FPS BEFORE EXECUTION.
18561					0			;FPS AFTER EXECUTION.
18562								
18563								
18564	074702	004737	075076					
18565	074706	067652	125252	CCC8:	JSR	PC,8#DIVFSUB		
18566	074712	056500	000000	1#:	.WORD	67652.125252		;AC
18567	074716	051343	107070	2#:	.WORD	56500.0		;FSRC
18568	074722	000000		3#:	.WORD	51343.107070		;RES
18569	074724	000000		4#:	0			;FPS BEFORE EXECUTION.
18570					0			;FPS AFTER EXECUTION.
18571								
18572	074726	004737	075076					
18573	074732	140400	000000	CCC9:	JSR	PC,8#DIVFSUB		
18574	074736	140500	000000	1#:	.WORD	140400.0		;AC
18575	074742	040052	125253	2#:	.WORD	140500.0		;FSRC
18576	074746	000000		3#:	.WORD	040052.125253		;RES
18577	074750	000000		4#:	0			;FPS BEFORE EXECUTION.
18578					0			;FPS AFTER EXECUTION.
18579								
18580	074752	004737	075076					
18581	074756	160077	000000	CCC10:	JSR	PC,8#DIVFSUB		
18582	074762	040277	000000	1#:	.WORD	160077.0		;AC
18583	074766	160000	000000	2#:	.WORD	40277.0		;FSRC
18584	074772	000007		3#:	.WORD	160000.0		;RES
18585	074774	000010		4#:	7			;FPS BEFORE EXECUTION.
18586					10			;FPS AFTER EXECUTION.
18587								
18588	074776	004737	075076					
18589	075002	040400	000000	CCC11:	JSR	PC,8#DIVFSUB		
18590	075006	140500	000000	1#:	.WORD	40400.0		;AC
18591	075012	140052	125253	2#:	.WORD	140500.0		;FSRC
18592	075016	000017		3#:	.WORD	140052.125253		;RES
18593	075020	000010		4#:	17			;FPS BEFORE EXECUTION.
18594					10			;FPS AFTER EXECUTION.
18595								
18596								
18597	075022	004737	075076					
18598	075026	060100	000001	CCC12:	JSR	PC,8#DIVFSUB		
18599	075032	040300	000000	1#:	.WORD	60100.1		;AC
18600	075036	060000	000000	2#:	.WORD	40300.0		;FSRC
18601	075042	000052		3#:	.WORD	60000.0		;RES
18602	075044	000040		4#:	52			;FPS BEFORE EXECUTION.
18603					40			;FPS AFTER EXECUTION.
18604								

;DIVF WITH POSITIVE OPERANDS AND ROUND MODE.

18605 075046 004767 000024
 18606 075052 060100 000001
 18607 075056 040300 000000
 18608 075062 060000 000001
 18609 075066 000005
 18610 075070 000000
 18611
 18612 075072 000137 075214
 18613
 18614
 18615
 18616
 18617
 18618
 18619
 18620
 18621
 18622
 18623
 18624
 18625
 18626
 18627
 18628
 18629
 18630
 18631
 18632
 18633
 18634
 18635
 18636
 18637
 18638
 18639
 18640 075076 012601
 18641 075100 012700 000200
 18642 075104 170100
 18643 075106 010100
 18644 075110 172410
 18645 075112 016100 000014
 18646 075116 170100
 18647 075120 010100
 18648 075122 062700 000004
 18649
 18650 075126 174410
 18651
 18652 075130 170204
 18653 075132 012700 000200
 18654 075136 170100
 18655
 18656 075140 012700 075204
 18657 075144 174010
 18658 075146 021061 000010
 18659 075152 001401
 (2) 075154 104000

```

CCC13: JSR      PC,DIVFSUB
14:      .WORD   60100,1      ;AC
24:      .WORD   40300,0      ;FSRC
34:      .WORD   60000,1      ;RES
44:      5          ;FPS BEFORE EXECUTION.
          0          ;FPS AFTER EXECUTION.

          JMP      @CCCCDONE    ;GO TO NEXT TEST.

;THIS SUBROUTINE, DIVFSUB, IS CALLED TO SET UP, EXECUTE
;AND CHECK THE RESULT OF A DIVF INSTRUCTION. IT IS CALLED THUS.
;
;          JSR      PC,@DIVFSUB
;          ACARG:  .WORD   X,X      ;AC OPERAND
;          FSRCARG: .WORD   X,X      ;FSRC OPERAND
;          RES:    .WORD   X,X      ;EXPECTED RESULT
;          FPSB:   .WORD   X        ;FPS BEFORE EXECUTION
;          FPSA:   .WORD   X        ;FPS AFTER EXECUTION
;          ERRES:  .WORD   X,X      ;ERROR RESULT
;          ERR:    ERROR  X        ;RESULT ERROR
;          CONT:   ;RETURN ADDRESS
;
;THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, DIVF IS EXECUTED.
;AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
;EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
;IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
;INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
;IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
;THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
;THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
;THEN THE FAILURE IS REPORTED IN DIVFSUB AND CONTROL IS PASSED TO
;CONT. IF NO ERRORS ARE DETECTED THEN DIVFSUB RETURNS CONTROL
;TO CONT.

DIVFSUB:  MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
          MOV      @200,R0      ;SET FD MODE.
          LDFPS   R0
          MOV      R1,R0        ;LOAD THE AC OPERAND.
          LDD     (R0),ACO
          MOV      14(R1),R0    ;LOAD THE FPS
          LDFPS   R0
          MOV      R1,R0
          ADD     @4,R0         ;ESTABLISH A POINTER TO FSRC.

14:      DIVF    (R0),ACO      ;TEST INSTRUCTION.

          STFPS   R4           ;GET THE FPS.
          MOV      @200,R0      ;SET FD MODE
          LDFPS   R0

          MOV      @DIVFT,R0    ;GET THE RESULT OF THE DIVF.
          STD     ACO,(R0)
          CMP     (R0),10(R1)   ;IS THE RESULT CORRECT?
          BEQ     24
          EMT
  
```

```
18660 075156 026061 000002 000012 24:  CMP 2(R0),12(R1)
18661 075164 001401 BEQ 34
(2) 075166 104000 EMT
18662 075170 026104 000016 34:  CMP 16(R1),R4 ;IS FPS CORRECT?
18663 075174 001401 BEQ 44
(2) 075176 104000 EMT
18664 075200 000161 000020 44:  JMP 20(R1) ;IF NO ERRORS OCCURRED RETURN.
18665
18666 075204 000000 000000 000000 DIVFT: .WORD 0,0,0,0
075212 000000
18667
18668 075214 CCCDONE:
(1) 075214 004767 027332 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
18669
18670
18677
18678 ;*****
(2) ;TEST 476 DIVD TEST
(3) ;*****
(2) 075220 TS476:
18679
18680 ;DIVD TEST WITH POSITIVE OPERANDS AND IN ROUND MODE.
18681 075220 004737 075564 DDD1: JSR PC,DDIVDSUB
18682 075224 034277 000000 000000 14: .WORD 34277,0,0,0 ;AC
075232 000000
18683 075234 040277 000000 000000 24: .WORD 40277,0,0,0 ;FSRC
075242 000000
18684 075244 034200 000000 000000 34: .WORD 34200,0,0,0 ;RES
075252 000000
18685 075254 000200 44: 200 ;FPS BEFORE EXECUTION.
18686 075256 000200 200 ;FPS AFTER EXECUTION.
18687
18688 ;DIVD WITH AC NEGATIVE AND FSRC POSITIVE IN TRUNCATE MODE.
18689 075260 004737 075564 DDD2: JSR PC,DDIVDSUB
18690 075264 134277 000000 000000 14: .WORD 134277,0,0,0 ;AC
075272 000000
18691 075274 040277 000000 000000 24: .WORD 40277,0,0,0 ;FSRC
075302 000000
18692 075304 134200 000000 000000 34: .WORD 134200,0,0,0 ;RES
075312 000000
18693 075314 000207 44: 207 ;FPS BEFORE EXECUTION.
18694 075316 000210 210 ;FPS AFTER EXECUTION.
18695
18696 ;DIVD TEST WITH OPERANDS BOTH NEGATIVE AND IN TRUNCATE MODE.
18697 075320 004767 000240 DDD3: JSR PC,DIVDSUB
18698 075324 134300 000000 000000 14: .WORD 134300,0,0,1 ;AC
075332 000001
18699 075334 140300 000000 000000 24: .WORD 140300,0,0,0 ;FSRC
075342 000000
18700 075344 034200 000000 000000 34: .WORD 34200,0,0,0 ;RES
075352 000000
18701 075354 000250 44: 250 ;FPS BEFORE EXECUTION.
```



```
18702 075356 000240                240                ;FPS AFTER EXECUTION.
18703
18704                                ;DIVD WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
18705 075360 004737 075564          DDD4: JSR      PC, @DIVDSUB
18706 075364 034300 000000 000000 1#: .WORD    34300,0,0,1    ;AC
      075372 000001
18707 075374 140300 000000 000000 2#: .WORD    140300,0,0,0    ;FSRC
      075402 000000
18708 075404 134200 000000 000000 3#: .WORD    134200,0,0,1    ;RES
      075412 000001
18709 075414 000207          4#:      207          ;FPS BEFORE EXECUTION.
18710 075416 000210          210          ;FPS AFTER EXECUTION.
18711
18712                                ;DIVD TEST.
18713 075420 004737 075564          DDD5: JSR      PC, @DIVDSUB
18714 075424 100400 000000 000000 1#: .WORD    100400,0,0,0    ;AC
      075432 000000
18715 075434 000500 000000 000000 2#: .WORD     500,0,0,0    ;FSRC
      075442 000000
18716 075444 140052 125252          3#: .WORD    140052,125252 ;RES
18717 075450 125252 125252          .WORD    125252,125252
18718 075454 007647          4#:      7647          ;FPS BEFORE EXECUTION.
18719 075456 007650          7650          ;FPS AFTER EXECUTION.
18720
18721
18722                                ;DIVD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
18723 075460 004737 075564          DDD6: JSR      PC, @DIVDSUB
18724 075464 000400 000000 000000 1#: .WORD     400,0,0,0    ;AC
      075472 000000
18725 075474 100500 000000 000000 2#: .WORD    100500,0,0,0    ;FSRC
      075502 000000
18726 075504 140052 125252          3#: .WORD    140052,125252 ;RES
18727 075510 125252 125253          .WORD    125252,125253
18728 075514 007707          4#:      7707          ;FPS BEFORE EXECUTION.
18729 075516 007710          7710          ;FPS AFTER EXECUTION.
18730
18731                                ;DIVD TEST.
18732 075520 004737 075564          DDD7: JSR      PC, @DIVDSUB
18733 075524 170360 170360          1#: .WORD    170360,170360 ;AC
18734 075530 170360 170360          .WORD    170360,170360
18735 075534 170360 170360          2#: .WORD    170360,170360 ;FSRC
18736 075540 170360 170360          .WORD    170360,170360
18737 075544 040200 000000 000000 3#: .WORD     40200,0,0,0    ;RES
      075552 000000
18738 075554 007717          4#:      7717          ;FPS BEFORE EXECUTION.
18739 075556 007700          7700          ;FPS AFTER EXECUTION.
18740
18741 075560 000137 075706          JMP      @DDDDONE    ;GO TO NEXT TEST.
18742
18743
18744                                ;THIS SUBROUTINE, DIVDSUB, IS CALLED TO SET UP, EXECUTE
18745                                ;AND CHECK THE RESULT OF A DIVD INSTRUCTION. IT IS CALLED THUS:
18746                                ;
18747                                ;          JSR      PC, @DIVDSUB
18748                                ;          ACARG: .WORD    X,X,X,X          ;AC OPERAND
18749                                ;          FSRCARG: .WORD    X,X,X,X          ;FSRC OPERAND
```

18750				:	RES:	.WORD	X,X,X,X	:	EXPECTED RESULT
18751				:	FPSB:	.WORD	X	:	FPS BEFORE EXECUTION
18752				:	FPSA:	.WORD	X	:	FPS AFTER EXECUTION
18753				:	ERRES:	.WORD	X,X,X,X	:	ERROR RESULT
18754				:	ERR:	ERROR	X	:	RESULT ERROR
18755				:	CONT:			:	RETURN ADDRESS
18756				:				:	
18757				:				:	
18758				:				:	
18759				:				:	
18760				:				:	
18761				:				:	
18762				:				:	
18763				:				:	
18764				:				:	
18765				:				:	
18766				:				:	
18767				:				:	
18768				:				:	
18769				:				:	
18770	075564	012601			DIVDSUB:	MOV	(SP)+,R1		;GET A POINTER TO THE ARGUMENTS.
18771	075566	012700	000200			MOV	#200,R0		;SET FD MODE.
18772	075572	170100				LDFPS	R0		
18773									
18774	075574	010100				MOV	R1,R0		;SET UP THE ACO OPERAND.
18775	075576	172410				LDD	(R0),ACO		
18776	075600	016100	000030			MOV	30(R1),R0		;LOAD THE FPS.
18777	075604	170100				LDFPS	R0		
18778									
18779	075606	010100				MOV	R1,R0		;ESTABLISH A POINTER TO FSRC.
18780	075610	062700	000010			ADD	#10,R0		
18781									
18782	075614	174410			1#:	DIVD	(R0),ACO		;EXECUTE THE TEST INSTRUCTION.
18783	075616	170204				STFPS	R4		;GET THE FPS.
18784	075620	012700	000200			MOV	#200,R0		;SET FD MODE.
18785	075624	170100				LDFPS	R0		
18786	075626	012700	075676			MOV	#DIVDT,R0		;GET THE RESULT.
18787	075632	174010				STD	ACO,(R0)		
18788	075634	010102				MOV	R1,R2		;CHECK THE RESULT.
18789	075636	062702	000020			ADD	#20,R2		
18790	075642	012703	075676			MOV	#DIVDT,R3		
18791	075646	012705	000004			MOV	#4,R5		
18792	075652	022223			2#:	CMF	(R2)+,(R3)+		
18793	075654	001401				BEQ	3#		
(2)	075656	104000				EMT			
18794	075660	077504			3#:	SQB	R5,2#		
18795									
18796	075662	026104	000032			CMF	32(R1),R4		;IS FPS CORRECT?
18797	075666	001401				BEQ	4#		
(2)	075670	104000				EMT			
18798	075672	000161	000034		4#:	JMP	34(R1)		;RETURN.
18799	075676	000000	000000	000000		DIVDT:	.WORD	0,0,0,0	
	075704	000000							
18800									
18801	075706				DDDDONE:				
(1)	075706	004767	026640			JSR	PC,.RSET		;GO INITIALIZE THE FPS AND STACK; AND

;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

(1)
(1)
(1)
(1)
18802
18803
18811
18812
(2)
(3)
(2) 075712
18813
18814
18815 075712 004737 076322
18816 075716 000000 000000
18817 075722 000000 000000
18818 075726 000000 000000
18819 075732 007517
18820 075734 007504
18821
18822
18823 075736 004737 076322
18824 075742 071625 034435
18825 075746 000000 000000
18826 075752 000000 000000
18827 075756 000013
18828 075760 000004
18829
18830
18831 075762 004737 076322
18832 075766 000000 000000
18833 075772 071625 153443
18834 075776 000000 000000
18835 076002 007500
18836 076004 007504
18837
18838
18839 076006 004737 076322
18840 076012 040200 000000
18841 076016 040177 177777
18842 076022 040177 177777
18843 076026 000017
18844 076030 000000
18845
18846
18847 076032 004767 000264
18848 076036 040177 177777
18849 076042 040200 000000
18850 076046 040177 177777
18851 076052 000040
18852 076054 000040
18853
18854
18855 076056 004737 076322
18856 076062 040100 000000
18857 076066 040100 000000

```
.....  
;TEST 477      MULF TEST  
;.....  
TS477:  
  
;MULF WITH (FSRC=AC=0)  
EEE1:  JSR      PC,BMULFSUB  
1$:    .WORD    0,0      ;AC  
2$:    .WORD    0,0      ;FSRC  
3$:    .WORD    0,0      ;RES  
4$:    7517      ;FPS BEFORE EXECUTION.  
      7504      ;FPS AFTER EXECUTION.  
  
;MULF WITH (FSRC=0).  
EEE2:  JSR      PC,BMULFSUB  
1$:    .WORD    71625,34435 ;AC  
2$:    .WORD    0,0      ;FSRC  
3$:    .WORD    0,0      ;RES  
4$:    13        ;FPS BEFORE EXECUTION.  
      4         ;FPS AFTER EXECUTION.  
  
;MULF WITH (AC=0)  
EEE3:  JSR      PC,BMULFSUB  
1$:    .WORD    0,0      ;AC  
2$:    .WORD    071625,153443 ;FSRC  
3$:    .WORD    0,0      ;RES  
4$:    7500      ;FPS BEFORE EXECUTION.  
      7504      ;FPS AFTER EXECUTION.  
  
;MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.  
EEE4:  JSR      PC,BMULFSUB  
1$:    .WORD    40200,0    ;AC  
2$:    .WORD    40177,-1  ;FSRC  
3$:    .WORD    40177,-1  ;RES  
4$:    17        ;FPS BEFORE EXECUTION.  
      0         ;FPS AFTER EXECUTION.  
  
;MULF WITH AC POSITIVE AND FSRC POSITIVE IN TRUNCATE MODE.  
EEE5:  JSR      PC,MULFSUB  
1$:    .WORD    40177,-1  ;AC  
2$:    .WORD    40200,0    ;FSRC  
3$:    .WORD    40177,-1  ;RES  
4$:    40        ;FPS BEFORE EXECUTION.  
      40        ;FPS AFTER EXECUTION.  
  
;MULF WITH BOTH OPERANDS POSITIVE NORMALIZE TEST.  
EEE6:  JSR      PC,BMULFSUB  
1$:    .WORD    40100,0    ;AC  
2$:    .WORD    40100,0    ;FSRC
```

18858 076072 040020 000000
 18859 076076 000012
 18860 076100 000000
 18861
 18862
 18863 076102 004737 076322
 18864 076106 017500 000000
 18865 076112 023652 125252
 18866 076116 003177 177777
 18867 076122 007417
 18868 076124 007400
 18869
 18870
 18871 076126 004737 076322
 18872 076132 040342 000000
 18873 076136 176542 000000
 18874 076142 176707 102000
 18875 076146 000007
 18876 076150 000010
 18877
 18878
 18879 076152 004737 076322
 18880 076156 140200 000000
 18881 076162 007417 007417
 18882 076166 107417 007417
 18883 076172 000000
 18884 076174 000010
 18885
 18886
 18887 076176 004737 076322
 18888 076202 144600 000000
 18889 076206 154000 000000
 18890 076212 060400 000000
 18891 076216 000017
 18892 076220 000000
 18893
 18894
 18895 076222 004737 076322
 18896 076226 140300 000000
 18897 076232 160000 000001
 18898 076236 060100 000002
 18899 076242 000010
 18900 076244 000000
 18901
 18902
 18903 076246 004737 076322
 18904 076252 060000 000001
 18905 076256 140300 000000
 18906 076262 160100 000001
 18907 076266 007547
 18908 076270 007550
 18909
 18910
 18911 076272 004737 076322
 18912 076276 040277 000000
 18913 076302 060000 000001

3# : .WORD 40020.0 ;RES
 4# : 12 ;FPS BEFORE EXECUTION.
 0 ;FPS AFTER EXECUTION

 ;MULF WITH BOTH OPERANDS POSITIVE IN ROUND MODE.
 EEE7: JSR PC, @MULFSUB
 1# : .WORD 17500.0 ;AC
 2# : .WORD 23652.125252 ;FSRC
 3# : .WORD 3177.-1 ;RES
 4# : 7417 ;FPS BEFORE EXECUTION.
 7400 ;FPS AFTER EXECUTION.

 ;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.
 EEE8: JSR PC, @MULFSUB
 1# : .WORD 40342.0 ;AC
 2# : .WORD 176542.0 ;FSRC
 3# : .WORD 176707.102000 ;RES
 4# : 7 ;FPS BEFORE EXECUTION.
 10 ;FPS AFTER EXECUTION.

 ;MULF WITH AC NEGATIVE AND FSRC POSITIVE IN ROUND MODE.
 EEE9: JSR PC, @MULFSUB
 1# : .WORD 140200.0 ;AC
 2# : .WORD 7417.7417 ;FSRC
 3# : .WORD 107417.7417 ;RES
 4# : 0 ;FPS BEFORE EXECUTION.
 10 ;FPS AFTER EXECUTION.

 ;MULF WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
 EEE10: JSR PC, @MULFSUB
 1# : .WORD 144600.0 ;AC
 2# : .WORD 154000.0 ;FSRC
 3# : .WORD 60400.0 ;RES
 4# : 17 ;FPS BEFORE EXECUTION.
 0 ;FPS AFTER EXECUTION.

 ;MULF BOTH OPERANDS NEGATIVE IN ROUND MODE.
 EEE11: JSR PC, @MULFSUB
 1# : .WORD 140300.0 ;AC
 2# : .WORD 160000.1 ;FSRC
 3# : .WORD 60100.2 ;RES
 4# : 10 ;FPS BEFORE EXECUTION.
 0 ;FPS AFTER EXECUTION.

 ;MULF WITH AC POSITIVE AND FSRC NEGATIVE IN TRUNCATE MODE.
 EEE12: JSR PC, @MULFSUB
 1# : .WORD 60000.1 ;AC
 2# : .WORD 140300.0 ;FSRC
 3# : .WORD 160100.1 ;RES
 4# : 7547 ;FPS BEFORE EXECUTION.
 7550 ;FPS AFTER EXECUTION.

 ;MULF WITH AC POSITIVE AND FSRC POSITIVE IN ROUND MODE.
 EEE13: JSR PC, @MULFSUB
 1# : .WORD 40277.0 ;AC
 2# : .WORD 60000.1 ;FSRC

18914 076306 060077 000001
 18915 076312 000014
 18916 076314 000000
 18917
 18918 076316 000167 000116
 18919
 18920
 18921
 18922
 18923
 18924
 18925
 18926
 18927
 18928
 18929
 18930
 18931
 18932
 18933
 18934
 18935
 18936
 18937
 18938
 18939
 18940
 18941
 18942
 18943
 18944
 18945
 18946 076322 012601
 18947 076324 012700 000200
 18948 076330 170100
 18949 076332 010100
 18950 076334 172410
 18951 076336 016100 000014
 18952 076342 170100
 18953 076344 010100
 18954 076346 062700 000004
 18955
 18956 076352 171010
 18957
 18958 076354 170204
 18959 076356 012700 000200
 18960 076362 170100
 18961
 18962 076364 012700 076430
 18963 076370 174010
 18964 076372 021061 000010
 18965 076376 001401
 (2) 076400 104000
 18966 076402 026061 000002 000012 24:
 18967 076410 001401
 (2) 076412 104000

```

34: .WORD 60077.1 ;RES
44: 14 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.

JMP EEEDONE ;GO TO THE NEXT TEST.

;THIS SUBROUTINE, MULFSUB, IS CALLED TO SET UP, EXECUTE
;AND CHECK THE RESULT OF A MULF INSTRUCTION. IT IS CALLED THUS:
;
; JSR PC,BMULFSUB
; ACARG: .WORD X,X ;AC OPERAND
; FSRCARG: .WORD X,X ;FSRC OPERAND
; RES: .WORD X,X ;EXPECTED RESULT
; FPSB: .WORD X ;FPS BEFORE EXECUTION
; FPSA: .WORD X ;FPS AFTER EXECUTION
; ERRES: .WORD X,X ;ERROR RESULT
; ERR: ERROR X ;RESULT ERROR
; CONT: ;RETURN ADDRESS
;
;THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
;FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, MULF IS EXECUTED.
;AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
;EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
;IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
;INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
;IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
;THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
;THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
;THEN THE FAILURE IS REPORTED IN MULFSUB AND CONTROL IS PASSED TO
;CONT. IF NO ERRORS ARE DETECTED THEN MULFSUB RETURNS CONTROL
;TO CONT.

MULFSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;LOAD THE AC OPERAND.
LDD (R0),ACO
MOV 14(R1),R0 ;LOAD THE FPS
LDFPS R0
MOV R1,R0
ADD #4,R0 ;ESTABLISH A POINTER TO FSRC.

14: MULF (R0),ACO ;TEST INSTRUCTION.

STFPS R4 ;GET THE FPS.
MOV #200,R0 ;SET FD MODE
LDFPS R0

MOV #MULFT,R0 ;GET THE RESULT OF THE MULF.
STD ACO,(R0)
CMP (R0),10(R1) ;IS THE RESULT CORRECT?
BEQ 24
;
;
24: CMP 2(R0),12(R1)
BEQ 34
;
;

```

```

18968 076414 026104 000016      3#:    CMP      16(R1),R4      ;IS FPS CORRECT?
18969 076420 001401              BEQ      4#
      (2) 076422 104000              EMT
18970 076424 000161 000020      4#:    JMP      20(R1)      ;IF NO ERRORS OCCURRED RETURN.
18971
18972 076430 000000 000000 000000 MULFT:  .WORD  0,0,0,0
      076436 000000
18973
18974 076440      EEEDONE:
      (1) 076440 004767 026106      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                                     ;SEE IF THE USER HAS EXPRESSED
      (1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                                     ;THE USER TYPED CONTROL G?).
18975
18976
18984
18985      ;*****
      (2)      ;TEST 500      MULD TEST
      (3)      ;*****
      (2) 076444      TS500:
18986
18987      ;MULD TEST WITH AC POSITIVE AND FSRC POSITIVE.
18988 076444 004737 076650      FFF1:  JSR      PC,BMULDSUB
18989 076450 040200 000000 000000 1#:    .WORD  40200,0,0,0      ;AC
      076456 000000
18990 076460 023777 177777 177777 2#:    .WORD  23777,-1,-1,-1 ;FSRC
      076466 177777
18991 076470 023777 177777 177777 3#:    .WORD  23777,-1,-1,-1 ;RES
      076476 177777
18992 076500 000217      4#:    217      ;FPS BEFORE EXECUTION.
18993 076502 000200      200      ;FPS AFTER EXECUTION.
18994
18995      ;MULD TEST WITH BOTH OPERANDS POSITIVE TRUNCATION TEST.
18996 076504 004767 000140      FFF2:  JSR      PC,MULDSUB
18997 076510 065400 000000 000000 1#:    .WORD  65400,0,0,1      ;AC
      076516 000001
18998 076520 037577 177777 177777 2#:    .WORD  37577,-1,-1,-2 ;FSRC
      076526 177776
18999 076530 064777 177777 177777 3#:    .WORD  64777,-1,-1,-1 ;RES
      076536 177777
19000 076540 000247      4#:    247      ;FPS BEFORE EXECUTION.
19001 076542 000240      240      ;FPS AFTER EXECUTION.
19002
19003      ;MULD TEST WITH BOTH OPERANDS NEGATIVE IN ROUND MODE.
19004 076544 004737 076650      FFF3:  JSR      PC,BMULDSUB
19005 076550 137577 177777 177777 1#:    .WORD  137577,-1,-1,-2 ;AC
      076556 177776
19006 076560 165400 000000 000000 2#:    .WORD  165400,0,0,1      ;FSRC
      076566 000001
19007 076570 065000 000000 000000 3#:    .WORD  65000,0,0,0      ;RES
      076576 000000
19008 076600 007717      4#:    7717      ;FPS BEFORE EXECUTION.
19009 076602 007700      7700      ;FPS AFTER EXECUTION.
19010
19011      ;MULD TEST WITH AC POSITIVE AND FSRC NEGATIVE IN ROUND MODE.

```

19012 076604 004737 076650
19013 076610 017500 000000
19014 076620 123652 125252
19015 076624 125252 125252
19016 076630 103177 177777
19017 076640 000200
19018 076642 000210
19019
19020 076644 000167 000122
19021
19022
19023
19024
19025
19026
19027
19028
19029
19030
19031
19032
19033
19034
19035
19036
19037
19038
19039
19040
19041
19042
19043
19044
19045
19046
19047 076650 012601
19048 076652 012700 000200
19049 076656 170100
19050
19051 076660 010100
19052 076662 172410
19053 076664 016100 000030
19054 076670 170100
19055
19056 076672 010100
19057 076674 062700 000010
19058
19059 076700 171010
19060
19061 076702 170204
19062 076704 012700 000200
19063 076710 170100
19064
19065 076712 012700 076762

```
FFF4: JSR PC, @MULDSUB
14: .WORD 17500,0,0,0 ;AC
24: .WORD 123652,125252 ;FSRC
34: .WORD 103177,1,-1,1 ;RES
44: 200 ;FPS BEFORE EXECUTION.
210 ;FPS AFTER EXECUTION.

JMP FFFDONE

; THIS SUBROUTINE, MULDSUB, IS CALLED TO SET UP, EXECUTE
; AND CHECK THE RESULT OF A MULD INSTRUCTION. IT IS CALLED THUS:;
; JSR PC, @MULDSUB
; ACARG: .WORD X,X,X,X ;AC OPERAND
; FSRCARG: .WORD X,X,X,X ;FSRC OPERAND
; RES: .WORD X,X,X,X ;EXPECTED RESULT
; FPSB: .WORD X ;FPS BEFORE EXECUTION
; FPSA: .WORD X ;FPS AFTER EXECUTION
; ERRES: .WORD X,X,X,X ;ERROR RESULT
; ERR: ERROR X ;RESULT ERROR
; CONT: ;RETURN ADDRESS

; THE OPERANDS ARE SET UP (USING ACO FOR THE AC OPERAND). THEN
; FPSB IS LOADED INTO THE FPS. THE INSTRUCTION, MULD IS EXECUTED.
; AFTER THE EXECUTION THE RESULT IS CHECKED AGAINST THE
; EXPECTED CORRECT RESULT, RES. IF IT IS CORRECT THEN THE FPS
; IS CHECKED WITH THE EXPECTED CORRECT FPS, FPSA. IF THE FPS WAS
; INCORRECT THEN IT IS REPORTED. IF THE RESULT WAS INCORRECT IT
; IS COMPARED WITH ERRES IN AN ATTEMPT TO ANALYSE THE ERROR. IF
; THE INCORRECT RESULT MATCHED ERRES THEN CONTROL IS PASSED TO
; THE ERROR CALL AT ERR. IF THE INCORRECT RESULT DID NOT MATCH ERRES
; THEN THE FAILURE IS REPORTED IN MULDSUB AND CONTROL IS PASSED TO
; CONT. IF NO ERRORS ARE DETECTED THEN MULDSUB RETURNS CONTROL
; TO CONT.

MULDSUB: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV #200,R0 ;SET FC MODE.
LDFPS R0

MOV R1,R0 ;SET UP THE ACO OPERAND.
LDD (R0),ACO
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0

MOV R1,R0 ;ESTABLISH A POINTER TO FSRC.
ADD #10,R0

14: MULD (R0),ACO ;EXECUTE THE TEST INSTRUCTION.

STFPS R4 ;GET THE FPS.
MOV #200,R0 ;SET FD MODE.
LDFPS R0

MOV @MULDT,R0 ;GET THE RESULT.
```

```

19066 J76716 174010 STD AC0,(R0)
19067 076720 010102 MOV R1,R2 ;CHECK THE RESULT
19068 076722 062702 000020 ADD #20,R2
19069 076726 012703 C76762 MOV #MULDT,R3
19070 076732 012705 000004 MOV #4,R5
19071 076736 022223 21: CMP (R2),.(R3).
19072 076740 001401 BEQ 31
(2) 076742 104000 EMT
19073 076744 077504 31: SOB R5,21
19074
19075 076746 026104 000032 CMP 32(R1),R4 ;IS FPS CORRECT?
19076 076752 001401 BEQ 41
(2) 076754 104000 EMT
19077 076756 000161 000034 41: JMP 34(R1) ;RETURN.
19078
19079 076762 000000 000000 000000 MULDT: .WORD 0,0,0,0
076770 000000
19080 076772 FFFDONE:
(1) 076772 004767 025554 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
19081
19082
19091
19092 ;*****
(2) ;TEST 501 UNDER/OVER FLOW, USING MULF WITH TRAPS DISABLED. TEST
(3) ;*****
(2) 076776 T5501:
19093
19094 ;UNDERFLOW, WITH EXPONENT OF RESULT = -129
19095 076776 004737 077142 III1: JSR PC,B00VUNFNT
19096 077002 020200 000000 11: .WORD 20200,0 ;AC
19097 077006 020000 000000 21: .WORD 20000,0 ;FSRC
19098 077012 000000 000000 31: .WORD 0,0 ;RES
19099 077016 000000 51: 0 ;FPS BEFORE EXECUTION.
19100 077020 000004 61: 4 ;FPS AFTER EXECUTION.
19101 077022 000012 61: 12 ;FEC
19102 077024 177777 61: -1 ;FLAG
19103
19104 ;UNDERFLOW, WITH EXPONENT OF RESULT = -193
19105 077026 004737 077142 III2: JSR PC,B00VUNFNT
19106 077032 010200 000000 11: .WORD 10200,0 ;AC
19107 077036 010000 000000 21: .WORD 10000,0 ;FSRC
19108 077042 000000 000000 31: .WORD 0,0 ;RES
19109 077046 005013 51: 5013 ;FPS BEFORE EXECUTION.
19110 077050 005004 51: 5004 ;FPS AFTER EXECUTION.
19111 077052 000012 61: 12 ;FEC
19112 077054 177777 61: -1 ;FLAG
19113
19114 077056 004737 077142 ;OVERFLOW, EXPONENT OF RESULT = 128
19115 077062 060200 000000 III3: JSR PC,B00VUNFNT
19116 077066 060000 000000 11: .WORD 60200,0 ;AC
19117 077072 000000 000000 21: .WORD 60000,0 ;FSRC
19118 077076 000000 31: .WORD 0,0 ;RES
51: 0 ;FPS BEFORE EXECUTION.

```


C10

19119 077100 000006
 19120 077102 000010
 19121 077104 000000
 19122
 19123 077106 004737 077142
 19124 077112 060200 000000
 19125 077116 060200 000000
 19126 077122 000000 000000
 19127 077126 006011
 19128 077130 006006
 19129 077132 000010
 19130 077134 000000
 19131 077136 000167 000132
 19132
 19133
 19134
 19135
 19136
 19137
 19138
 19139
 19140
 19141
 19142
 19143
 19144
 19145
 19146
 19147
 19148
 19149
 19150
 19151
 19152
 19153
 19154
 19155
 19156
 19157
 19158
 19159
 19160
 19161
 19162
 19163
 19164
 19165
 19166
 19167
 19168
 19169
 19170
 19171
 19172
 19173 077142 012601
 19174 077144 012700 000200

```

      6      ;FPS AFTER EXECUTION.
61:    10    ;FEC
      0      ;FLAG
      ;OVERFLOW, EXPONENT OF RESULT = 130
III4:  JSR   PC, @OVUNFNT
11:    .WORD 60200.0 ;AC
21:    .WORD 60200.0 ;FSRC
31:    .WORD 0.0     ;RES
51:    6011    ;FPS BEFORE EXECUTION.
      6006    ;FPS AFTER EXECUTION.
61:    10     ;FEC
      0      ;FLAG
81:    JMP   IIIDONE ;GO TO NEXT TEST.
  
```

;THIS SUBROUTINE, OVUNFNT, IS USED TO SET UP THE OPERANDS, EXECUTE
 ;THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
 ;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
 ;TO IT IS MADE THUS:

```

;
;          ACARG: .WORD X,X      ;AC OPERAND
;          FSRCARG: .WORD X,X    ;FSRC OPERAND
;          RES: .WORD X,X       ;EXPECTED RESULT
;          ERRES: .WORD X,X     ;ERROR RESULT
;          FPSB: .WORD X        ;FPS BEFORE EXECUTION
;          FPSA: .WORD X        ;FPS AFTER EXECUTION
;          FEC: .WORD X         ;EXPECTED FEC
;          FLAG: .WORD X        ;0/-1.OVER/UNDER FLOW FLAG
;          ERR1: ERROR X        ;TRAP ERROR.
;          BR   CONT
;          ERR2: ERROR X        ;DATA, RESULT ERROR
;          CONT:                ;RETURN ADDRESS
  
```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 ;THE MULF INSTRUCTION IS EXECUTED. IF NO TRAP OCCURS THEN THE
 ;RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 ;COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFNT RETURNS CONTROL
 ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFNT
 ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 ;MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFNT
 ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
 ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFNT WILL
 ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
 ;IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNFNT WILL READ THE FEC.
 ;SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNFNT WILL
 ;STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
 ;FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNFNT WILL REPORT
 ;THE ERROR AND RETURN TO CONT. NOTE THAT OVUNFNT USES THE FLAG
 ;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
 ;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

```

OVUNFNT:  MOV   (SP)+,R1    ;GET A POINTER TO THE ARGUMENTS.
          MOV   @200,R0     ;SET FD MODE.
  
```

```

19175 077150 170100 LDFPS R0
19176
19177 077152 010100 MOV R1,R0 ;LOAD ACO, OPERAND
19178 077154 172410 LDD (R0),ACO
19179 077156 016100 000014 MOV 14(R1),R0 ;LOAD THE FPS
19180 077162 170100 LDFPS R0
19181 077164 012737 077244 000244 MOV #254,B#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
19182 ;OF ERROR.
19183 077172 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF FSRC.
19184 077174 062700 000004 ADD #4,R0
19185
19186 077200 171010 18: MULF (R0),ACO ;TEST INSTRUCTION.
19187
19188 077202 170204 28: STFPS R4 ;GET FPS.
19189 077204 170305 STST R5 ;GET FEC.
19190 077206 012700 000200 MOV #200,R0 ;SET FD MODE.
19191 077212 170100 LDFPS R0
19192 077214 012700 077264 MOV #OVFNTR,R0 ;GET THE RESULT.
19193 077220 174010 STD ACO,(R0)
19194 077222 012700 077264 MOV #OVFNTR,R0 ;CHECK THE RESULT.
19195 077226 010102 MOV R1,R2
19196 077230 062702 000010 ADD #10,R2
19197 077234 012703 000002 MOV #2,R3
19198 077240 022022 38: CMP (R0)+,(R2)+
19199 077242 001401 BEQ 58
(1) 077244 258:
(2) 077244 104000 EMT
19200 077246 077304 58: SOB R3,58
19201
19202 077250 026104 000016 CMP 16(R1),R4 ;WAS FPS CORRECT?
19203 077254 001401 BEQ 48
(2) 077256 104000 EMT
19204 077260 000161 000024 48: JMP 24(R1) ;RETURN, TEST COMPLETED.
19205
19206 077264 000000 000000 000000 OVFNTR: .WORD 0,0,0,0
077272 000000
19207
19208 077274 IIIDONE:
(1) 077274 004767 025252 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
19209
19210
19211
19220
19221 ;*****
(2) ;TEST 502 UNDER\OVER FLOW, USING MULD WITH TRAP DISABLED, TEST
(3) ;*****
(2) 077300 T5502:
19222
19223 ;UNDERFLOW, EXPONENT OF RESULT=-129
19224 077300 004737 077524 JJJ1: JSR PC,B#OVUNDNT
19225 077304 020200 000000 18: .WORD 20200,0 ;AC
19226 077310 127272 000000 .WORD 127272,0

```

```

19227 077314 020000 000000 000000 24: .WORD 20000,0,0,0 ;FSRC
      077322 000000
19228 077324 000000 000000 000000 34: .WORD 0,0,0,0 ;RES
      077332 000000
19229 077334 000200 54: 200 ;FPS BEFORE EXECUTION.
19230 077336 000204 64: 204 ;FPS AFTER EXECUTION.
19231 077340 000012 64: 12 ;FEC
19232 077342 177777 -1 ;FLAG
19233
19234 ;UNDERFLOW, EXPONENT OF RESULT = -193
19235 077344 004737 077524 JJJ2: JSR PC,B#OVUNDNT
19236 077350 010200 000000 14: .WORD 10200,0 ;AC
19237 077354 123456 000000 .WORD 123456,0
19238 077360 010000 000000 000000 24: .WORD 10000,0,0,0 ;FSRC
      077366 000000
19239 077370 000000 000000 000000 34: .WORD 0,0,0,0 ;RES
      077376 000000
19240 077400 005213 54: 5213 ;FPS BEFORE EXECUTION.
19241 077402 005204 64: 5204 ;FPS AFTER EXECUTION.
19242 077404 000012 64: 12 ;FEC
19243 077406 177777 -1 ;FLAG
19244
19245 ;OVERFLOW, EXPONENT OF RESULT = 128
19246 077410 004737 077524 JJJ3: JSR PC,B#OVUNDNT
19247 077414 060200 000000 14: .WORD 60200,0 ;AC
19248 077420 065432 000000 .WORD 65432,0
19249 077424 060000 000000 000000 24: .WORD 60000,0,0,0 ;FSRC
      077432 000000
19250 077434 000000 000000 000000 34: .WORD 0,0,0,0 ;RES
      077442 000000
19251 077444 000200 54: 200 ;FPS BEFORE EXECUTION.
19252 077446 000206 64: 206 ;FPS AFTER EXECUTION.
19253 077450 000010 64: 10 ;FEC
19254 077452 000000 0 ;FLAG
19255
19256 ;OVERFLOW, EXPONENT OF RESULT = 130
19257 077454 004737 077524 JJJ4: JSR PC,B#OVUNDNT
19258 077460 060200 000000 14: .WORD 60200,0 ;AC
19259 077464 125252 000000 .WORD 125252,0
19260 077470 060200 000000 000000 24: .WORD 60200,0,0,0 ;FSRC
      077476 000000
19261 077500 000000 000000 000000 34: .WORD 0,0,0,0 ;RES
      077506 000000
19262 077510 006211 54: 6211 ;FPS BEFORE EXECUTION.
19263 077512 006206 64: 6206 ;FPS AFTER EXECUTION.
19264 077514 000010 64: 10 ;FEC
19265 077516 000000 0 ;FLAG
19266 077520 000137 077656 84: JMP B#JJJDONE ;GO TO NEXT TEST
19267
19268 ;THIS SUBROUTINE, OVUNDNT, IS USED TO SET UP THE OPERANDS, EXECUTE
19269 ;THE MULD INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
19270 ;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
19271 ;TO IT IS MADE THUS:
19272 ;
19273 ; ACARG: .WORD X,X,X,X ;AC OPERAND
19274 ; FSRCARG: .WORD X,X,X,X ;FSRC OPERAND

```

19275	:	RES:	.WORD	X,X,X,X	; EXPECTED RESULT
19276	:	ERRES:	.WORD	X,X,X,X	; ERROR RESULT
19277	:	FPSB:	.WORD	X	; FPS BEFORE EXECUTION
19278	:	FPSA:	.WORD	X	; FPS AFTER EXECUTION
19279	:	FEC:	.WORD	X	; EXPECTED FEC
19280	:	FLAG:	.WORD	X	; 0/-1, OVER/UNDER FLOW FLAG
19281	:	ERR1:	ERROR	X	; TRAP ERROR.
19282	:	BR	CONT		
19283	:	ERR2:	ERROR	X	; DATA, RESULT ERROR
19284	:	CONT:			; RETURN ADDRESS

19285 :
 19286 ; THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR), THEN
 19287 ; THE MULD INSTRUCTION IS EXECUTED. IF NO TRAP OCCURS THEN THE
 19288 ; RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 19289 ; COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNDNT RETURNS CONTROL
 19290 ; TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNDNT
 19291 ; REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 19292 ; MULD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 19293 ; ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 19294 ; THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNDNT
 19295 ; WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
 19296 ; RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDNT WILL
 19297 ; REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
 19298 ; IF A TRAP OCCURS (IT SHOULD NOT) THEN OVUNDNT WILL READ THE FEC.
 19299 ; SHOULD THE FEC MATCH THE ANTICIPATED FEC OVUNDNT WILL
 19300 ; STORE ALL DATA AND TRANSFER CONTROL TO THE ERROR CALL AT ERR1. IF THE
 19301 ; FEC IS NOT THE SAME AS THE ANTICIPATED FEC OVUNDNT WILL REPORT
 19302 ; THE ERROR AND RETURN TO CONT. NOTE THAT OVUNDNT USES THE FLAG
 19303 ; TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
 19304 ; UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

19305					
19306	077524	012601			
19307	077526	012700	000200		
19308	077532	170100			
19309					
19310	077534	010100			
19311	077536	172410			
19312	077540	016100	000030		
19313	077544	170100			
19314	077546	012737	077626	000244	
19315					
19316	077554	010100			
19317	077556	062700	000010		
19318					
19319	077562	171010			
19320					
19321	077564	170204			
19322	077566	170305			
19323	077570	012700	000200		
19324	077574	170100			
19325	077576	012700	077646		
19326	077602	174010			
19327	077604	012700	077646		
19328	077610	010102			
19329	077612	062702	000020		
19330	077616	012703	000004		

OVUNDNT:	MOV	(SP)+,R1		; GET A POINTER TO THE ARGUMENTS.
	MOV	#200,R0		; SET FD MODE.
	LDFPS	R0		
	MOV	R1,R0		; LOAD ACO, OPERAND.
	LDD	(R0),ACO		
	MOV	30(R1),R0		; LOAD THE FPS.
	LDFPS	R0		
	MOV	#254,#FPVECT		; SET UP THE FP TRAP VECTOR IN CASE
				; OF ERROR.
	MOV	R1,R0		; COMPUTE THE ADDRESS OF FSRC.
	ADD	#10,R0		
14:	MULD	(R0),ACO		; TEST INSTRUCTION.
24:	STFPS	R4		; GET FPS.
	STST	R5		; GET FEC.
	MOV	#200,R0		; SET FD MODE.
	LDFPS	R0		
	MOV	#OVDNNT,R0		; GET THE RESULT.
	STD	ACO,(R0)		
	MOV	#OVDNNT,R0		; CHECK THE RESULT.
	MOV	R1,R2		
	ADD	#20,R2		
	MOV	#4,R3		

19331 077622 022022
 19332 077624 001401
 (1) 077626
 (2) 077626 104000
 19333 077630 077304
 19334
 19335 077632 026104 000032
 19336 077636 001401
 (2) 077640 104000
 19337 077642 000161 000040
 19338
 19339 077646 000000 000000 000000
 077654 000000
 19340
 19341 077656
 (1) 077656 004767 024670
 (1)
 (1)
 (1)
 (1)
 19342
 19343
 19344
 19356
 19357
 (2)
 (3)
 (2) 077662
 19358
 19359
 19360 077662 004737 100026
 19361 077666 020123 045676
 19362 077672 020200 000000
 19363 077676 000123 045676
 19364 077702 002000
 19365 077704 102004
 19366 077706 000012
 19367 077710 177777
 19368
 19369
 19370 077712 004737 100026
 19371 077716 010127 127272
 19372 077722 010200 000000
 19373 077726 060127 127272
 19374 077732 007017
 19375 077734 107000
 19376 077736 000012
 19377 077740 177777
 19378
 19379
 19380 077742 004737 100026
 19381 077746 060252 125252
 19382 077752 060000 000000
 19383 077756 000052 125252
 19384 077762 001000
 19385 077764 101006

34: CMP (R0), (R2).
 BEQ 54
 254: EMT
 54: SOB R3, 34 ;
 CMP 32(R1), R4 ; WAS FPS CORRECT?
 BEQ 44
 EMT ;
 44: JMP 40(R1) ; RETURN, TEST COMPLETED.
 OVDNTT: .WORD 0,0,0,0
 JJJDONE:
 JSR PC, .RSET ; GO INITIALIZE THE FPS AND STACK; AND
 ; SEE IF THE USER HAS EXPRESSED
 ; THE DESIRE TO CHANGE THE SOFTWARE
 ; VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ; THE USER TYPED CONTROL G?).

 ; TEST 503 UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST

 T5503:

; UNDERFLOW, EXPONENT OF RESULT = -129
 KKK1: JSR PC, @OVUNFT
 14: .WORD 20123,45676 ; AC
 24: .WORD 20200,0 ; FSRC
 34: .WORD 123,45676 ; RES
 54: 2000 ; FPS BEFORE EXECUTION.
 102004 ; FPS AFTER EXECUTION.
 64: 12 ; FEC
 -1 ; FLAG

; UNDERFLOW, EXPONENT OF THE RESULT = -193
 KKK3: JSR PC, @OVUNFT
 14: .WORD 10127,127272 ; AC
 24: .WORD 10200,0 ; FSRC
 34: .WORD 60127,127272 ; RES
 54: 7017 ; FPS BEFORE EXECUTION.
 107000 ; FPS AFTER EXECUTION.
 64: 12 ; FEC
 -1

; OVERFLOW, EXPONENT OF THE RESULT = 128
 KKK4: JSR PC, @OVUNFT
 14: .WORD 60252,125252 ; AC
 24: .WORD 60000,0 ; FSRC
 34: .WORD 000052,125252 ; RES
 54: 1000 ; FPS BEFORE EXECUTION.
 101006 ; FPS AFTER EXECUTION.

```

19386 077766 000010
19387 077770 000000
19388
19389
19390 077772 004737 100026
19391 077776 060345 067654
19392 100002 060200 000000
19393 100006 000345 067654
19394 100012 007015
19395 100014 107002
19396 100016 000010
19397 100020 000000
19398 100022 000167 000162
19399
19400
19401
19402
19403
19404
19405
19406
19407
19408
19409
19410
19411
19412
19413
19414
19415
19416
19417
19418
19419
19420
19421
19422
19423
19424
19425
19426
19427
19428
19429
19430
19431
19432
19433
19434
19435
19436 100026 012601
19437 100030 012700 000200
19438 100034 170100
19439 100036 010100
19440 100040 172410
19441 100042 016100 000014

```

```

68: 10 ;FEC
0 ;FLAG
;OVERFLOW, EXPONENT OF RESULT = 130
KKK5: JSR PC,@OVUNFT
18: .WORD 60345,67654 ;AC
28: .WORD 60200,0 ;FSRC
38: .WORD 345,67654 ;RES
58: 7015 ;FPS BEFORE EXECUTION.
107002 ;FPS AFTER EXECUTION.
68: 10 ;FEC
0 ;FLAG
88: JMP KKKDONE

```

```

;THIS SUBROUTINE, OVUNFT, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE MULF INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
;TO IT IS MADE THUS:

```

```

;
; ACARG: .WORD X,X ;AC OPERAND
; FSRCARG: .WORD X,X ;FSRC OPERAND
; RES: .WORD X,X ;EXPECTED RESULT
; ERRES: .WORD X,X ;ERROR RESULT
; FPSB: .WORD X ;FPS BEFORE EXECUTION
; FPSA: .WORD X ;FPS AFTER EXECUTION
; FEC: .WORD X ;EXPECTED FEC
; FLAG: .WORD X ;0/-1,OVER/UNDER FLOW FLAG
; ERR1: .ERROR X ;TRAP ERROR.
; BR CONT
; ERR2: .ERROR X ;DATA, RESULT ERROR
; CONT: ;RETURN ADDRESS

```

```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE MULF INSTRUCTION IS EXECUTED. IF THE TRAP OCCURS THEN THE
;RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNFT RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNFT
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. THE FEC IS TREATED
;IN THE SAME WAY. IF THE RESULT OF THE
;MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNFT
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNFT WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
;IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
;NOTE THAT OVUNFT USES THE FLAG
;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).

```

```

OVUNFT: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS.
MOV @200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO, OPERAND.
LDD (R0),ACO
MOV 14(R1),R0 ;LOAD THE FPS.

```

CJKL580 LCP-5 CPU CLSTR DIAG
CJML58 P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 26-17
T503 UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST

SEQ 0332

19442	100046	170100			LDFPS	R0	
19443	100050	012737	100072	0C0244	MOV	#50#,2#FPVECT	;SET UP THE FP TRAP VECTOR IN CASE
19444							;OF ERROR.
19445	100056	010100			MOV	R1,R0	;COMPUTE THE ADDRESS OF FSRC.
19446	100060	062700	000004		ADD	#4,R0	
19447							
19448	100064	171010			1#:	MULF	(R0),AC0 ;TEST INSTRUCTION. SHOULD CAUSE TRAP
19449	100066	170000			2#:	CFCC	
19450	100070	104000				EMT	
19451	100072	011602			50#:	MOV	(SP),R2 ;TRAP TO HERE AND SEE IF THE PC OF THE
19452	100074	020227	100066			CMP	R2,#2# ;TRAP WAS THAT OF THE MULF INSTRUCTION.
19453	100100	001401				BEQ	51#
(2)	100102	104000				EMT	
19454	100104	022626			51#:	CMP	(SP)+,(SP)+ ;RESET THE STACK
19455	100106	170204				STFPS	R4 ;GET FPS.
19456	100110	170305				STST	R5 ;GET FEC.
19457	100112	012700	000200			MOV	#200,R0 ;SET FD MODE.
19458	100116	170100				LDFPS	R0
19459	100120	12700	100200			MOV	#OVFTT,R0 ;GET THE RESULT.
19460	100124	174010				STD	AC0,(R0)
19461	100126	012700	100200			MOV	#OVFTT,R0 ;CHECK THE RESULT.
19462	100132	010102				MOV	R1,R2
19463	100134	062702	000010			ADD	#10,R2
19464	100140	012703	000002			MOV	#2,R3
19465	100144	022022			3#:	CMP	(R0)+,(R2)+
19466	100146	001401				BEQ	5#
(2)	100150	104000				EMT	
19467	100152	077304			5#:	SQB	R3,3#
19468							
19469	100154	026104	000016			CMP	16(R1),R4 ;WAS FPS CORRECT?
19470	100160	001401				BEQ	6#
(2)	100162	104000				EMT	
19471	100164	026105	000020		6#:	CMP	20(R1),R5 ;IS FEC CORRECT?
19472	100170	001401				BEQ	4#
(2)	100172	104000				EMT	
19473	100174	000161	000024		4#:	JMP	24(R1) ;RETURN, TEST COMPLETED.
19474							
19475	100200	000000	000000	000000	OVFTT:	.WORD	0,0,0,0
	100206	000000					
19476							
19477	100210				KKKDONE:		
(1)	100210	004767	024336			JSR	PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1)							;SEE IF THE USER HAS EXPRESSED
(1)							;THE DESIRE TO CHANGE THE SOFTWARE
(1)							;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)							;THE USER TYPED CONTROL G?).
19478							
19479							
19480							
19488							
19489							
(2)							
(3)							
(2)	100214						
19490							
19491							

```

;*****
;TEST 504 UNDER\OVER FLOW, USING MULF WITH TRAPS ENABLED, TEST
;*****
TS504:
;UNDERFLOW, EXPONENT OF RESULT = -129
    
```

```
19492 100214 004737 100440 LLL1: JSR PC, @OVUNDT
19493 100220 020052 125252 1#: .WORD 20052,125252 ;AC
19494 100224 125252 125252 .WORD 125252,125252
19495 100230 020300 000000 000000 2#: .WORD 20300,0,0,0 ;FSRC
100236 000000
19496 100240 000177 177777 177777 3#: .WORD 177, 1, 1, 1 ;RES
100246 177777
19497 100250 002200 5#: 2200 ;FPS BEFORE EXECUTION.
19498 100252 102204 102204 ;FPS AFTER EXECUTION.
19499 100254 000012 6#: 12 ;FEC
19500 100256 177777 -1 ;FLAG
19501
19502 ;UNDERFLOW, EXPONENT OF THE RESULT = -193
19503 100260 004737 100440 LLL2: JSR PC, @OVUNDT
19504 100264 010327 127272 1#: .WORD 10327,127272 ;AC
19505 100270 036363 045454 .WORD 36363,45454
19506 100274 010000 000000 000000 2#: .WORD 10000,0,0,0 ;FSRC
100302 000000
19507 100304 060127 127272 3#: .WORD 60127,127272 ;RES
19508 100310 036363 045454 .WORD 36363,45454
19509 100314 007217 5#: 7217 ;FPS BEFORE EXECUTION.
19510 100316 107200 107200 ;FPS AFTER EXECUTION.
19511 100320 000012 6#: 12 ;FEC
19512 100322 177777 -1 ;FLAG
19513
19514 ;OVERFLOW, EXPONENT OF THE RESULT = 128
19515 100324 004737 100440 LLL3: JSR PC, @OVUNDT
19516 100330 060252 125252 1#: .WORD 60252,125252 ;AC
19517 100334 125252 125252 .WORD 125252,125252 ;FSRC
19518 100340 160100 000000 000000 2#: .WORD 160100,0,0,0 ;FSRC
100346 000000
19519 100350 100177 177777 177777 3#: .WORD 100177,-1,-1,-1 ;RES
100356 177777
19520 100360 001200 5#: 1200 ;FPS BEFORE EXECUTION.
19521 100362 101216 101216 ;FPS AFTER EXECUTION.
19522 100364 000010 6#: 10 ;FEC
19523 100366 000000 0 ;FLAG
19524
19525 ;OVERFLOW, EXPONENT OF THE RESULT = 130
19526 100370 004737 100440 LLL4: JSR PC, @OVUNDT
19527 100374 060345 067654 1#: .WORD 60345,67654 ;AC
19528 100400 056765 045676 .WORD 56765,45676
19529 100404 060200 000000 000000 2#: .WORD 60200,0,0,0 ;FSRC
100412 000000
19530 100414 000345 067654 3#: .WORD 345,67654 ;RES
19531 100420 056765 045676 .WORD 56765,45676
19532 100424 007215 5#: 7215 ;FPS BEFORE EXECUTION.
19533 100426 107202 107202 ;FPS AFTER EXECUTION.
19534 100430 000010 6#: 10 ;FEC
19535 100432 000000 0 ;FLAG
19536 100434 000137 100622 8#: JMP @ALLDONE
19537
19538 ;THIS SUBROUTINE, OVUNDT, IS USED TO SET UP THE OPERANDS, EXECUTE
19539 ;THE MULD INSTRUCTION AND CHECK THE RESULTS OF AN INSTRUCTION WITH
19540 ;OPERANDS WHICH SHOULD RESULT IN EITHER OVERFLOW OR UNDERFLOW. A CALL
19541 ;TO IT IS MADE THUS:
```


19542	:				
19543	:	ACARG:	.WORD	X,X,X,X	;AC OPERAND
19544	:	FSRCARG:	.WORD	X,X,X,X	;FSRC OPERAND
19545	:	RES:	.WORD	X,X,X,X	;EXPECTED RESULT
19546	:	ERRES:	.WORD	X,X,X,X	;ERROR RESULT
19547	:	FPSB:	.WORD	X	;FPS BEFORE EXECUTION

```

19549      ;          FPSA:  .WORD  X          ;FPS AFTER EXECUTION
19550      ;          FEC:   .WORD  X          ;EXPECTED FEC
19551      ;          FLAG:  .WORD  X          ;0/-1,OVER/UNDER FLOW FLAG
19552      ;          ERR1:  ERROR X          ;TRAP ERROR.
19553      ;          BR    CONT
19554      ;          ERR2:  ERROR  X          ;DATA, RESULT ERROR
19555      ;          CONT:                ;RETURN ADDRESS
19556      ;
19557      ;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
19558      ;THE MULD INSTRUCTION IS EXECUTED. IF THE TRAP OCCURS THEN THE
19559      ;RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
19560      ;COMPARED WITH FPSA IF THIS TOO IS CORRECT OVUNDT RETURNS CONTROL
19561      ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD OVUNDT
19562      ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. THE FEC IS TREATED
19563      ;IN THE SAME WAY. IF THE RESULT OF THE
19564      ;MULF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
19565      ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
19566      ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN OVUNDT
19567      ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR2. OTHERWISE THE
19568      ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND OVUNDT WILL
19569      ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
19570      ;IF NO TRAP OCCURS CONTROL IS PASSED TO ERR1.
19571      ;NOTE THAT OVUNDT USES THE FLAG
19572      ;TO TELL WHETHER OR NOT THESE PARTICULAR OPERANDS WILL RESULT IN
19573      ;UNDERFLOW (FLAG=-1) OR OVERFLOW (FLAG=0).
19574
19575 100440 012601      OVUNDT: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
19576 100442 012700 0C0200  MOV      #20J,R0      ;SET FD MODE.
19577 100446 170100      LDFPS   R0
19578
19579 100450 010100      MOV      R1,R0      ;LOAD ACO, OPERAND.
19580 100452 172410      LDD      (R0),ACO
19581 100454 016100 000030  MOV      30(R1),R0      ;LOAD THE FPS.
19582 100460 170100      LDFPS   R0
19583 100462 012737 100504 000244  MOV      #50J,#FPVECT ;SET UP THE FP TRAP VECTOR IN CASE
19584                                     ;OF ERROR.
19585 100470 010100      MOV      R1,R0      ;COMPUTE THE ADDRESS OF FSRC.
19586 100472 062700 000010  ADD      #10,R0
19587
19588 100476 171010      14:    MULD   (R0),ACO      ;TEST INSTRUCTION. SHOULD CAUSE TRAP.
19589 100500 170000      24:    CFCC
19590 100502 104000      EMT
19591 100504 011602      504:   MOV      (SP),R2      ;TRAP TO HERE AND SEE IF THE PC OF THE
19592 100506 020227 100500  CMP      R2,#24      ;TRAP WAS THAT OF THE MULF INSTRUCTION.
19593 100512 001401      BEQ     514
19594 100514 104000      EMT      ;BRANCH IF YES.
19595 100516 022626      514:   CMP      (SP)+,(SP)+      ;RESET THE STACK
19596 100520 170204      STFPS  R4      ;GET FPS.
19597 100522 170305      STST  R5      ;GET FEC.
19598 100524 012700 000200  MOV      #200,R0      ;SET FD MODE.
19599 100530 170100      LDFPS  R0
19600 100532 012700 100612  MOV      #OVDTT,R0      ;GET THE RESULT.
19601 100536 174010      STD    ACO,(R0)
19602 100540 012700 100612  MOV      #OVDTT,R0      ;CHECK THE RESULT.
19603 100544 010102      MOV    R1,R2
19604 100546 062702 000020  ADD    #20,R2

```

19605 100552 012703 000004
 19606 100556 022022
 19607 100560 0014 J1
 (2) 100562 104 J00
 19608 100564 07 304
 19609 100566 026104 000032
 19610 100572 001401
 (2) 100574 104000
 19611 100576 026105 000034
 19612 100602 001401
 (2) 100604 104000
 19613 100606 000161 000040
 19614
 19615 100612 000000 000000 000000
 100620 000000
 19616
 19617 100622
 (1) 100622 004767 023724
 (1)
 (1)
 (1)
 (1)
 19618
 19619
 19620
 19621
 19629
 19630
 (2)
 (3)
 (2) 100626
 19631
 19632
 19633 100626 004737 101352
 19634 100632 000000 000000
 19635 100636 000000 000000
 19636 100642 000000 000000
 19637 100646 000000 000000
 19638 100652 000013
 19639 100654 000004
 19640
 19641
 19642 100656 004737 101352
 19643 100662 123456 076543
 19644 100666 000000 000000
 19645 100672 000000 000000
 19646 100676 000000 000000
 19647 100702 000000
 19648 100704 000004
 19649
 19650
 19651 100706 004737 101352
 19652 100712 000000 000000
 19653 100716 076543 021234
 19654 100722 000000 000000
 19655 100726 000000 000000

```

MOV      #4,R3
3$:     CMP      (R0),.(R2).
        BEQ      5$
        EMT
        ;
5$:     SOB      R3,3$
        CMP      32(R1),R4      ;WAS FPS CORRECT?
        BEQ      6$
        EMT
        ;
6$:     CMP      34(R1),R5      ;IS FEC CORRECT?
        BEQ      4$
        EMT
        ;
4$:     JMP      40(R1)         ;RETURN, TEST COMPLETED.

OVDTT:  .WORD    0,0,0,0

LLLLDONE:
        JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
        ;SEE IF THE USER HAS EXPRESSED
        ;THE DESIRE TO CHANGE THE SOFTWARE
        ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
        ;THE USER TYPED CONTROL G?).

;*****
;TEST 505      MODF TEST
;*****
TS505:

;MODF WITH (FSRC=AC=0)
GGG1:   JSR      PC,@MODFSUB
1$:     .WORD    0,0           ;AC
2$:     .WORD    0,0           ;FSRC
3$:     .WORD    0,0           ;FRACTIONAL RES.
4$:     .WORD    0,0           ;INTEGER RES.
7$:     13                ;FPS BEFORE EXECUTION.
        4                ;FPS AFTER EXECUTION.

;MODF TEST, WITH (FSRC=0)
GGG2:   JSR      PC,@MODFSUB
1$:     .WORD    123456,76543 ;AC
2$:     .WORD    0,0           ;FSRC
3$:     .WORD    0,0           ;FRACTIONAL RES.
4$:     .WORD    0,0           ;INTEGER RESULT.
7$:     0                ;FPS BEFORE EXECUTION.
        4                ;FPS AFTER EXECUTION.

;MODF TEST WITH (AC=0)
GGG3:   JSR      PC,@MODFSUB
1$:     .WORD    0,0           ;AC
2$:     .WORD    76543,21234  ;FSRC
3$:     .WORD    0,0           ;FRACTIONAL RES.
4$:     .WORD    0,0           ;INTEGER RES.

```


19712
 19713
 19714 101156 004737 101352
 19715 101162 042377 100000
 19716 101166 040200 000000
 19717 101172 000000 000000
 19718 101176 042377 100000
 19719 101202 000013
 19720 101204 000004
 19721
 19722
 19723 101206 004737 101352
 19724 101212 040177 177777
 19725 101216 040200 000000
 19726 101222 040177 177777
 19727 101226 000000 000000
 19728 101232 000017
 19729 101234 000000
 19730
 19731
 19732 101236 004737 101352
 19733 101242 034377 177777
 19734 101246 040200 000000
 19735 101252 034377 177777
 19736 101256 000000 000000
 19737 101262 000000
 19738 101264 000000
 19739
 19740
 19741 101266 004737 101352
 19742 101272 020000 000001
 19743 101276 040300 000000
 19744 101302 020100 000002
 19745 101306 000000 000000
 19746 101312 000000
 19747 101314 000000
 19748
 19749
 19750 101316 004737 101352
 19751 101322 142777 170000
 19752 101326 040200 000000
 19753 101332 140000 000000
 19754 101336 142777 160000
 19755 101342 000007
 19756 101344 000010
 19757 101346 000167 000204
 19758
 19759
 19760
 19761
 19762
 19763
 19764
 19765
 19766
 19767

```

;MODF TEST WITH EXPONENT OF THE RESULT = 9
GGG10: JSR PC, @MODFSUB
18: .WORD 42377,100000 ;AC
28: .WORD 40200,0 ;FSRC
38: .WORD 0,0 ;FRACTIONAL RES.
48: .WORD 42377,100000 ;INTEGER RES.
78: 13 ;FPS BEFORE EXECUTION.
4 ;FPS AFTER EXECUTION.
  
```

```

;MODF TEST WITH EXPONENT OF THE RESULT = 0
GGG11: JSR PC, @MODFSUB
18: .WORD 40177,-1 ;AC
28: .WORD 40200,0 ;FSRC
38: .WORD 40177,-1 ;FRACTIONAL RES.
48: .WORD 0,0 ;INTEGER RES.
78: 17 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
  
```

```

;MODF TEST WITH EXPONENT OF THE RESULT = -15
GGG12: JSR PC, @MODFSUB
18: .WORD 34377,-1 ;AC
28: .WORD 40200,0 ;FSRC
38: .WORD 34377,-1 ;FRACTIONAL RES.
48: .WORD 0,0 ;INTEGER RES.
78: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
  
```

```

;MODF TEST WITH EXPONENT OF RESULT = -64, IN ROUND MODE
GGG13: JSR PC, @MODFSUB
18: .WORD 20000,1 ;AC
28: .WORD 40300,0 ;FSRC
38: .WORD 20100,2 ;FRACTIONAL RES.
48: .WORD 0,0 ;INTEGER RES.
78: 0 ;FPS BEFORE EXECUTION.
0 ;FPS AFTER EXECUTION.
  
```

```

;MODF TEST WITH EXPONENT OF RESULT = 11
GGG14: JSR PC, @MODFSUB
18: .WORD 142777,170000 ;AC
28: .WORD 40200,0 ;FSRC
38: .WORD 140000,0 ;FRACTIONAL RES.
48: .WORD 142777,160000 ;INTEGER RES.
78: 7 ;FPS BEFORE EXECUTION.
10 ;FPS AFTER EXECUTION.
98: JMP GGGDONE ;GO TO NEXT TEST.
  
```

```

;THIS SUBROUTINE, MODFSUB, IS CALLED TO SETUP THE
;OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
;IT IS CALLED THUS:
  
```

```

;
; ACARG: .WORD X,X ;AC OPERAND
; FSRCARG: .WORD X,X ;FSRC OPERAND
; FRES: .WORD X,X ;FRACTIONAL RESULT
; INTRES: .WORD X,X ;INTEGER RESULT
  
```

19768	:	ERFRES: .WORD	X,X	;ERROR FRACTION RESULT
19769	:	ERINTRES: .WORD	X,X	;ERROR INTEGER RESULT
19770	:	FPSB: .WORD	X	;FPS BEFORE EXECUTION
19771	:	FPSA: .WORD	X	;FPS AFTER EXECUTION
19772	:	ERR1: ERROR	X	;FRACTION ERROR
19773	:	BR	CONT	
19774	:	ERR2: ERROR	X	;INTEGER ERROR
19775	:	CONT:		;RETURN ADDRESS

; THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
 ; INSTRUCTION IS EXECUTED. THEN THE RESULT ARE RETRIEVED.
 ; THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
 ; THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
 ; THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
 ; THEN MODFSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
 ; IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
 ; THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
 ; THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
 ; FAILURE MATCHES THE TRUE RESULT THEN MODFSUB PASSES CONTROL TO THE
 ; ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS
 ; NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
 ; FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
 ; IF A MATCH IS MADE HOWEVER, MODFSUB WILL RETURN CONTROL TO THE ERROR
 ; CALL AT ERR2.

19793	101352	012601		MODFSUB:	MOV	(SP),R1		;GET A POINTER TO THE ARGUMENTS
19794	101354	012700	000200		MOV	#200,R0		;SET FD MODE.
19795	101360	170100			LDFPS	R0		
19796	101362	010100			MOV	R1,R0		;SET UP ACO
19797	101364	172410			LDD	(R0),ACO		
19798	101366	012700	101546		MOV	#MODP1,R0		;PUT A BACKGROUND PATTERN INTO AC1.
19799	101372	172510			LDD	(R0),AC1		
19800	101374	016100	000020		MOV	20(R1),R0		;SET UP THE FPS.
19801	101400	170100			LDFPS	R0		
19802	101402	010100			MOV	R1,R0		;COMPUTE THE ADDRESS OF THE FSRC.
19803	101404	062700	000004		ADD	#4,R0		
19804								
19805	101410	171410		11:	MODF	(R0),ACO		;EXECUTE THE TEST INSTRUCTION.
19806								
19807	101412	170204			STFPS	R4		;GET THE FPS.
19808	101414	012700	000200		MOV	#200,R0		;SET FD MODE.
19809	101420	170100			LDFPS	R0		
19810	101422	012700	101526		MOV	#MODFT0,R0		;GET THE FRACTIONAL RESULT.
19811	101426	174010			STD	ACO,(R0)		
19812	101430	012700	101536		MOV	#MODFT1,R0		;GET THE INTEGER RESULT.
19813	101434	174110			STD	AC1,(R0)		
19814	101436	012702	101526		MOV	#MODFT0,R2		;CHECK THE FRACTIONAL RESULT.
19815	101442	026112	000010		CMP	10(R1),(R2)		
19816	101446	001401			BEQ	21		
(2)	101450	104000			EMT			
19817	101452	026162	000012	000002	21:	CMP	12(R1),2(R2)	
19818	101460	001401			BEQ	31		
(2)	101462	104000			EMT			
19819	101464	012702	101536		31:	MOV	#MODFT1,R2	;CHECK THE INTEGER RESULT.
19820	101470	026112	000014		CMP	14(R1),(R2)		
19821	101474	001401			BEQ	41		

```

(2) 101476 104000
19822 101500 026162 000016 000002 48: EMT
CMP 16(R1),2(R2)
19823 101506 001401 BEQ 58
(2) 101510 104000 EMT
19824 101512 026104 000022 58: CMP 22(R1),R4 ;CHECK THE FPS.
19825 101516 001401 BEQ 98
(2) 101520 104000 EMT
19826 101522 000161 000024 98: JMP 24(R1) ;RETURN.
19827
19828 101526 000000 000000 000000 MODFT0: .WORD 0,0,0,0
101534 000000
19829
19830 101536 000000 000000 000000 MODFT1: .WORD 0,0,0,0
101544 000000
19831
19832 101546 177777 177777 177777 MODP1: .WORD -1,-1,-1,-1
101554 177777
19833
19834 101556 GGGDONE:
(1) 101556 004767 022770 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
19835
19836
19837
19845
19846 ;*****
(2) ;TEST 506 MODD TEST
(3) ;*****
(2) 101562 TS506:
19847
19848 ;MODD WITH (FSRC=AC=0)
19849 101562 004737 102576 HMM1: JSR PC,@MODDSUB
19850 101566 000000 000000 000000 18: .WORD 0,0,0,0 ;AC
101574 000000
19851 101576 000000 000000 000000 28: .WORD 0,0,0,0 ;FSRC
101604 000000
19852 101606 000000 000000 000000 38: .WORD 0,0,0,0 ;FRACTIONAL RES.
101614 000000
19853 101616 000000 000000 000000 48: .WORD 0,0,0,0 ;INTEGER RES.
101624 000000
19854 101626 000200 8: 200 ;FPS BEFORE EXECUTION.
19855 101630 000204 204 ;FPS AFTER EXECUTION.
19856
19857 ;MODD TEST WITH FSRC=0
19858 101632 004737 102576 HMM2: JSR PC,@MODDSUB
19859 101636 012345 067012 18: .WORD 012345,67012 ;AC
19860 101642 034567 012345 .WORD 34567,012345
19861 101646 000000 000000 000000 28: .WORD 0,0,0,0 ;FSRC
101654 000000
19862 101656 000000 000000 000000 38: .WORD 0,0,0,0 ;FRACTIONAL RES.
101664 000000
19863 101666 000000 000000 000000 48: .WORD 0,0,0,0 ;INTEGER RES.
101674 000000

```


19908	102142	004737	102576		MMH7:	JSR	PC,8#MODDSUB	
19909	102146	056000	000000	000000	1#:	.WORD	56000,0,0,1	;AC
	102154	000001						
19910	102156	040340	000000	000000	2#:	.WORD	40340,0,0,0	;FSRC
	102164	000000						
19911	102166	040100	000000	000000	3#:	.WORD	40100,0,0,0	;FRACTIONAL RES.
	102174	000000						
19912	102176	056140	000000	000000	4#:	.WORD	56140,0,0,1	;INTEGER RES.
	102204	000001						
19913	102206	000213			7#:		213	;FPS BEFORE EXECUTION.
19914	102210	000200					200	;FPS AFTER EXECUTION.
19915								
19916								;MODD TEST WITH EXPONENT OF THE RESULT = 36
19917	102212	004737	102576		MMH8:	JSR	PC,8#MODDSUB	
19918	102216	051177	177777	177777	1#:	.WORD	51177,-1,-1,-1	;AC
	102224	177777						
19919	102226	040200	000000	000000	2#:	.WORD	40200,0,0,0	;FSRC
	102234	000000						
19920	102236	040177	177760	000000	3#:	.WORD	40177,-20,0,0	;FRACTIONAL RES.
	102244	000000						
19921	102246	051177	177777	177760	4#:	.WORD	51177,-1,-20,0	;INTEGER RES.
	102254	000000						
19922	102256	000217			7#:		217	;FPS BEFORE EXECUTION.
19923	102260	000200					200	;FPS AFTER EXECUTION.
19924								
19925								;MODD TEST WITH EXPONENT OF THE RESULT = 30
19926	102262	004737	102576		MMH9:	JSR	PC,8#MODDSUB	
19927	102266	040200	000000	000000	1#:	.WORD	40200,0,0,0	;AC
	102274	000000						
19928	102276	047577	177777		2#:	.WORD	47577,-1	;FSRC
19929	102302	176000	000001			.WORD	176000,1	
19930	102306	031600	000000	000000	3#:	.WORD	31600,0,0,0	;FRACTIONAL RES.
	102314	000000						
19931	102316	047577	177777		4#:	.WORD	47577,-1	;INTEGER RES.
19932	102322	176000	000000			.WORD	176000,0	
19933	102326	000200			7#:		200	;FPS BEFORE EXECUTION.
19934	102330	000200					200	;FPS AFTER EXECUTION.
19935								
19936								;MODD TEST WITH EXPONENT OF THE RESULT = 31
19937	102332	004737	102576		MMH10:	JSR	PC,8#MODDSUB	
19938	102336	047777	177777		1#:	.WORD	47777,-1	;AC
19939	102342	177000	000000			.WORD	177000,0	
19940	102346	040200	000000	000000	2#:	.WORD	40200,0,0,0	;FSRC
	102354	000000						
19941	102356	000000	000000	000000	3#:	.WORD	0,0,0,0	;FRACTIONAL RES.
	102364	000000						
19942	102366	047777	177777		4#:	.WORD	47777,-1	;INTEGER RES.
19943	102372	177000	000000			.WORD	177000,0	
19944	102376	000213			7#:		213	;FPS BEFORE EXECUTION.
19945	102400	000204					204	;FPS AFTER EXECUTION.
19946								
19947								;MODD TEST WITH EXPONENT OF THE RESULT = 0
19948	102402	004737	102576		MMH11:	JSR	PC,8#MODDSUB	
19949	102406	040200	000000	000000	1#:	.WORD	40200,0,0,0	;AC
	102414	000000						
19950	102416	040177	072727		2#:	.WORD	40177,72727	;FSRC

```

19951 102422 127272 072727          .WORD 127272,72727
19952 102426 040177 072727          3#: .WORD 40177,72727 ;FRACTIONAL RES.
19953 102432 127272 072727          .WORD 127272,72727
19954 102436 000000 000000 000000 4#: .WORD 0,0,0,0 ;INTEGER RES.
      102444 000000
19955 102446 000200          7#: 200 ;FPS BEFORE EXECUTION.
19956 102450 000200          200 ;FPS AFTER EXECUTION.
19957
19958 ;MODD TEST WITH EXPONENT OF THE RESULT = -115
19959 102452 004737 102576 MM12: JSR PC, @MODDSUB
19960 102456 003377 177777          1#: .WORD 3377,-1 ;AC
19961 102462 177777 052525          .WORD -1,52525
19962 102466 040200 000000 000000 2#: .WORD 40200,0,0,0 ;FSRC
      102474 000000
19963 102476 003377 177777          3#: .WORD 3377,-1 ;FRACTIONAL RES.
19964 102502 177777 052525          .WORD -1,52525
19965 102506 000000 000000 000000 4#: .WORD 0,0,0,0 ;INTEGER RES.
      102514 000000
19966 102516 000200          7#: 200 ;FPS BEFORE EXECUTION.
19967 102520 000200          200 ;FPS AFTER EXECUTION.
19968
19969 ;MODD TEST WITH EXPONENT OF THE RESULT = -63, IN ROUND MODE.
19970 102522 004737 102576 MM13: JSR PC, @MODDSUB
19971 102526 040300 000000 000000 1#: .WORD 40300,0,0,0 ;AC
      102534 000000
19972 102536 020200 000000 000000 2#: .WORD 20200,0,0,1 ;FSRC
      102544 000001
19973 102546 020300 000000 000000 3#: .WORD 20300,0,0,2 ;FRACTIONAL RES.
      102554 000002
19974 102556 000000 000000 000000 4#: .WORD 0,0,0,0 ;INTEGER RES.
      102564 000000
19975 102566 000200          7#: 200 ;FPS BEFORE EXECUTION.
19976 102570 000200          200 ;FPS AFTER EXECUTION.
19977 102572 000137 102772          9#: JMP @MMHNDONE ;GO TO THE NEXT TEST.
19978
19979 ;THIS SUBROUTINE, MODDSUB, IS CALLED TO SETUP THE
19980 ;OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
19981 ;IT IS CALLED THUS:
19982 ;
19983 ; ACARG: .WORD X,X,X,X ;AC OPERAND
19984 ; FSRCARG: .WORD X,X,X,X ;FSRC OPERAND
19985 ; FRES: .WORD X,X,X,X ;FRACTIONAL RESULT
19986 ; INTRES: .WORD X,X,X,X ;INTEGER RESULT
19987 ; ERFRES: .WORD X,X,X,X ;ERROR FRACTION RESULT
19988 ; ERINTRES: .WORD X,X,X,X ;ERROR INTEGER RESULT
19989 ; FPSB: .WORD X ;FPS BEFORE EXECUTION
19990 ; FPSA: .WORD X ;FPS AFTER EXECUTION
19991 ; ERR1: ERROR X ;FRACTION ERROR
19992 ; BR CONT
19993 ; ERR2: ERROR X ;INTEGER ERROR
19994 ; CONT: ;RETURN ADDRESS
19995 ;
19996 ;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
19997 ;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
19998 ;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
19999 ;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT

```

```

; THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
; THEN MODDSUB WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
; IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
; THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
; THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
; FAILURE MATCHES THE TRUE RESULT THEN MODDSUB PASSES CONTROL TO THE
; ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS
; NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
; FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
; IF A MATCH IS MADE HOWEVER, MODDSUB WILL RETURN CONTROL TO THE ERROR
; CALL AT ERR2.
  
```

```

20000
20001
20002
20003
20004
20005
20006
20007
20008
20009
20010
20011
20012 102576 012601
20013 102600 012700 000200
20014 102604 170100
20015 102606 010100
20016 102610 172410
20017 102612 012700 101546
20018 102616 172510
20019 102620 016100 000040
20020 102624 170100
20021 102626 010100
20022 102630 062700 000010
20023
20024 102634 171410
20025
20026 102636 170204
20027 102640 012700 000200
20028 102644 170100
20029 102646 012700 102752
20030 102652 174010
20031 102654 012700 102762
20032 102660 174110
20033 102662 012702 102752
20034 102666 010103
20035 102670 062703 000020
20036 102674 012705 000004
20037 102700 022223
20038 102702 001401
(2) 102704 104000
20039 102706 077504
20040 102710 012702 102762
20041 102714 010103
20042 102716 062703 000030
20043 102722 012705 000004
20044 102726 022223
20045 102730 001401
(2) 102732 104000
20046 102734 077504
20047 102736 026104 000042
20048 102742 001401
(2) 102744 104000
20049 102746 000161 000044
20050
20051 102752 000000 000000 000000 MODDT0: .WORD 0,0,0,0
102760 000000
  
```

```

MODDSUB:      MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS
              MOV      #200,R0      ;SET FD MODE.
              LDFPS    R0
              MOV      R1,R0        ;SET UP ACO
              LDD      (R0),ACO
              MOV      #MODP1,R0     ;PUT A BACKGROUND PATTERN INTO AC1.
              LDD      (R0),AC1
              MOV      40(R1),R0     ;SET UP THE FPS.
              LDFPS    R0
              MOV      R1,R0        ;COMPUTE THE ADDRESS OF THE FSRC.
              ADD      #10,R0
14:           MODD      (R0),ACO     ;EXECUTE THE TEST INSTRUCTION.
              STFPS    R4           ;GET THE FPS.
              MOV      #200,R0      ;SET FD MODE.
              LDFPS    R0
              MOV      #MODDT0,R0   ;GET THE FRACTIONAL RESULT.
              STD      ACO,(R0)
              MOV      #MODDT1,R0   ;GET THE INTEGER RESULT.
              STD      AC1,(R0)
              MOV      #MODDT0,R2   ;CHECK THE FRACTIONAL RESULT.
              MOV      R1,R3
              ADD      #20,R3
              MOV      #4,R5
24:           CMP      (R2)+,(R3)+
              BEQ      44
              EMT
44:           SOB      R5,24
              MOV      #MODDT1,R2   ;CHECK THE INTEGER RESULT.
              MOV      R1,R3
              ADD      #30,R3
              MOV      #4,R5
34:           CMP      (R2)+,(R3)+
              BEQ      54
              EMT
54:           SOB      R5,34
              CMP      42(R1),R4     ;CHECK THE FPS.
              BEQ      94
              EMT
94:           JMP      44(R1)       ;RETURN.
  
```

20052
20053 102762 000000 000000 000000
102770 000000
20054
20055 102772
(1) 102772 004767 021554
(1)
(1)
(1)
(1)
20056
20057
20058
20066
20067
(2)
(3)
(2) 102776
20068
20069
20070 102776 004767 000214
20071 103002 020123 045676
20072 103006 020200 000000
20073 103012 000123 045676
20074 103016 000000 000000
20075 103022 042000
20076 103024 142004
20077 103026 000012
20078 103030 104000
20079
20080
20081 103032 004737 103216
20082 103036 010200 000000
20083 103042 010000 000000
20084 103046 000000 000000
20085 103052 000000 000000
20086 103056 005013
20087 103060 005004
20088 103062 000012
20089 103064 000240
20090
20091
20092 103066 004737 103216
20093 103072 060052 125252
20094 103076 060200 000000
20095 103102 000000 000000
20096 103106 000052 125252
20097 103112 041000
20098 103114 141006
20099 103116 000010
20100 103120
(2) 103120 104000
20101
20102 103122 004737 103216
20103 103126 060345 067654
20104 103132 060200 000000

MODD1: .WORD 0,0,0,0

MMHDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 507 UNDER/OVER FLOW, USING MODF WITH TRAPS DISABLED, TEST
;*****
T5507:

;UNDERFLOW TEST, WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1
MMH1: JSR PC,MODFOV
1#: .WORD 20123,45676 ;AC
2#: .WORD 20200,0 ;FSRC
3#: .WORD 123,45676 ;FRACTIONAL RES.
4#: .WORD 0,0 ;INTEGER RES.
7#: 42000 ;FPS BEFORE EXECUTION.
142004 ;FPS AFTER EXECUTION.
12 ;FEC
EMT ;

;UNDERFLOW EXP OF RESULT = -193, FIU = 0, FID = 1
MMH2: JSR PC,MMMODFOV
1#: .WORD 10200,0 ;AC
2#: .WORD 10000,0 ;FSRC
3#: .WORD 0,0 ;FRACTIONAL RES.
4#: .WORD 0,0 ;INTEGER RES.
7#: 5013 ;FPS BEFORE EXECUTION.
5004 ;FPS AFTER EXECUTION.
12 ;FEC
NOP

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1
MMH3: JSR PC,MMMODFOV
1#: .WORD 60052,125252 ;AC
2#: .WORD 60200,0 ;FSRC
3#: .WORD 0,0 ;FRACTIONAL RES.
4#: .WORD 52,125252 ;INTEGER RES.
7#: 41000 ;FPS BEFORE EXECUTION.
141006 ;FPS AFTER EXECUTION.
10 ;FEC
EMT ;

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1
MMH4: JSR PC,MMMODFOV
1#: .WORD 60345,67654 ;AC
2#: .WORD 60200,0 ;FSRC

20105 103136 000000 000000
 20106 103142 000000 000000
 20107 103146 006011
 20108 103150 006006
 20109 103152 000010
 20110 103154 000240
 20111
 20112
 20113 103156 004737 103216
 20114 103162 160252 125252
 20115 103166 060000 000000
 20116 103172 000000 000000
 20117 103176 100052 125252
 20118 103202 041000
 20119 103204 141006
 20120 103206 000010
 20121 103210
 (2) 103210 104000
 20122 103212 000137 103432
 20123
 20124
 20125
 20126
 20127
 20128
 20129
 20130
 20131
 20132
 20133
 20134
 20135
 20136
 20137
 20138
 20139
 20140
 20141
 20142
 20143
 20144
 20145
 20146
 20147
 20148
 20149
 20150
 20151
 20152
 20153
 20154
 20155
 20156
 20157
 20158 103216 012601
 20159 103220 012700 000200

```

3#: .WORD 0,0 ;FRACTIONAL RES.
4#: .WORD 0,0 ;INTEGER RES.
7#: 6011 ;FPS BEFORE EXECUTION
    6006 ;FPS AFTER EXECUTION.
    10 ;FEC
8#: NOP
;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, RESULT NEGATIVE
;AND FIV = 1, FID = 1
MMMS: JSR PC, @MODFOV
1#: .WORD 160252,125252 ;AC
2#: .WORD 60000,0 ;FSRC
3#: .WORD 0,0 ;FRACTIONAL RES.
4#: .WORD 100052,125252 ;INTEGER RES.
7#: 41000 ;FPS BEFORE EXECUTION.
    141006 ;FPS AFTER EXECUTION.
    10 ;FEC
8#:
9#: EMT ;
    JMP @MMMDONE ;GO TO THE NEXT TEST.
  
```

;THIS SUBROUTINE, MODFOV, IS CALLED TO SETUP THE
 ;OPERANDS, EXECUTE THE MODF INSTRUCTION AND CHECK THE RESULTS.
 ;IT IS CALLED THIS:

```

:
: ACARG: .WORD X,X ;AC OPERAND
: FSRCARG: .WORD X,X ;FSRC OPERAND
: FRES: .WORD X,X ;FRACTIONAL RESULT
: INTRES: .WORD X,X ;INTEGER RESULT
: ERFRES: .WORD X,X ;ERROR FRACTION RESULT
: ERINTRES: .WORD X,X ;ERROR INTEGER RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;FEC
: ERR1: ERROR X ;FEC ERROR
: BR CONT
: ERR2: ERROR X ;INTEGER ERROR
: CONT: ;RETURN ADDRESS
  
```

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODF
 ;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
 ;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
 ;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
 ;THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
 ;THEN MODFOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
 ;IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
 ;THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
 ;THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
 ;FAILURE MATCHES THE TRUE RESULT THEN MODFOV PASSES CONTROL TO THE
 ;ERROR CALL AT ERR1. LIKEWISE IF THE INTEGER PART OF THE RESULT IS
 ;NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
 ;FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
 ;IF A MATCH IS MADE HOWEVER, MODFOV WILL RETURN CONTROL TO THE ERROR
 ;CALL AT ERR2.

```

MODFOV: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
        MOV #200,R0 ;SET FD MODE.
  
```

20160	103224	170100				LDFPS	R0		
20161	103226	010100				MOV	R1,R0	;SET UP ACO	
20162	103230	172410				LDD	(R0),ACO		
20163	103232	012700	101546			MOV	#MODP1,R0	;PUT A BACKGROUND PATTERN INTO AC1.	
20164	103236	172510				LDD	(R0),AC1		
20165	103240	016100	000020			MOV	20(R1),R0	;SET UP THE FPS.	
20166	103244	170100				LDFPS	R0		
20167	103246	010100				MOV	R1,R0	;COMPUTE THE ADDRESS OF THE FSRC.	
20168	103250	062700	000004			ADD	#4,R0		
20169									
20170	103254	171410			1#:	MODF	(R0),ACO	;EXECUTE THE TEST INSTRUCTION.	
20171									
20172	103256	170204				STFPS	R4	;GET THE FPS.	
20173	103260	170305				STST	R5	;GET FEC.	
20174	103262	012700	000200			MOV	#200,R0	;SET FD MODE.	
20175	103266	170100				LDFPS	R0		
20176	103270	012700	103412			MOV	#MODFDO,R0	;GET THE FRACTIONAL RESULT.	
20177	103274	174010				STD	ACO,(R0)		
20178	103276	012700	103422			MOV	#MODFD1,R0	;GET THE INTEGER RESULT.	
20179	103302	174110				STD	AC1,(R0)		
20180	103304	012702	103412			MOV	#MODFDO,R2	;CHECK THE FRACTIONAL RESULT.	
20181	103310	026112	000010			CMP	10(R1),(R2)		
20182	103314	001401				BEQ	2#		
(2)	103316	104000				EMT			
20183	103320	026162	000012	000002	2#:	CMP	12(R1),2(R2)		
20184	103326	001401				BEQ	3#		
(2)	103330	104000				EMT			
20185	103332	012702	103422		3#:	MOV	#MODFD1,R2	;CHECK THE INTEGER RESULT.	
20186	103336	026112	000014			CMP	14(R1),(R2)		
20187	103342	001401				BEQ	4#		
(2)	103344	104000				EMT			
20188	103346	026162	000016	000002	4#:	CMP	16(R1),2(R2)		
20189	103354	001401				BEQ	5#		
(2)	103356	104000				EMT			
20190	103360	026104	000022		5#:	CMP	22(R1),R4	;CHECK THE FPS.	
20191	103364	001401				BEQ	6#		
(2)	103366	104000				EMT			
20192	103370	026105	000024		6#:	CMP	24(R1),R5	;CHECK THE FEC.	
20193	103374	001002				BNE	25#	;BRANCH IF INCORRECT.	
20194									
20195	103376	000161	000030		9#:	JMP	30(R1)	;RETURN.	
20196					;REPORT	FEC ERROR.			
20197	103402	010102			25#:	MOV	R1,R2		
20198	103404	062702	000026			ADD	#26,R2		
20199	103410	000112				JMP	(R2)		
20200									
20201	103412	000000	000000	000000	MODFDO:	.WORD	0,0,0,0		
	103420	000000							
20202									
20203	103422	000000	000000	000000	MODFD1:	.WORD	0,0,0,0		
	103430	000000							
20204									
20205	103432				MMIDONE:				
(1)	103432	004767	021114			JSR	PC,.RSET	;GO INITIALIZE THE FPS AND STACK; AND	
(1)								;SEE IF THE USER HAS EXPRESSED	
(1)								;THE DESIRE TO CHANGE THE SOFTWARE	

;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

(1)
(1)
20206
20207
20208
20216
20217
(2)
(3)
(2) 103436
20218
20219
20220 103436 004737 103722
20221 103442 020252 125252
20222 103446 125252 125252
20223 103452 020100 000000 000000
103460 000000
20224 103462 000177 177777 177777
103470 177777
20225 103472 000000 000000 000000
103500 000000
20226 103502 042200
20227 103504 142204
20228 103506 000012
20229 103510
(2) 103510 104000
20230
20231 103512 004737 103722
20232 103516 010000 000000
20233 103522 123456 000000
20234 103526 010200 000000 000000
103534 000000
20235 103536 000000 000000 000000
103544 000000
20236 103546 000000 000000 000000
103554 000000
20237 103556 005213
20238 103560 005204
20239 103562 000012
20240 103564 000240
20241
20242 103566 004737 103722
20243 103572 060252 125252
20244 103576 125252 125252
20245 103602 060100 000000 000000
103610 000000
20246 103612 000000 000000 000000
103620 000000
20247 103622 000177 177777 177777
103630 177777
20248 103632 041200
20249 103634 141206
20250 103636 000010
20251 103640
(2) 103640 104000
20252

;*****
;TEST 510 UNDER\OVER FLOW, USING MODF WITH TRAPS DISABLED, TEST
;*****
TS510:

;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -129, FIU = 1, FID = 1

NNN1: JSR PC,B#MODDOV

1#: .WORD 20252,125252 ;AC

.WORD 125252,125252

2#: .WORD 20100,0,0,0 ;FSRC

3#: .WORD 177,-1,-1,-1 ;FRACTIONAL RES.

4#: .WORD 0,0,0,0 ;INTEGER RES.

7#: 42200 ;FPS BEFORE EXECUTION.

142204 ;FPS AFTER EXECUTION.

12 ;FEC

8#: EMT ;

;UNDERFLOW TEST WITH EXPONENT OF THE RESULT = -193, FIU = 0, FID = 1

NNN2: JSR PC,B#MODDOV

1#: .WORD 10000,0 ;AC

.WORD 123456,0

2#: .WORD 10200,0,0,0 ;FSRC

3#: .WORD 0,0,0,0 ;FRACTIONAL RES.

4#: .WORD 0,0,0,0 ;INTEGER RES.

7#: 5213 ;FPS BEFORE EXECUTION.

5204 ;FPS AFTER EXECUTION.

12 ;FEC

8#: NOP ;

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 128, FIV = 1, FID = 1

NNN3: JSR PC,B#MODDOV

1#: .WORD 60252,125252 ;AC

.WORD 125252,125252

2#: .WORD 60100,0,0,0 ;FSRC

3#: .WORD 0,0,0,0 ;FRACTIONAL RES.

4#: .WORD 177,-1,-1,-1 ;INTEGER RES.

7#: 41200 ;FPS BEFORE EXECUTION.

141206 ;FPS AFTER EXECUTION.

10 ;FEC

8#: EMT ;

;OVERFLOW TEST WITH EXPONENT OF THE RESULT = 130, FIV = 0, FID = 1

20253 103642 004737 103722
20254 103646 060200 000000
20255 103652 125252 000000
20256 103656 060200 000000 000000
103664 000000
20257 103666 000000 000000 000000
103674 000000
20258 103676 000000 000000 000000
103704 000000
20259 103706 006211
20260 103710 006206
20261 103712 000010
20262 103714 000240
20263 103716 000137 104136
20264
20265
20266
20267
20268
20269
20270
20271
20272
20273
20274
20275
20276
20277
20278
20279
20280
20281
20282
20283
20284
20285
20286
20287
20288
20289
20290
20291
20292
20293
20294
20295
20296
20297
20298 103722 012601
20299 103724 012700 000200
20300 103730 170100
20301 103732 010100
20302 103734 172410
20303 103736 012700 101546
20304 103742 172510
20305 103744 016100 000040

NNN4: JSR PC, @MODDOV
1#: .WORD 60200,0 ;AC
.WORD 125252,0
2#: .WORD 60200,0,0,0 ;FSRC
3#: .WORD 0,0,0,0 ;FRACTIONAL RES.
4#: .WORD 0,0,0,0 ;INTEGER RES.
7#: 6211 ;FPS BEFORE EXECUTION.
6206 ;FPS AFTER EXECUTION.
10 ;FEC
8#: NOP
9#: JMP @NNNDONE ;GO TO NEXT TEST.

;THIS SUBROUTINE, MODDOV, IS CALLED TO SETUP THE
;OPERANDS, EXECUTE THE MODD INSTRUCTION AND CHECK THE RESULTS.
;IT IS CALLED THUS:

```

:
:          ACARG: .WORD X,X,X,X          ;AC OPERAND
:          FSRCARG: .WORD X,X,X,X        ;FSRC OPERAND
:          FRES: .WORD X,X,X,X          ;FRACTIONAL RESULT
:          INTRES: .WORD X,X,X,X        ;INTEGER RESULT
:          ERFRES: .WORD X,X,X,X        ;ERROR FRACTION RESULT
:          ERINTRES: .WORD X,X,X,X      ;ERROR INTEGER RESULT
:          FPSB: .WORD X                 ;FPS BEFORE EXECUTION
:          FPSA: .WORD X                 ;FPS AFTER EXECUTION
:          ERR1: ERROR X                 ;FRACTION ERROR
:          BR CONT                       ;
:          ERR2: ERROR X                 ;INTEGER ERROR
:          CONT:                          ;RETURN ADDRESS
:

```

;THE OPERANDS ARE SET UP (USING ACO FOR THE AC ARGUMENT). THE MODD
;INSTRUCTION IS EXECUTED. THEN THE RESULTS ARE RETRIEVED.
;THE FRACTION PART OF THE RESULT IS COMPARED WITH FRES. IF THIS IS CORRECT
;THEN THE INTEGER PART IS COMPARED WITH INTRES. IF BOTH OF THESE ARE CORRECT
;THEN THE FPS IS COMPARED WITH FPSA. AFTER EXECUTION IF NO ERRORS OCCURRED
;THEN MODDOV WILL RETURN TO CONT. IF THE FPS WAS INCORRECT
;IT IS REPORTED HERE. IF THE FRACTION IS INCORRECT IT IS COMPARED WITH
;THE ANTICIPATED BAD FRACTION, ERFRES. IF THIS DOESN'T MATCH
;THE TRUE RESULT THEN THE ERROR IS REPORTED HERE. IF THE ANTICIPATED
;FAILURE MATCHES THE TRUE RESULT THEN MODDOV PASSES CONTROL TO THE
;ERROR CALL AT ERR1. LIKewise IF THE INTEGER PART OF THE RESULT IS
;NOT CORRECT THEN IT IS COMPARED WITH THE ANTICIPATED INTEGER
;FAILURE. IF THIS DOESN'T MATCH THEN THE ERROR IS REPORTED HERE.
;IF A MATCH IS MADE HOWEVER, MODDOV WILL RETURN CONTROL TO THE ERROR
;CALL AT ERR2.

MODDOV: MOV (SP)+,R1 ;GET A POINTER TO THE ARGUMENTS
MOV #200,R0 ;SET FD MODE.
LDFPS R0
MOV R1,R0 ;SET UP ACO
LDD (R0),ACO
MOV @MODP1,R0 ;PUT A BACKGROUND PATTERN INTO AC1.
LDD (R0),AC1
MOV 40(R1),R0 ;SET UP THE FPS.


```

20306 103750 170100 LDFPS R0
20307 103752 010100 MOV R1,R0 ;COMPUTE THE ADDRESS OF THE FSRC
20308 103754 062700 000010 ADD #10,R0
20309
20310 103760 171410 1#: MODD (R0),AC0 ;EXECUTE THE TEST INSTRUCTION.
20311
20312 103762 170305 STST R5 ;GET THE FPS.
20313 103764 170204 STFPS R4 ;GET THE FPS.
20314 103766 012700 000200 MOV #200,R0 ;SET FD MODE.
20315 103772 170100 LDFPS R0
20316 103774 012700 104116 MOV #MODDD0,R0 ;GET THE FRACTIONAL RESULT.
20317 104000 174010 STD AC0,(R0)
20318 104002 012700 104126 MOV #MODDD1,R0 ;GET THE INTEGER RESULT.
20319 104006 174110 STD AC1,(R0)
20320 104010 012702 104116 MOV #MODDD0,R2 ;CHECK THE FRACTIONAL RESULT.
20321 104014 010103 MOV R1,R3
20322 104016 062703 000020 ADD #20,R3
20323 104022 012700 000004 MOV #4,R0
20324 104026 022223 2#: CMP (R2)+,(R3)+
20325 104030 001401 BEQ 4#
(2) 104032 104000 EMT ;
20326 104034 077004 4#: SOB R0,2#
20327 104036 012702 104126 MOV #MODDD1,R2 ;CHECK THE INTEGER RESULT
20328 104042 010103 MOV R1,R3
20329 104044 062703 000030 ADD #30,R3
20330 104050 012700 000004 MOV #4,R0
20331 104054 022223 3#: CMP (R2)+,(R3)+
20332 104056 001401 BEQ 5#
(2) 104060 104000 EMT ;
20333 104062 077004 5#: SOB R0,3#
20334 104064 026104 000042 CMP 42(R1),R4 ;CHECK THE FPS.
20335 104070 001401 BEQ 6#
(2) 104072 104000 EMT ;
20336 104074 026105 000044 6#: CMP 44(R1),R5 ;CHECK THE FEC.
20337 104100 001002 BNE 25#
20338
20339 104102 000161 000050 9#: JMP 50(R1) ;RETURN.
20340 ;REPORT FEC ERROR.
20341 104106 010102 25#: MOV R1,R2
20342 104110 062702 000046 ADD #46,R2
20343 104114 000112 JMP (R2)
20344
20345 104116 000000 000000 000000 MODDD0: .WORD 0,0,0,0
104124 000000
20346
20347 104126 000000 000000 000000 MODDD1: .WORD 0,0,0,0
104134 000000
20348
20349 104136 NNNDONE:
(1) 104136 004767 020410 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
20350
20356

```

B11

Address	Instruction	OpCode	Target	Comments
20357	(2)			
20358	104142	012737	104156	000244
20359	104150	170227	000000	
20360	104154	000401		
20361	104156			
20362	(2)	104156	104000	
20363	104160	012700	177777	
20364	104164	170127	000000	
20365	104170	170200		
20366	104172	005700		
20367	104174	001401		
20368	(2)	104176	104000	
20369	104200	012700	104734	
20370	104204	172440		
20371	104206	022700	104730	
20372	(2)	104212	104000	
20373	104214	170200		
20374	104216	022700	000004	
20375	104220	001401		
20376	(2)	104224	104000	
20377	104226	170127	000000	
20378	104230	012700	104734	
20379	104234	174040		
20380	104240	022700	104730	
20381	104242	001401		
20382	(2)	104246	104000	
20383	104250	170200		
20384	104252	005700		
20385	(2)	104254	104000	
20386	104256	001401		
20387	104260	104000		
20388	104262	170127	000000	
20389	104266	012737	104312	000244
20390	104274	170127	004000	
20391	104300	172437	104734	
20392	104304	174437	104764	
20393	104310	104000		
20394	104312	170200		
20395	104314	022700	104004	
20396	104320	001401		
20397	(2)	104322	104000	
20398	104324	012700	104724	
20399	104330	174010		
20400	104332	005737	104724	
20401	104336	001401		
20402	(2)	104340	104000	
20403	104342	012737	104362	000244
20404	104350	170127	004000	
20405	104354	177437	104764	
20406	104360	104000		

```

;*****
;TEST 511      MORE MICROCODES COVERAGE
;*****
T5511:
XT1:   MOV     #XT1A,B#244
      STFPS  #0
      BR     XT2

XT1A:  EMT

XT2:   MOV     # 1,R0
      LDFPS  #0
      STFPS  R0
      TST   R0
      BEQ   XT2A
      EMT

XT2A:  MOV     #XPATO,R0
      LDF    -(R0),AC0
      CMP    #XPATO-4,R0
      BEQ   XT2B
      EMT

XT2B:  STFPS  R0
      CMP    #4,R0
      BEQ   XT3
      EMT

XT3:   LDFPS  #0
      MOV    #XPATO,R0
      STF   AC0,-(R0)
      CMP    #XPATO-4,R0
      BEQ   XT3A
      EMT

XT3A:  STFPS  R0
      TST   R0
      BEQ   XT4
      EMT

XT4:   LDFPS  #0
      MOV    #XT4A,B#244
      LDFPS  #04000
      LDF    #XPATO,AC0
      DIVF  #XPAT3,AC0
      EMT

XT4A:  STFPS  R0
      CMP    #104004,R0
      BEQ   XT4B
      EMT

XT4B:  MOV     #XBUF,R0
      STF   AC0,(R0)
      TST   #XBUF
      BEQ   XT5
      EMT

XT5:   MOV     #XT5A,B#244
      LDFPS  #04000
      LDCDF #XPAT3,AC0
      EMT

;INTRPT ON UNDEFINED VARIABLE
;GET UNDEFINED VARIABLE, _0
;CHECK: FER,FIUV,FZ ARE SET?
;INTRPT ON UNDEFINED VARIBALE
;GET UNDEFINED VARIABLE, 0

```


20450 104614 170127 000000
20451 104620 172437 104774
20452 104624 173437 104774
20453 104630 170200
20454 104632 022700 000004
20455 104636 001401
(2) 104640 104000

XT11: LDFPS #0
LDF @XPAT4,ACO
CHEF @XPAT4,ACO
STFPS RO
CMP #4,RO ;CHECK IF FZ IS SET?
BEQ XT12
EMT ;

20456
20457 104642 170127 000000
20458 104646 172437 104774
20459 104652 174437 104754
20460 104656 012700 104724
20461 104662 174010
20462 104664 022737 040176 104724
20463 104672 001401
(2) 104674 104000

XT12: LDFPS #0
LDF @XPAT4,ACO
DIVF @XPAT2,ACO
MOV @XBUF,RO
STF ACO,(RO)
CMP #040176,@XBUF ;CHECK DATA
BEQ XT13
EMT ;

20464
20465 104676 170127 000000
20466 104702 172437 105004
20467 104706 174437 105014
20468 104712 170200
20469 104714 022700 000004
20470 104720 001445
(2) 104722 104000

XT13: LDFPS #0
LDF @XPAT5,ACO
DIVF @XPAT6,ACO
STFPS RO
CMP #4,RO
BEQ XTDONE
EMT ;

20471
20472

20473 104724 000000 000000 000000
104732 000000
20474 104734 000000 000000 000000
104742 000000
20475 104744 000001 000001 000001
104752 000001
20476 104754 040401 000000 000000
104762 000000
20477 104764 100000 000000 000000
104772 000000
20478 104774 040400 000000 000000
105002 000000
20479 105004 000207 000000 000000
105012 000000
20480 105014 077007 000000 000000
105022 000000
20481 105024 000000 000000 000000
105032 000000

XBUF: .WORD 0,0,0,0
XPAT0: .WORD 0,0,0,0
XPAT1: .WORD 1,1,1,1
XPAT2: .WORD 40401,0,0,0
XPAT3: .WORD 100000,0,0,0
XPAT4: .WORD 040400,0,0,0
XPAT5: .WORD 207,0,0,0
XPAT6: .WORD 77007,0,0,0
XPAT0: .WORD 0,0,0,0

20482
20483 105034
(1) 105034 004767 017512
(1)
(1)
(1)
(1)

XTDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

20484
20485
20486
20492
20493

;*****

```

(2)
(3)
(2) 105040
20494
20495 105040 005000
20496 105042 170100
20497
20498 105044 012737 105064 000244
20499 105052 012737 105060 037576
20500
20501 105060 174007
20502
20503
20504
20505 105062
20506 105062 104000
20507
20508
20509 105064 011600
20510 105066 022700 105062
20511 105072 001420
(3) 105074 104000
20512
20513 105076 170204
20514 105100 170305
20515 105102 012702 100000
20516 105106 012703 000002
20517 105112 022626
20518
20519 105114 020204
20520 105116 001401
(2) 105120 104000
20521 105122 020305
20522 105124 001401
(2) 105126 104000
20523
20524 105130
(1) 105130 004767 017416
(1)
(1)
(1)
(1)
(1)
20525
20526
20527
20533
20534
(2)
(3)
(2) 105134
20535
20536
20537 105134 012700 177777
20538 105140 012701 105250
20539 105144 012702 000014
20540 105150 010021

;TEST 512 STF WITH ILLEGAL ACCUMULATOR TEST
;*****
TS512:
CLR R0 ;SET THE FPS.
LDFPS R0
MOV #000T,#FPVECT ;SET UP FOR FP TRAPS.
MOV #10,#TMP2
11: STF ACC,-C7 ;THIS TEST INSTRUCTION SHOULD
;CAUSE A TRAP.

;REPORT FAILURE OF USE OF ILLEGAL ACCUMULATOR 7 TO CAUSE AN FPP TRAP.
0002:
EMT ;INSTRUCTION DID NOT TRAP

;TRAP TO 000T, HERE, WHEN THE EXPECTED ERROR OCCURS.
000T: MOV (SP),R0 ;MAKE SURE THE ERROR OCCURRED
CMP #0002,R0 ;AT THE CORRECT ADDRESS.
BEQ TS513
EMT ;FLOATING POINT TRAP DID NOT OPERATE RIGHT

0003: STF#S R4 ;GET FPS.
STST R5 ;GET FEC.
MOV #100000,R2 ;EXPECTED FPS
MOV #2,R3 ;EXPECTED FEC
CMP (SP)+,(SP)+ ;RESET THE STACK.

CMP R2,R4 ;WAS FPS CORRECT?
BEQ 0004
EMT
0004: CMP R3,R5 ;FPS INCORRECTLY SET AFTER USE OF ILLEGAL ACC
BEQ 0000ONE ;WAS THE FEC CORRECT?
EMT ;INCORRECT FEC AFTER USE OF ILLEGAL ACC

0000ONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 513 FDST MODE 1, FLOATING MODE, TEST
;*****
TS513:
MOV #-1,R0 ;SET UP A BACKGROUND PATTERN IN THE
MOV #PPBF0,R1 ;INPUT BUFFER.
MOV #14,R2
PPP2: MOV R0,(R1)+

```

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58 P11 07 JAN-85 09:05

MACY11 30(1046) 07 JAN 85 09:28 PAGE 27 20
TS13 FDST MODE 1, FLOATING MODE, TEST

SEQ 0355

20541 105152 077202
20542
20543 105154 012700 000200
20544 105160 17C100
20545 105162 012700 105300
20546 105166 172410
20547

S08 R2,PPP2
MOV @2C0,R0 ;SET FD MODE.
LDFPS R0
MOV @PPPTP1,R0 ;PUT TEST DATA INTO ACO.
LDD (R0),ACO

20549 105170 012700 105264
20550 105174 005002
20551 105176 170102
20552
20553 105200 174010
20554
20555 105202 022700 105264
20556 105206 001401
(2) 105210 104000
20557
20558 105212 012700 105264
20559 105216 012701 105300
20560 105222 022021
20561 105224 001031
20562 105226 022011
20563 105230 001027
20564 105232 022720 177777
20565 105236 001024
20566 105240 022710 177777
20567 105244 001021
20568 105246 000421
20569
20570 105250 177777 177777 177777
105256 177777 177777 177777
20571
20572 105264 177777 177777 177777
105272 177777 177777 177777
20573
20574 105300 123456 023456
20575 105304 034567 045671
20576
20577 105310
(2) 105310 104000
20578 105312
(1) 105312 004767 017234
(1)
(1)
(1)
(1)
20579
20580
20581
20587
20588
(2)
(3)
(2) 105316
20589
20590
20591
20592 105316 012700 177777
20593 105322 012701 105432
20594 105326 012702 000014
20595 105332 010021
20596 105334 077202
20597

```
MOV @PPPBF1,R0 ;FDST ADDRESS.  
CLR R2 ;CLEAR THE FPS.  
LDFPS R2  
PPP3: STF ACO,(R0) ;TEST INSTRUCTION  
CMP @PPPBF1,R0 ;WAS R0 MODIFIED DURING EXECUTION?  
BEQ PPP4  
EMT ;R0 MODIFIED  
PPP4: MOV @PPPBF1,R0 ;CHECK THE DATA IN THE OUTPUT BUFFER.  
MOV @PPPTP1,R1  
CMP (R0),.(R1).  
BNE PPP10 ;BRANCH IF INCORRECT.  
CMP (R0),.(R1)  
BNE PPP10 ;BRANCH IF INCORRECT.  
CMP #-1,(R0). ;WAS FLOATING MODE USED?  
BNE PPP10 ;BRANCH IF NOT.  
BR PPPDONE ;GO TO NEXT TEST.  
PPPBF0: .WORD -1,-1,-1,-1, 1, 1  
PPPBF1: .WORD -1,-1,-1,-1, 1,-1  
PPPTP1: .WORD 123456,23456  
.WORD 34567,45671  
PPP10:  
EMT ;  
PPPDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND  
;SEE IF THE USER HAS EXPRESSED  
;THE DESIRE TO CHANGE THE SOFTWARE  
;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
;THE USER TYPED CONTROL G?).
```

;TEST 514 FDST MODE 2 TEST

TS514:

```
;FIRST TEST STF.  
MOV #-1,R0 ;SET UP THE OUTPUT BUFFER.  
MOV @QQQBF0,R1  
MOV #14,R2  
QQQ2: MOV R0,(R1).  
SOB R2,QQQ2
```

20598	105336	012700	000200			MOV	#200,R0	;SET FD MODE
20599	105342	170100				LDFPS	R0	
20600	105344	012700	105452			MOV	#000TP1,R0	;SETUP ACO.
20601	105350	172410				LDD	(R0),ACO	
20602								
20603	105352	012700	105446			MOV	#000BF1,R0	;FDST ADDRESS.
20604	105356	005002				CLR	R2	
20605	105360	170102				LDFPS	R2	;SET FPS.
20606	105362	174020			0003:	STF	ACO,(R0).	;TEST INSTRUCTION.
20607								
20608	105364	022700	105452			CMP	#000BF1+4,R0	;WAS R0 INCREMENTED BY 4 PROPERLY?
20609								
20610	105370	001401				BEQ	0004	
(2)	105372	104000				EMT		;REPORT R0 INCORRECT AFTER FDST MODE 2
20611	105374	012700	105446		0004:	MOV	#000BF1,R0	;WAS THE OUTPUT DATA CORRECT?
20612	105400	012701	105462			MOV	#000TP1,R1	
20613	105404	022021				CMP	(R0)+,(R1)+	
20614	105406	001031				BNE	00010	;BRANCH IF INCORRECT.
20615	105410	022021				CMP	(R0)+,(R1)+	
20616	105412	001027				BNE	00010	;BRANCH IF INCORRECT.
20617	105414	022027	177777			CMP	(R0)+,#-1	;SEE IF ANY OTHER DATA BUFFER WORDS WERE MODIFIED.
20618	105420	001024				BNE	00010	;BRANCH IF INCORRECT.
20619	105422	022027	177777			CMP	(R0)+,#-1	
20620	105426	001021				BNE	00010	;BRANCH IF INCORRECT.
20621	105430	000421				BR	00020	
20622	105432	177777	177777	177777	000BF0:	.WORD	-1,-1,-1,1,1	
	105440	177777	177777	177777				
20623	105446	177777	177777	177777	000BF1:	.WORD	-1,-1,-1,1,-1,1	
	105454	177777	177777	177777				
20624	105462	076543			000TP1:	76543		
20625	105464	065432				65432		
20626	105466	054321				54321		
20627	105470	043210				43210		
20628								;REPORT OUTPUT DATA INCORRECT:
20629	105472				00010:	EMT		
(2)	105472	104000						
20630								
20631								;NOW TEST STD MODE 2.
20632								
20633	105474	012700	105432		00020:	MOV	#000BF0,R0	;SET UP DEFAULT INPUT DATA BUFFER.
20634	105500	010001				MOV	R0,R1	
20635	105502	012702	000014			MOV	#14,R2	
20636	105506	010021			00022:	MOV	R0,(R1)+	
20637	105510	077202				SQB	R2,00022	
20638	105512	012700	000200			MOV	#200,R0	;ENTER FLOATING DOUBLE MODE.
20639	105516	170100				LDFPS	R0	
20640	105520	012700	105462			MOV	#000TP1,R0	;LOAD ACO.
20641	105524	172410				LDD	(R0),ACO	
20642	105526	012700	105446			MOV	#000BF1,R0	;SET DESTINATION ADDRESS.
20643	105532	012737	105540	037576		MOV	#00023,B#TMP2	
20644	105540	174020			00023:	STD	ACO,(R0)+	;TEST INSTRUCTION.
20645	105542	022700	105456			CMP	#000BF1+10,R0	;WAS R0 INCREMENTED BY 10 CORRECTLY?
20646	105546	001401				BEQ	00024	
(2)	105550	104000				EMT		;REPORT R0 INCORRECTLY INCREMENTED
20647	105552	012700	105446		00024:	MOV	#000BF1,R0	;DID THE DATA REACH THE OUTPUT BUFFER CORRECTLY?
20648	105556	012701	105462			MOV	#000TP1,R1	

20649 10552 012702 000004
20650 10556 022021
20651 10557 001002
20652 10557 077203
20653 105574 000401
20654
20655 105576
(2) 105576 104000
20656 105600
(1) 105600 004767 016746
(1)
(1)
(1)
(1)
20657
20663
(2)
(3)
(2) 105604
20664
20665 105604 012700 105654
20666 105610 012701 105722
20667 105614 012702 000004
20668 105620 012021
20669 105622 077202
20670 105624 012700 000200
20671 105630 170100
20672 105632 012700 105732
20673 105636 172410
20674 105640 012737 105720 000004
20675 105646 005001
20676 105650 005004
20677
20678
20679
20680
20681
20682
20683 105652 174027
20684 105654 005201
20685 105656 005201
20686 105660 005201
20687 105662 005201
20688 105664 012700 105742
20689 105670 012702 105654
20690 105674 012703 000004
20691 105700 022022
20692 105702 001006
20693 105704 077303
20694 105706 005704
20695 105710 001003
20696 105712 022701 000003
20697 105716 001415
(1) 105720
(2) 105720 104000
20698

11,
MOV #4,R2
14: CMP (R0),.(R1).
BNE 00Q25 ;BRANCH IF INCORRECT.
SOB R2,14
BR 00QDONE
;REPORT DATA INCORRECT.
00Q25: EMT ;
00QDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 515 FDST MODE 2, WITH GR7, TEST
;*****
TS515:
MOV #RRR3,R0 ;SET UP THE DATA BUFFER FOLLOWING THE TEST INSTRUCTION
MOV #RRRTP1,R1
MOV #4,R2
14: MOV (R0),.(R1).
SOB R2,14
MOV #200,R0 ;ENTER FLOATING DOUBLE MODE.
LDFPS R0
MOV #RRRTP2,R0 ;SET UP ACO.
LDD (R0),ACO
MOV #RRR10,#ERRVECT ;SET UP FOR AN ODD ADDRESS.
CLR R1
CLR R4
;THIS IS THE TEST INSTRUCTION. IT SHOULD MODIFY THE FIRST LOCATION
;AFTER IT TO BE AN INCREMENT R4, INC R4, INSTRUCTION INSTEAD
;OF AN INCREMENT R1 INSTRUCTION. THE INCREMENT R4 SHOULD NOT BE
;EXECUTED SINCE THE PC SHOULD BE INCREMENTED BY TWO DURING IMMEDIATE
;MODE ADDRESSING. THUS AFTER THE EXECUTION OF THE NEXT 5 INSTRUCTIONS
;R1 SHOULD CONTAIN 3 AND R4 SHOULD CONTAIN 0.
RRR2: STD ACO,(R7). ;TEST INSTRUCTION.
RRR3: INC R1 ;THE STD INSTRUCTION SHOULD CHANGE THIS TO INC R4.
INC R1
INC R1
INC R1
MOV #RRREXP,R0 ;SEE IF THE DATA WAS OUTPUT CORRECTLY.
MOV #RRR3,R2
RRR4: MOV #4,R3
CMP (R0),.(R2).
BNE RRR10 ;BRANCH IF INCORRECT.
SOB R3,RRR4
TST R4 ;MAKE SURE R4 IS 0.
BNE RRR10 ;BRANCH IF R4 IS INCORRECT.
CMP #3,R1 ;SEE IF R1 IS CORRECT.
BEQ RRRDONE
RRR10: EMT ;
;THESE ARE TEST DATA PATTERNS USED TO SET UP THE OUTPUT BUFFER AT RRR3.

20699 105722 005201
 20700 105724 005201
 20701 105726 005201
 20702 105730 005201
 20703
 20704 105732 005204
 20705 105734 005204
 20706 105736 005204
 20707 105740 005204
 20708
 20709 105742 005204
 20710 105744 005201
 20711 105746 005201
 20712 105750 005201
 20713 105752
 (1) 105752 004767 016574
 (1)
 (1)
 (1)
 (1)
 20714
 20720
 (2)
 (3)
 (2) 105756
 20721
 20722 105756 012700 177777
 20723 105762 012701 106104
 20724 105766 012702 000010
 20725 105772 010021
 20726 105774 077202
 20727 105776 012700 000200
 20728 106002 170100
 20729 106004 012700 106124
 20730 106010 172410
 20731 106012 012737 106134 000004
 20732 106020 012700 106114
 20733
 20734 106024 174040
 20735 106026 005201
 20736 106030 020027 106104
 20737 106034 001037
 20738 106036 012700 106104
 20739 106042 012701 106124
 20740 106046 012702 000004
 20741 106052 022021
 20742 106054 001027
 20743 106056 077203
 20744 106060 012700 177777
 20745 106064 012701 106114
 20746 106070 012702 000004
 20747 106074 020021
 20748 106076 001016
 20749 106100 077203
 20750 106102 000415
 20751

```

RRRTP1: INC R1
        INC R1
        INC R1
        INC R1
;THIS IS THE DATA PUT IN ACO BEFORE EXECUTION OF THE STD.
RRRTP2: INC R4
        INC R4
        INC R4
        INC R4
;THIS IS THE EXPECTED DATA AT RRR3 AFTER EXECUTION OF THE STD.
RRREXP: INC R4
        INC R1
        INC R1
        INC R1
RRRDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 516      FDST MODE 4 TEST
;*****
TS516:
        MOV    #-1,R0 ;SET UP THE OUTPUT BUFFER.
        MOV    #SSSBF0,R1
        MOV    #10,R2
1$:     MOV    R0,(R1)+
        SOB    R2,1$
        MOV    #200,R0 ;ENTER FLOATING DOUBLE MODE.
        LDFPS R0
        MOV    #SSSTP1,R0 ;SET UP ACO.
        LDD    (R0),ACO
        MOV    #SSS10,@ERRVECT ;SET UP FOR A TRAP TO 4.
        MOV    #SSSA1,R0 ;SET UP THE DESTINATION ADDRESS.

SSS2:  STD    ACO,-(R0) ;TEST INSTRUCTION.
        INC    R1
        CMP    R0,#SSSBF0 ;SEE IF R0 WAS DECREMENTED PROPERLY.
        BNE   SSS10 ;BRANCH IF R0 IS INCORRECT.
        MOV    #SSSBF0,R0 ;WAS THE OUTPUT DATA CORRECT?
        MOV    #SSSTP1,R1
        MOV    #4,R2
1$:     CMP    (R0)+,(R1)+
        BNE   SSS10 ;BRANCH IF INCORRECT.
        SOB    R2,1$
        MOV    #-1,R0 ;IS THE REST OF THE OUTPUT BUFFER CORRECT. 1?
        MOV    #SSSA1,R1
        MOV    #4,R2
2$:     CMP    R0,(R1)+
        BNE   SSS10 ;BRANCH IF INCORRECT.
        SOB    R2,2$
        BR    SSSDONE

```

1412

20752
20753 106104 177777
20754 106106 177777
20755 106110 177777
20756 106112 177777
20757 106114 177777
20758 106116 177777
20759 106120 177777
20760 106122 177777
20761
20762
20763 106124 147250
20764 106126 036147
20765 106130 025036
20766 106132 147250
20767
20768 106134
(2) 106134 104000
20769 106136
(1) 106136 004767 016410
(1)
(1)
(1)
(1)
(1)
20770
20776
(2)
(3)
(2) 106142
20777
20778 106142 012701 106252
20779 106146 012700 177777
20780 106152 012702 000013
20781 106156 010021
20782 106160 077202
20783 106162 012737 106252 106266
20784 106170 012700 000200
20785 106174 170100
20786 106176 012700 106270
20787 106202 172410
20788 106204 012737 106300 000004
20789 106212 012700 106266
20790
20791 106216 174030
20792
20793 106220 020027 106270
20794 106224 001025
20795 106226 012701 106252
20796 106232 012702 106270
20797 106236 012703 000004
20798 106242 022122
20799 106244 001015
20800 106246 077303
20801 106250 000414
20802
20803

;THIS IS THE OUTPUT DATA BUFFER.

SSSBFO: 1
-1
-1
1
SSSA1: 1
-1
-1
1

;THIS IS THE TEST DATA LOADED INTO ACO:

SSSTP1: 147250
36147
25036
147250

SSS10:

EMT

SSSDONE:

JSR PC,.RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 517 FDST MODE 3 TEST
;*****
TSS17:

MOV #TTTBFO,R1 ;SET UP THE OUTPUT DATA BUFFER.
MOV #-1,R0
MOV #13,R2
14: MOV R0,(R1)+
SOB R2,14
MOV #TTTBFO,#TTTA2
MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV #TTTTP1,R0 ;SET UP ACO.
LDD (R0),ACO
MOV #TTT10,#ERRVECT ;SET UP FOR TRAPS TO 4.
MOV #TTTA2,R0 ;SET UP THE DESTINATION ADDRESS.
TTT2: STD ACO,#(R0)+ ;TEST INSTRUCTION.
CMP R0,#TTTA2+2 ;SEE IF R0 WAS INCREMENTED CORRECTLY.
BNE TTT10 ;BRANCH IF INCORRECT.
MOV #TTTBFO,R1 ;CHECK THE OUTPUT DATA BUFFER.
MOV #TTTTP1,R2
MOV #4,R3
TTT3: CMP (R1)+,(R2)+
BNE TTT10 ;BRANCH IF NOT CORRECT.
SOB R3,TTT3
BR TTTDONE

;THIS IS THE OUTPUT DATA BUFFER:

20804 106252 177777
 20805 106254 177777
 20806 106256 177777
 20807 106260 177777
 20808 106262 177777
 20809 106264 177777
 20810 106266 106252
 20811 106270 101213
 20812 106272 141516
 20813 106274 071727
 20814 106276 037475
 20815
 20816 106300
 (2) 106300 104000
 20817
 20818 106302
 (1) 106302 004767 016244
 (1)
 (1)
 (1)
 (1)
 20819
 20825
 (2)
 (3)
 (2) 106306
 20826
 20827 106306 012701 106416
 20828 106312 012700 177777
 20829 106316 012702 000013
 20830 106322 010021
 20831 106324 077202
 20832 106326 012737 106416 106430
 20833 106334 012700 000200
 20834 106340 170100
 20835 106342 012700 106434
 20836 106346 172410
 20837 106350 012737 106444 000004
 20838 106356 012700 106432
 20839 106362 174050
 20840 106364 020027 106430
 20841 106370 001025
 20842 106372 012701 106416
 20843 106376 012702 106434
 20844 106402 012703 000004
 20845 106406 022122
 20846 106410 001015
 20847 106412 077303
 20848 106414 000414
 20849
 20850
 20851 106416 177777
 20852 106420 177777
 20853 106422 177777
 20854 106424 177777
 20855 106426 177777

```

TTBFO: 1
        1
        -1
        -1
        -1
TTTA1: -1
TTTA2: TTBFO
TTTP1: 101213
        141516
        71727
        37475

TTT10:
        EMT

TTTDONE:
        JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                                ;SEE IF THE USER HAS EXPRESSED
                                ;THE DESIRE TO CHANGE THE SOFTWARE
                                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                ;THE USER TYPED CONTROL G?).

;*****
;TEST 520      FDST MODE 5 TEST
;*****
TS520:
        MOV      @UUUBFO,R1      ;SET UP THE OUTPUT DATA BUFFER.
        MOV      #-1,R0
        MOV      #13,R2
14:     MOV      R0,(R1)+
        SOB      R2,14
        MOV      @UUUBFO,@UUUA1
        MOV      #200,R0          ;ENTER DOUBLE FLOATING MODE.
        LDFPS   R0
        MOV      @UUUTP1,R0      ;SET UP ACO.
        LDD     (R0),ACO
        MOV      @UUU10,@MERRVECT ;GET READY FOR ANY TRAPS TO 4.
        MOV      @UUUA2,R0      ;SET UP THE DESTINATION ADDRESS.
UUU2:   STD     ACO,@-(R0)      ;TEST INSTRUCTION.
        CMP     R0,@UUUA2-2     ;WAS R0 DECRIMENTED PROPERLY?
        BNE    UUU10           ;BRANCH IF R0 IS INCORRECT.
        MOV     @UUUBFO,R1      ;WAS THE DATA OUTPUT CORRECTLY?
        MOV     @UUUTP1,R2
        MOV     #4,R3
UUU3:   CMP     (R1)+,(R2)+
        BNE    UUU10           ;BRANCH IF DATA IS INCORRECT.
        SOB    R3,UUU3
        BR     UUDONE

;THIS IS THE OUTPUT DATA BUFFER
UUUBFO: -1
        -1
        -1
        1
        -1

```

MLL

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 28 6
T520 FDST MODE 5 TEST

EQ 0362

20856 106430 106416
 20857 106432 177777
 20858 106434 020212
 20859 106436 023242
 20860 106440 026273
 20861 106442 031323
 20862
 20863 106444
 (2) 106444 104000
 20864 106446
 (1) 106446 004767 016100
 (1)
 (1)
 (1)
 (1)
 20865
 20871
 (2)
 (3)
 (2) 106452
 20872
 20873 106452 012700 000200
 20874 106456 170100
 20875 106460 012701 106562
 20876 106464 012700 177777
 20877 106470 012702 000004
 20878 106474 010021
 20879 106476 077202
 20880 106500 012737 106602 000004
 20881 106506 012700 106572
 20882 106512 172410
 20883 106514 012700 100661
 20884 106520 012701 000001
 20885 106524 174060 005701
 20886
 20887 106530 020027 100661
 20888 106534 001022
 20889 106536 012702 106562
 20890 106542 012703 106572
 20891 106546 012704 000004
 20892 106552 022223
 20893 106554 001012
 20894 106556 077403
 20895 106560 000411
 20896 106562 177777
 20897 106564 177777
 20898 106566 177777
 20899 106570 177777
 20900 106572 030313
 20901 106574 023334
 20902 106576 035363
 20903 106600 074041
 20904
 20905 106602
 (2) 106602 104000
 20906 106604

UUUA1: UUUBFO
 UUUA2: -1
 UUUTP1: 20212
 23242
 26273
 031323

UUU10:
 EMT ;
 UUUDONE: JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK, AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 521 FDST MODE 6, INDEX MODE, TEST
 ;*****
 TS521:
 MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
 LDFPS R0
 MOV #VVVBFO,R1 ;SET UP THE OUT PUT DATA BUFFER.
 MOV #-1,R0
 MOV #4,R2
 1\$: MOV R0,(R1)+
 SOB R2,1\$
 MOV #VVV10,#ERRVECT ;SET UP VECTOR 4 INCASE OF ERROR.
 MOV #VVVTP1,R0 ;SET UP ACO.
 LDD (R0),ACO
 MOV #VVVBFO-5701,R0 ;SET UP THE DESTINATION ADDRESS.
 MOV #1,R1
 VVV2: STD ACO,5701(R0) ;TEST INSTRUCTION.
 CMP R0,#VVVBFO-5701 ;SEE IF R0 WAS MODIFIED.
 BNE VVV10 ;BRANCH IF INCORRECT.
 MOV #VVVBFO,R2 ;WAS THE OUTPUT DATA CORRECT.
 MOV #VVVTP1,R3
 MOV #4,R4
 1\$: CMP (R2)+,(R3)+
 BNE VVV10 ;BRANCH IF INCORRECT DATA.
 SOB R4,1\$
 BR VVVDONE
 VVVBFO: -1
 -1
 1
 -1
 VVVTP1: 30313
 23334
 35363
 74041
 VVV10:
 EMT ;
 VVVDONE:

Line	Address	Value	Label	Field	Value	Field	Value
20954							
20960							
(2)							
(3)							
(2)	106764						
20961							
20962							
20963	106764	004767	000262	XXX1:	JSR	PC,STCFDS	
20964	106770	000000		18:	0		;AC
20965	106772	000000			0		
20966	106774	000000			0		
20967	106776	000000			0		
20968	107000	000000		24:	0		;RES
20969	107002	000000			0		
20970	107004	000000			0		
20971	107006	000000			0		
20972	107010	000000		34:	0		;ERROR RES.
20973	107012	000000			0		
20974	107014	177777			-1		
20975	107016	177777			-1		
20976	107020	047000		44:	47000		;FPS BEFORE EXECUTION.
20977	107022	047004			47004		;FPS AFTER EXECUTION.
20978	107024	177777			-1		;FEC
20979	107026	147004			147004		;ERROR FPS.
20980							
20981							
20982	107030	004767	000216	XXX2:	JSR	PC,STCFDS	
20983	107034	017203		18:	17203		;AC
20984	107036	142536			142536		
20985	107040	047506			47506		
20986	107042	172031			172031		
20987	107044	017203		24:	17203		;RES
20988	107046	142536			142536		
20989	107050	000000			0		
20990	107052	000000			0		
20991	107054	017203		34:	17203		;ERROR RES.
20992	107056	142536			142536		
20993	107060	047506			47506		
20994	107062	172031			172031		
20995	107064	040000		44:	40000		;FPS BEFORE EXECUTION.
20996	107066	040000			40000		;FPS AFTER EXECUTION.
20997	107070	177777			-1		;FEC
20998	107072	177777			-1		;ERROR FPS.
20999							
21000							
21001	107074	004767	000152	XXX3:	JSR	PC,STCFDS	
21002	107100	050717		18:	50717		;AC
21003	107102	027374			27374		
21004	107104	075767			75767		
21005	107106	077071			77071		
21006	107110	050717		24:	50717		;RES
21007	107112	027374			27374		
21008	107114	000000			0		
21009	107116	000000			0		
21010	107120	000000		34:	0		;ERROR RES.
21011	107122	000000			0		

013

21012	107124	000000			0	
21013	107126	000000			0	
21014	107130	047000		44:	47000	;FPS BEFORE EXECUTION.
21015	107132	047000			47000	;FPS AFTER EXECUTION.
21016	107134	177777			1	;FEC
21017	107136	174002			174002	;ERROR FPS.
21018						
21019						
21020	107140	004767	000106	XXX4:	JSR	PC,STCFDS
21021	107144	020212		14:	20212	;AC
21022	107146	032425			32425	
21023	107150	026272			26272	
21024	107152	002123			02123	
21025	107154	020212		24:	20212	;RES
21026	107156	032425			32425	
21027	107160	000000			0	
21028	107162	000000			0	
21029	107164	020212		34:	20212	;ERROR RES.
21030	107166	032425			32425	
21031	107170	100000			100000	
21032	107172	000000			0	
21033	107174	040000		44:	40000	;FPS BEFORE EXECUTION.
21034	107176	040000			40000	;FPS AFTER EXECUTION.
21035	107200	177777			-1	;FEC
21036	107202	177777			-1	;ERROR FPS.
21037						
21038						
21039	107204	004767	000042	XXX5:	JSR	PC,STCFDS
21040	107210	121314		14:	121314	;AC
21041	107212	151617			151617	
21042	107214	101112			101112	
21043	107216	131415			131415	
21044	107220	121314		24:	121314	;RES
21045	107222	151617			151617	
21046	107224	000000			0	
21047	107226	000000			0	
21048	107230	021314		34:	21314	;ERROR RES.
21049	107232	151617			151617	
21050	107234	000000			0	
21051	107236	000000			0	
21052	107240	040000		44:	40000	;FPS BEFORE EXECUTION.
21053	107242	040010			40010	;FPS AFTER EXECUTION.
21054	107244	177777			-1	;FEC
21055	107246	177777			-1	;ERROR FPS.
21056	107250	000460		64:	BR	XXXDONE

21057
 21058
 21059
 21060
 21061
 21062
 21063
 21064
 21065
 21066
 21067

;THIS SUBROUTINE, STCFDS, IS USED TO SET UP THE OPERANDS, EXECUTE
 ;THE STCFD INSTRUCTION AND CHECK THE RESULTS. A CALL
 ;TO IT IS MADE THUS:

```

;
;          JSR      PC,STCFDS
;          ACARG:  .WORD  X,X,X,X          ;AC OPERAND
;          RES:    .WORD  X,X,X,X          ;EXPECTED RESULT
;

```


D17

```

21068 :
21069 :
21070 :
21071 :
21072 :
21073 :
21074 :
21075 :
21076 :
21077 :
21078 :
21079 :
21080 :
21081 :
21082 :
21083 :
21084 :
21085 :
21086 :
21087 :
21088 :
21089 :
21090 :
21091 :
21092 :
21093 :
21094 :
21095 :
21096 :
21097 :
21098 :
21099 :
21100 :
21101 :
21102 :
21103 :
21104 :
21105 :
21106 :
21107 :
21108 :
21109 :
21110 :
21111 :
21112 :
21113 :
21114 :
21115 :
21116 :
21117 :
21118 :
21119 :
21120 :
21121 :
21122 :
21123 :

```

```

ERRES: .WORD X,X,X,X ;ERROR RESULT
FPSB: .WORD X ;FPS BEFORE EXECUTION
FPSA: .WORD Y ;FPS AFTER EXECUTION
FEC: .WORD X ;EXPECTED FEC
ERFPS: .WORD X ;ERROR FPS.
ERR1: ERROR X ;DATA ERROR.
BR CONT
ERR2: ERROR X ;FPS ERROR.
CONT: ;RETURN ADDRESS

```

```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE STCFD INSTRUCTION IS EXECUTED.
;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT STCFDS RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCFDS
;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STCFDS WILL RETURN
;TO THE ERROR CALL AT ERR2, OTHERWISE STCFDS ITSELF
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
;STCFD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCFDS
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCFDS WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

```

STCFDS: MOV (SP),R1 ;PICK UP THE POINTER TO THE OPERANDS.
MOV #200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO.
LDD (R0),ACO
MOV #-1,R0 ;FILL THE OUTPUT BUFFER WITH -1'S.
MOV #STCFT,R2
MOV #4,R3
11: MOV R0,(R2)
SOB R3,11
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0
MOV #STCFT,R0 ;SET UP THE DESTINATION ADDRESS.
21: STCFD ACO,(R0) ;TEST INSTRUCTION.

STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV R1,R2 ;CHECK THE RESULT.
ADD #10,R2
MOV #STCFT,R3
MOV #4,R0
31: CMP (R2),,(R3)
BNE 101 ;BRANCH IF INCORRECT.
SOB R0,31

MOV 32(R1),R2
CMP R2,R4 ;IS THE FPS CORRECT?
BNE 101 ;BRANCH IF FPS INCORRECT.
TST R2 ;IF EXPECTED FPS IS NEGATIVE, THEN
BPL 41 ;GO AHEAD AND CHECK THE FEC.
CMP 36(R1),R5

```

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 28 11
T523 STCDF TEST

SEQ 0367

21124 107372 001002
21125 107374 000161 000040
21126 107400
(2) 107400 104000
21127 107402 177777 177777 177777
107410 177777
21128 107412
(1) 107412 004767 015134
(1)
(1)
(1)
(1)

BNE 104
JMP 40(R1)
EHT
STCFT: 1. 1. -1. 1
XXXDONE:
JSR PC.,RSET

;BRANCH IF FEC IS INCORRECT
;RETURN.
;
;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

21129
21135
(2)
(3)
(2) 107416

;*****
;TEST 524 STCDF TEST
;*****
T5524:

21136
21137
21138 107416 004767 000262
21139 107422 000000
21140 107424 000000
21141 107426 000000
21142 107430 000000
21143 107432 000000
21144 107434 000000
21145 107436 177777
21146 107440 177777
21147 107442 000000
21148 107444 000000
21149 107446 000000
21150 107450 000000
21151 107452 047200
21152 107454 047204
21153 107456 177777
21154 107460 177777

;AC=0
YYY1: JSR PC,STCDF5
1#: 0
0
0
0
2#: 0
0
-1
-1
3#: 0
0
0
0
4#: 47200
47204
1
1

;AC
;RES
;ERROR RES.
;FPS BEFORE EXECUTION.
;FPS AFTER EXECUTION
;FEC
;ERROR FPS.

21155
21156
21157 107462 004767 000216
21158 107466 067574
21159 107470 073727
21160 107472 170777
21161 107474 067574
21162 107476 067574
21163 107500 073730
21164 107502 177777
21165 107504 177777
21166 107506 067574
21167 107510 073727
21168 107512 177777
21169 107514 177777
21170 107516 040200
21171 107520 040200
21172 107522 177777
21173 107524 177777
21174

YYY2: JSR PC,STCDF5
1#: 67574
73727
170777
2#: 67574
73730
-1
-1
3#: 67574
73727
-1
-1
4#: 40200
40200
-1
-1

;ACO
;RES
;ERROR RES.
;FPS BEFORE EXECUTION.
;FPS AFTER EXECUTION.
;FEC
;ERROR FPS

21175								
21176	107526	004757	000152	YYY3:	JSR	PC,STCDF5		
21177	107532	077777		1#:	77777			;ACO
21178	107534	177777			-1			
21179	107536	100000			100000			
21180	107540	000000			0			
21181	107542	000000		2#:	0			;RES
21182	107544	000000			0			
21183	107546	177777			-1			
21184	107550	177777			-1			
21185	107552	077777		3#:	77777			;ERROR RES
21186	107554	177777			-1			
21187	107556	177777			-1			
21188	107560	177777			-1			
21189	107562	040200		4#:	40200			;FPS BEFORE EXECUTION.
21190	107564	040206			40206			;FPS AFTER EXECUTION.
21191	107566	177777			1			;FEC
21192	107570	040204			40204			;ERROR FPS.
21193								
21194								
21195	107572	004767	000106	YYY4:	JSR	PC,STCDF5		
21196	107576	077777		1#:	77777			;ACO
21197	107600	177777			-1			
21198	107602	100000			100000			
21199	107604	000000			0			
21200	107606	000000		2#:	0			;RES
21201	107610	000000			0			
21202	107612	177777			-1			
21203	107614	177777			-1			
21204	107616	077777		3#:	77777			;ERROR RES.
21205	107620	177777			-1			
21206	107622	177777			-1			
21207	107624	177777			-1			
21208	107626	040200		4#:	40200			;FPS BEFORE EXECUTION.
21209	107630	040206			40206			;FPS AFTER EXECUTION.
21210	107632	177777			-1			;FEC
21211	107634	140206			140206			;ERROR FPS.
21212								
21213								
21214	107636	004767	000042	YYY5:	JSR	PC,STCDF5		
21215	107642	177777		1#:	177777			;ACO
21216	107644	177777			-1			
21217	107646	100000			100000			
21218	107650	000000			0			
21219	107652	100000		2#:	100000			;RES
21220	107654	000000			0			
21221	107656	177777			-1			
21222	107660	177777			-1			
21223	107662	000000		3#:	0			;ERROR RES.
21224	107664	000000			0			
21225	107666	177777			-1			
21226	107670	177777			1			
21227	107672	047200		4#:	47200			;FPS BEFORE EXECUTION.
21228	107674	147216			147216			;FPS AFTER EXECUTION.
21229	107676	000010			10			;FEC
21230	107700	047206			47206			;ERROR FPS.

21231 107702 000460
 21232
 21233
 21234
 21235
 21236
 21237
 21238
 21239
 21240
 21241
 21242
 21243
 21244
 21245
 21246
 21247
 21248
 21249
 21250
 21251
 21252
 21253
 21254
 21255
 21256
 21257
 21258
 21259
 21260
 21261
 21262
 21263
 21264 107704 012601
 21265 107706 012700 000200
 21266 107712 170100
 21267 107714 010100
 21268 107716 172410
 21269 107720 012700 177777
 21270 107724 012702 110034
 21271 107730 012703 000004
 21272 107734 010022
 21273 107736 077302
 21274 107740 016100 000030
 21275 107744 170100
 21276 107746 012700 110034
 21277 107752 176010
 21278
 21279 107754 170204
 21280 107756 170305
 21281 107760 010102
 21282 107762 062702 000010
 21283 107766 012703 110034
 21284 107772 012700 000004
 21285 107776 022223
 21286 110000 001014

```

64: BR YYDONE
;THIS SUBROUTINE, STCDF, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE STCDF INSTRUCTION AND CHECK THE RESULTS. A CALL
;TO IT IS MADE THUS:
:
: JSR PC,@STCFDS
: ACARG: .WORD X,X,X,X ;AC OPERAND
: RES: .WORD X,X,X,X ;EXPECTED RESULT
: ERRES: .WORD X,X,X,X ;ERROR RESULT
: FPSB: .WORD X ;FPS BEFORE EXECUTION
: FPSA: .WORD X ;FPS AFTER EXECUTION
: FEC: .WORD X ;EXPECTED FEC
: ERFPS: .WORD X ;ERROR FPS.
: ERR1: ERROR X ;DATA ERROR.
: BR CONT
: ERR2: ERROR X ;FPS ERROR.
: CONT: ;RETURN ADDRESS
:
;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE STCDF INSTRUCTION IS EXECUTED.
;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT STCFDS RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCFDS
;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STCFDS WILL RETURN
;TO THE ERROR CALL AT ERR2, OTHERWISE STCFDS ITSELF
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
;STCDF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCFDS
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCFDS WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

STCFDS: MOV (SP)+,R1 ;PICK UP THE POINTER TO THE OPERANDS.
MOV @200,R0 ;ENTER DOUBLE FLOATING MODE.
LDFPS R0
MOV R1,R0 ;LOAD ACO.
LDD (R0),ACO
MOV #-1,R0 ;FILL THE OUTPUT BUFFER WITH 1'S.
MOV @STCDT,R2
MOV @4,R3
14: MOV R0,(R2)+
SOB R3,14
MOV 30(R1),R0 ;LOAD THE FPS.
LDFPS R0
MOV @STCDT,R0 ;SET UP THE DESTINATION ADDRESS.
24: STCDF ACO,(R0) ;TEST INSTRUCTION.

STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV R1,R2 ;CHECK THE RESULT.
ADD @10,R2
MOV @STCDT,R3
MOV @4,R0
34: CMP (R2)+,(R3)+
BNE 104 ;BRANCH IF INCORRECT.

```

```

21287 110002 077003          SOB      R0,3$
21288
21289 110004 016102 000032    MOV      32(R1),R2
21290 110010 020204          CMP      R2,R4          ;IS THE FPS CORRECT?
21291 110012 001007          BNE     10$           ;BRANCH IF FPS INCORRECT.
21292 110014 005702          TST     R2           ;IF EXPECTED FPS IS NEGATIVE, THEN
21293 110016 100003          BPL     4$           ;GO AHEAD AND CHECK THE FEC.
21294 110020 026105 000034    CMP      34(R1),R5
21295 110024 001002          BNE     10$           ;BRANCH IF FEC IS INCORRECT.
21296 110026 000161 000040    4$:     JMP      40(R1)   ;RETURN.
21297 110032          10$:
(2) 110032 104000          EMT
21298 110034 177777 177777 177777 STCDT:  -1, 1,-1,-1
110042 177777
21299 110044          YYYDONE:
(1) 110044 004767 014502    JSR      PC,.RSET    ;GO INITIALIZE THE FPS AND STACK; AND
(1)                                     ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
21305          ;*****
(2)          ;TEST 525      STCFD WITH ILLEGAL ACCUMULATOR TEST
(3)          ;*****
(2) 110050          TS525:
21306
21307 110050 012700 040000    MOV      #40000,R0    ;DISSABLE INTERRUPTS.
21308 110054 170100          LDFPS   R0
21309 110056 176006          ZZZ2:   STCFD     AC0,AC6  ;THIS TEST INSTRUCTION SHOULD CAUSE AN ERROR.
21310
21311 110060 170204          STFPS   R4           ;GET FPS.
21312 110062 170305          STST    R5           ;GET FEC.
21313 110064 020427 140000    CMP      R4,#140000  ;IS FPS CORRECT?
21314 110070 001004          BNE     ZZZ10        ;BRANCH IF INCORRECT FPS.
21315 110072 022705 000002    CMP      #2,R5       ;IS FEC CORRECT?
21316 110076 001001          BNE     ZZZ10        ;BRANCH IF INCORRECT.
21317 110100 000401          BR      ZZZDONE
21318
21319 110102          ZZZ10:
(2) 110102 104000          EMT
21320
21321 110104          ZZZDONE:
(1) 110104 004767 014442    JSR      PC,.RSET    ;GO INITIALIZE THE FPS AND STACK; AND
(1)                                     ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
21322
21328          ;*****
(2)          ;TEST 526      CLRD TEST
(3)          ;*****
(2) 110110          TS526:
21329 110110 012700 110214    MOV      #AABTP1,R0   ;SET UP OUTPUT BUFFER
21330 110114 012701 110204    MOV      #AABBFO,R1
21331 110120 012702 000004    MOV      #4,R2
21332 110124 012021          1$:     MOV      (R0)+,(R1)+
21333 110126 077202          SOB      R2,1$

```

```
21334 110130 012700 110204      MOV      #AABBF0,R0      ;SET UP DESTINATION OPERAND ADDRESS
21335 110134 012701 000213      MOV      #213,R1        ;SET UP FPS.
21336 110140 170101              LDFPS   R1
21337 110142 170410      2$:    CLRD      (R0)      ;TEST INSTRUCTION
21338
21339 110144 170205              STFPS   R5              ;GET FPS.
21340 110146 012702 000004      MOV      #4,R2          ;SEE IF RESULT CLEAR, 0.
21341 110152 012701 110204      MOV      #AABBF0,R1
21342 110156 005721      3$:    TST      (R1)      ;BRANCH IF RESULT INCORRECT, NOT 0
21343 110160 001010              BNE     AAB2
21344 110162 077203              SOB     R2,3$
21345 110164 022775 000204      CMP     #204,R5        ;SEE IF FPS IS CORRECT.
21346 110170 001004              BNE     AAB2            ;BRANCH IF INCORRECT.
21347 110172 020027 110204      CMP     R0,#AABBF0    ;SEE IF R0 IS CORRECT.
21348 110176 001001              BNE     AAB2            ;BRANCH IF R0 IS INCORRECT.
21349 110200 000411              BR      AABDONE
21350
21351
21352 110202      AAB2:  EMT              ;
      (2) 110202 104000
21353
21354      ;THIS IS THE TEST DATA BUFFER, OUTPUT DATA BUFFER.
21355 110204 073475      AABBF0: 73475
21356 110206 067707              67707
21357 110210 127347              127347
21358 110212 056770              56770
21359      ;THIS IS THE DATA USED TO SET UP THE OUTPUT BUFFER.
21360 110214 073475      AABTP1: 73475
21361 110216 067707              67707
21362 110220 127347              127347
21363 110222 056770              56770
21364 110224      AABDONE:
      (1) 110224 004767 014322      JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                          ;SEE IF THE USER HAS EXPRESSED
      (1)                          ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                          ;THE USER TYPED CONTROL G?).
21365
21371      ;*****
      (2)      ;TEST 527      CLRD WITH ILLEGAL ACCUMULATOR TEST
      (3)      ;*****
      (2) 110230      TS527:
21372 110230 012700 040200      MOV      #40200,R0     ;SET UP THE FPS, NO INTERRUPTS AND FD=1.
21373 110234 170100              LDFPS   R0
21374 110236 170407      CCB2:  CLRD      AC7     ;TEST INSTRUCTION.
21375
21376 110240 170204              STFPS   R4            ;GET FPS.
21377 110242 170305              STST    R5            ;GET FEC.
21378 110244 020427 140200      CMP     R4,#140200    ;IS THE FPS CORRECT?
21379 110250 001004              BNE     CCB10         ;BRANCH IF FPS IS INCORRECT.
21380 110252 022705 000002      CMP     #2,R5        ;IS THE FEC CORRECT?
21381 110256 001001              BNE     CCB10         ;BRANCH IF FEC IS INCORRECT.
21382 110260 000401              BR      CCBDONE
21383
21384 110262      CCB10:
      (2) 110262 104000      EMT              ;
```

21385 110264
 (1) 110264 004767 014262
 (1)
 (1)
 (1)
 (1)
 (1)
 21394
 21395
 (2)
 (3)
 (2) 110270
 21396
 21397 110270 012700 040200
 21398 110274 170100
 21399 110276 170707
 21400
 21401 110300 170204
 21402 110302 170305
 21403
 21404 110304 022704 140200
 21405 110310 001004
 21406 110312 022705 000002
 21407 110316 001001
 21408 110320 000401
 21409 110322
 (2) 110322 104000
 21410
 21411 110324
 (1) 110324 004767 014222
 (1)
 (1)
 (1)
 (1)
 (1)
 21412
 21420
 (2)
 (3)
 (2) 110330
 21421
 21422 110330 012700 000200
 21423 110334 170100
 21424 110336 012700 110430
 21425 110342 172410
 21426 110344 005000
 21427 110346 170100
 21428 110350 012700 110440
 21429 110354 172410
 21430
 21431 110356 012700 000201
 21432 110362 170100
 21433 110364 170700
 21434
 21435 110366 170205
 21436 110370 012700 000200
 21437 110374 170100
 21438 110376 012700 110450

```

CCBDONE:
      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                                ;SEE IF THE USER HAS EXPRESSED
                                ;THE DESIRE TO CHANGE THE SOFTWARE
                                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                ;THE USER TYPED CONTROL G?).

;*****
;TEST 530      NEGF, ABSF AND TSTF SOURCE MODE 0 WITH ILLEGAL AC7, TEST
;*****
TS530:
      MOV      #40200,R0      ;SET UP THE FPS, FID=1 AND FD=1.
      LDFPS   R0
      VVB2:   NEG D          AC7      ;TEST INSTRUCTION.

      STFPS   R4              ;GET FPS.
      STST    R5              ;GET FEC.

      CMP     #140200,R4      ;IS FPS CORRECT?
      BNE     VVB10           ;BRANCH IF FPS IS INCORRECT.
      CMP     #2,R5           ;IS FEC CORRECT?
      BNE     VVB10           ;BRANCH IF FEC IS INCORRECT.
      BR      VVBDONE

VVB10:
      EMT

VVBDONE:
      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                                ;SEE IF THE USER HAS EXPRESSED
                                ;THE DESIRE TO CHANGE THE SOFTWARE
                                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                ;THE USER TYPED CONTROL G?).

;*****
;TEST 531      NEGF, ABSF AND TSTF SOURCE MODE 0 TEST
;*****
TS531:
      MOV      #200,R0        ;SET FD MODE.
      LDFPS   R0
      MOV     #DDBTP1,R0     ;SET UP ACO.
      LDD     (R0),ACO       ;SET ACO = 0
      CLR     R0              ;CLEAR THE FPS.
      LDFPS   R0
      MOV     #DDBTP2,R0     ;LOAD ACO TO BE A FLOATING 0.
      LDF     (R0),ACO       ;SET ACO=ZERO
                                ;FLOAT
                                ;SET FD MODE.
      DDB2:   NEG D          ACO      ;TEST INSTRUCTION

      STFPS   R5              ;GET FPS.
      MOV     #200,R0        ;SET FD MODE.
      LDFPS   R0
      MOV     #DDBBF0,R0     ;GET THE RESULT OUT OF ACO
  
```

CJKL580 LCP-5 CPU CLSTR DIAG
C JKL58 P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 28-17
T531 NEGF, ABSF AND TSTF SOURCE MODE 0 TEST

SEQ 0373

21439 110402 174010
 21440
 21441 110404 012701 000004
 21442 110410 005720
 21443 110412 001005
 21444 110414 077103
 21445 110416 022705 000204
 21446 110422 001001
 21447 110424 000415
 21448 110426
 (2) 110426 104000
 21449
 21450
 21451 110430 101112
 21452 110432 131415
 21453 110434 161710
 21454 110436 111213
 21455 110440 000000
 21456 110442 000000
 21457 110444 000000
 21458 110446 000000
 21459
 21460 110450 177777
 21461 110452 177777
 21462 110454 177777
 21463 110456 177777
 21464
 21465 110460
 (1) 110460 004767 014066
 (1)
 (1)
 (1)
 (1)
 (1)
 21466
 21467
 (2)
 (3)
 (2) 110464
 21468
 21469 110464 012700 110564
 21470 110470 012701 110604
 21471 110474 012702 000004
 21472 110500 012021
 21473 110502 077202
 21474 110504 012700 000200
 21475 110510 170100
 21476 110512 012700 110604
 21477 110516 012737 110614 000004
 21478 110524 170710
 21479
 21480 110526 170205
 21481 110530 012701 110604
 21482 110534 012702 000004
 21483 110540 005721
 21484 110542 001024
 21485 110544 077203

STD ACO.(R0)
 ;SEE IF THE RESULT IS CORRECT.
 1\$: MOV #4,R1
 TST (R0)+
 BNE DOB5 ;BRANCH IF THE RESULT IS INCORRECT.
 SOB R1,1\$
 CMP #204,R5 ;IS THE FPS CORRECT?
 BNE DOB5 ;BRANCH IF THE FPS IS INCORRECT.
 BR DOBDONE
 DOB5: EMT ;
 ;THESE ARE TEST DATA TABLES AND AN OUTPUT BUFFER.
 DOBTP1: 101112
 131415
 161710
 111213
 DOBTP2: 0
 0
 0
 0
 DOBBF0: -1
 -1
 -1
 -1
 DOBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

 ;TEST 532 NEGF, ABSF AND TSTF SOURCE MODE 1 TEST

T5532:
 MOV #EEBTP1,R0 ;SET UP THE DATA BUFFER.
 MOV #EEBBF1,R1
 MOV #4,R2
 1\$: MOV (R0)+,(R1)+
 SOB R2,1\$
 MOV #200,R0 ;SET FD MODE.
 LDFPS R0
 MOV #EEBBF1,R0 ;SET UP THE OPERAND ADDRESS.
 MOV #EEB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF ERROR.
 EEB2: NEG0 (R0) ;TEST INSTRUCTION.
 STFPS R5 ;GET FPS.
 MOV #EEBBF1,R1 ;SEE IF RESULT IS CORRECT.
 MOV #4,R2
 1\$: TST (R1)+
 BNE EEB10 ;BRANCH IF NOT CORRECT.
 SOB R2,1\$

21486
 21487 110546 020027 110604
 21488 110552 001020
 21489 110554 022705 000204
 21490 110560 001015
 21491 110562 000415
 21492
 21493
 21494 110564 000177
 21495 110566 167574
 21496 110570 137271
 21497 110572 107675
 21498 110574 177777
 21499 110576 177777
 21500 110600 177777
 21501 110602 177777
 21502 110604 177777
 21503 110606 177777
 21504 110610 177777
 21505 110612 177777
 21506 110614
 (2) 110614 104000
 21507 110616
 (1) 110616 004767 013730
 (1)
 (1)
 (1)
 (1)
 (1)
 21508
 21509
 (2)
 (3)
 (2) 110622
 21510
 21511 110622 012700 110722
 21512 110626 012701 110732
 21513 110632 012702 000004
 21514 110636 012021
 21515 110640 077202
 21516 110642 012700 000200
 21517 110646 170100
 21518 110650 012700 110732
 21519 110654 012737 110742 000004
 21520
 21521 110662 170620
 21522
 21523 110664 170205
 21524 110666 012701 110732
 21525 110672 012702 000004
 21526 110676 005721
 21527 110700 001020
 21528 110702 077203
 21529
 21530 110704 020027 110742
 21531 110710 001014
 21532 110712 022705 000204

```

CMP      RO,#EEBBF1      ;IS RO CORRECT?
BNE      EEB10            ;BRANCH IF NOT CORRECT.
CMP      #204,R5         ;IS THE FPS CORRECT?
BNE      EEB10            ;BRANCH IF NOT CORRECT.
BR       EEBDONE

;THESE ARE TEST DATA TABLES AND A BUFFER.
EEBTP1: 177
        167574
        137271
        107675
EEBBF0: 1
        -1
        -1
        1
EEBBF1: -1
        1
        -1
        -1
EEB10:  EMT
EEBDONE: JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 533      NEGF, ABSF AND TSTF SOURCE MODE 2 TEST
;*****
T533:

        MOV      #FFBTP1,R0      ;SET UP THE DATA BUFFER.
        MOV      #FFBBF1,R1
        MOV      #4,R2
1$:     MOV      (R0)+,(R1)+
        SOB      R2,1$
        MOV      #200,R0      ;SET FD.
        LDFPS   R0
        MOV      #FFBBF1,R0      ;SET UP THE OPERAND ADDRESS.
        MOV      #FFB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.

FFB2:   ABSD      (R0)+          ;TEST INSTRUCTION.

        STFPS   R5              ;GET FPS.
        MOV      #FFBBF1,R1      ;CHECK RESULT.
        MOV      #4,R2
1$:     TST      (R1)+
        BNE      FFB10          ;BRANCH IF INCORRECT.
        SOB      R2,1$

        CMP      RO,#FFBBF1+10  ;IS RO CORRECT?
        BNE      FFB10          ;BRANCH IF INCORRECT.
        CMP      #204,R5         ;IS THE FPS CORRECT?

```

21533 110716 001011
 21534 110720 000411
 21535
 21536
 21537 110722 000177
 21538 110724 167574
 21539 110726 137271
 21540 110730 107675
 21541 110732 177777
 21542 110734 177777
 21543 110736 177777
 21544 110740 177777
 21545 110742
 (2) 110742 104000
 21546 110744
 (1) 110744 004767 013602
 (1)
 (1)
 (1)
 (1)
 21547
 (2)
 (3)
 (2) 110750

BNE FFB10 ;BRANCH IF INCORRECT.
 BR FFBONE
 ;THESE ARE TEST DATA TABLES AND DATA BUFFER.
 FFBTP1: 177
 167574
 137271
 107675
 FFBDF1: -1
 -1
 -1
 1
 FFB10: EMT ;
 FFBONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).
 ;*****
 ;TEST 534 NEGF, ABSF AND TSTF SOURCE MODE 4 TEST
 ;*****
 T5534:


```
21596 111124 077202 SOB R2,11
21597 111126 012700 000200 MOV #200,R0 ;SET FD.
21598 111132 170100 LDFPS R0
21599 111134 012700 111236 MOV #HBBF1,R0 ;SET UP THE OPERAND ADDRESS.
21600 111140 012737 111246 G00004 MOV #HBB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
21601
21602 111146 170630 HMB2: ABSD B(R0). ;TEST INSTRUCTION.
21603
21604 111150 170205 STFPS R5 ;GET FPS.
21605 111152 012701 111226 MOV #HBBF0,R1 ;CHECK RESULT.
21606 111156 012702 000004 MOV #4,R2
21607 111162 005721 11: TST (R1).
21608 111164 001030 BNE HMB10 ;BRANCH IF INCORRECT.
21609 111166 077203 SOB R2,11
21610 111170 020027 111240 CMP R0,#HBBF1*2 ;IS R0 CORRECT?
21611 111174 001024 BNE HMB10 ;BRANCH IF INCORRECT.
21612 111176 022705 000204 CMP #204,R5 ;IS THE FPS CORRECT?
21613 111202 001021 BNE HMB10 ;BRANCH IF INCORRECT.
21614 111204 000421 BR HMBDONE
```

21615
21616 ;THESE ARE TEST DATA TABLES AND DATA BUFFER.

```
HMBTP1: 177
147576
177071
21620 111214 107576 111226 177777 107576,HBBF0,-1,-1,-1
111222 177777 177777
```

```
21621 111226 177777 HMBBF0: -1
21622 111230 177777 -1
21623 111232 177777 -1
21624 111234 177777 -1
21625 111236 177777 HMBBF1: -1
21626 111240 177777 -1
21627 111242 177777 -1
21628 111244 177777 -1
```

```
21629 111246 HMB10:
(2) 111246 104000 EMT ;
21630 111250 HMBDONE:
(1) 111250 004767 013276 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
```

```
21631 ;*****
(2) ;TEST 536 NEGF, ABSF AND TSTF SOURCE MODE 5 TEST
(3) ;*****
(2) 111254 T536:
```

```
21632
21633 111254 012700 111354 MOV #IIBTP1,R0 ;SET UP THE DATA BUFFER.
21634 111260 012701 111374 MOV #IIBBF0,R1
21635 111264 012702 000010 MOV #10,R2
21636 111270 012021 11: MOV (R0).,(R1).
21637 111272 077202 SOB R2,11
21638 111274 012700 000200 MOV #200,R0 ;SET FD.
21639 111300 170100 LDFPS R0
21640 111302 012700 111406 MOV #IIBBF1*2,R0 ;SET UP THE OPERAND ADDRESS.
21641 111306 012737 111414 000004 MOV #IIB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
```

```
21642
21643 111314 170750 IIB2: NEG0 0 (R0) ;TEST INSTRUCTION
21644
21645 111316 170205 STFPS R5 ;GET FPS.
21646 111320 012701 111374 MOV #IIBBF0,R1 ;CHECK RESULT
21647 111324 012702 000004 MOV #4,R2
21648 111330 005721 I+: TST (R1).
21649 111332 001030 BNE IIB10 ;BRANCH IF INCORRECT.
21650 111334 077203 SOB R2,1#
21651 111336 020027 111404 CMP R0,#IIBBF1 ;IS R0 CORRECT?
21652 111342 001024 BNE IIB10 ;BRANCH IF INCORRECT.
21653 111344 022705 000204 CMP #204,R5 ;IS THE FPS CORRECT?
21654 111350 001021 BNE IIB10 ;BRANCH IF INCORRECT.
21655 111352 000421 BR IIBDONE
21656
21657 ;THESE ARE TEST DATA TABLES AND DATA BUFFER.
21658 111354 000176 IIBTP1: 176
21659 111356 177074 177074
21660 111360 127374 127374
21661 111362 157677 111374 177777 157677,IIBBF0,-1,-1,-1
21662 111370 177777 177777
21663 111374 177777 IIBBF0: -1
21664 111376 177777 -1
21665 111400 177777 -1
21666 111402 177777 IIBBF1: -1
21667 111404 177777 -1
21668 111406 177777 -1
21669 111410 177777 -1
21670 111412 177777 -1
21671 111414 IIB10:
21672 (2) 111414 104000 IIBDONE: EMT ;
21673 (1) 111416 004767 013130 JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
21674 ;*****
21675 (2) ;TEST 537 NEGF, ABSF AND TSTF SOURCE MODE 6 TEST
21676 (3) ;*****
21677 (2) 111422 TS537:
21678 111422 012700 111524 MOV #JJBTP1,R0 ;SET UP THE DATA BUFFER.
21679 111426 012701 111536 MOV #JJBBF0,R1
21680 111432 012702 000004 MOV #4,R2
21681 111436 012021 I+: MOV (R0),.(R1).
21682 111440 077202 SOB R2,1#
21683 111442 012700 000200 MOV #200,R0 ;SET FD.
21684 111446 170100 LDFPS R0
21685 111450 012700 111527 MOV #JJBBF0-7,R0 ;SET UP THE OPERAND ADDRESS.
21686 111454 012737 111546 000004 MOV #JJB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
21687 111462 170660 000007 JJB2: ABS0 7(R0) ;TEST INSTRUCTION.
21688 111466 170205 STFPS R5 ;GET FPS.
```

21688 111470 012701 111536
 21689 111474 012702 000004
 21690 111500 005721
 21691 111502 001021
 21692 111504 077203
 21693 111506 020027 111527
 21694 111512 001015
 21695 111514 022705 000204
 21696 111520 001012
 21697 111522 000412
 21698
 21699
 21700 111524 000177
 21701 111526 161524
 21702 111530 131273
 21703 111532 107174 000000
 21704 111536 177777
 21705 111540 177777
 21706 111542 177777
 21707 111544 177777
 21708 111546
 (2) 111546 104000
 21709 111550
 (1) 111550 004767 012776
 (1)
 (1)
 (1)
 (1)
 (1)
 21710
 (2)
 (3)
 (2) 111554
 21711
 21712 111554 012700 111656
 21713 111560 012701 111676
 21714 111564 012702 000010
 21715 111570 012021
 21716 111572 077202
 21717 111574 012700 000200
 21718 111600 170100
 21719 111602 012700 111677
 21720 111606 012737 111716 000004
 21721
 21722 111614 170770 000007
 21723
 21724 111620 170205
 21725 111622 012701 111676
 21726 111626 012702 000004
 21727 111632 005721
 21728 111634 001030
 21729 111636 077203
 21730 111640 020027 111677
 21731 111644 001024
 21732 111646 022705 000204
 21733 111652 001021
 21734 111654 000421

```

MOV #JBBF0,R1 ;CHECK RESULT.
MOV #4,R2
11: TST (R1).
BNE JJB10 ;BRANCH IF INCORRECT.
SOB R2,11
CMP RO,#JBBF0-7 ;IS RO CORRECT?
BNE JJB10 ;BRANCH IF INCORRECT.
CMP #204,R5 ;IS THE FPS CORRECT?
BNE JJB10 ;BRANCH IF INCORRECT.
BR JBDONE

;THESE ARE TEST DATA TABLES AND DATA BUFFER.
JJBTP1: 177
        161524
        131273
        107174,
JBBF0: 1
        -1
        -1
        -1
JJB10: EMT
JBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 540 NEGF, ABSF AND TSTF SOURCE MODE 7 TEST
;*****
T5540:
MOV #KKBTP1,RO ;SET UP THE DATA BUFFER.
MOV #KKBBF0,R1
MOV #10,R2
11: MOV (RO),.(R1).
SOB R2,11
MOV #200,RO ;SET FD.
LDFPS RO
MOV #KKBBF1-7,RO ;SET UP THE OPERAND ADDRESS.
MOV #KKB10,#ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR
KKB2: NEG0 #7(RO) ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #KKBBF0,R1 ;CHECK RESULT.
MOV #4,R2
11: TST (R1).
BNE KKB10 ;BRANCH IF INCORRECT.
SOB R2,11
CMP RO,#KKBBF1-7 ;IS RO CORRECT?
BNE KKB10 ;BRANCH IF INCORRECT.
CMP #204,R5 ;IS THE FPS CORRECT?
BNE KKB10 ;BRANCH IF INCORRECT.
BR KBDONE

```

21735
21736
21737 111656 000177
21738 111660 167574
21739 111662 137271
21740 111664 107675 111676 177777
111672 177777 177777
21741 111676 177777
21742 111700 177777
21743 111702 177777
21744 111704 177777
21745 111706 177777
21746 111710 177777
21747 111712 177777
21748 111714 177777
21749 111716
(2) 111716 104000
21750 111720
(1) 111720 004767 012626
(1)
(1)
(1)
(1)
21751
(2)
(3)
(2) 111724
21752 111724 012700 112014
21753 111730 012701 112024
21754 111734 012702 000004
21755 111740 012021
21756 111742 077202
21757 111744 012700 000200
21758 111750 170100
21759 111752 012737 112034 000004
21760
21761 111760 170767 000040
21762
21763 111764 170205
21764 111766 012701 112024
21765 111772 012702 000004
21766 111776 005721
21767 112000 001015
21768 112002 077203
21769 112004 022705 000204
21770 112010 001011
21771 112012 000411
21772
21773
21774 112014 000127
21775 112016 137475
21776 112020 147372
21777 112022 117057
21778 112024 177777
21779 112026 177777
21780 112030 177777

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

KKBT1: 177
167574
137271
107675, KKBBF0, 1, 1, 1

KKBBF0: 1
1
-1
-1
KKBBF1: 1
1
1
-1
KKB10:
EMT
KKBDONE:

JSR PC, .RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 541 NEGF, ABSF AND TSTF SOURCE MODE 6, GR7, TEST
;*****

TS541:
MOV #LLBTP1, R0 ;SET UP THE DATA BUFFER.
MOV #LLBBF0, R1
MOV #4, R2
14: MOV (R0)+, (R1)+
SOB R2, 14
MOV #200, R0 ;SET FD.
LDFPS R0
MOV #LLB10, @ERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
LLB2: NEG0 LLBBF0 ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #LLBBF0, R1 ;CHECK RESULT.
MOV #4, R2
14: TST (R1)+
BNE LLB10 ;BRANCH IF INCORRECT.
SOB R2, 14
CMP #204, R5 ;IS THE FPS CORRECT?
BNE LLB10 ;BRANCH IF INCORRECT.
BR LLBDONE

;THESE ARE TEST DATA TABLES AND DATA BUFFER.

LLBTP1: 127
137475
147372
117057
LLBBF0: -1
1
1

```
21781 112032 177777 1
21782
21783 112034 LLB10:
(2) 112034 104000 EMT
21784 112036 LLBDONE:
(1) 112036 004767 012510 JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
21785 ;*****
(2) ;TEST 542 NEGF, ABSF AND TSTF SOURCE MODE 7, GR7, TEST
(3) ;*****
(2) 112042 TS542:
21786 MOV #MMBTP1,R0 ;SET UP THE DATA BUFFER.
21787 112042 012700 112132 MOV #MMBBF0,R1
21788 112046 012701 112152 MOV #10,R2
21789 112052 012702 000010 MOV (R0),.(R1).
21790 112056 012021 11: SOB R2,1#
21791 112060 077202 SOB R2,1#
21792 112062 012700 000200 MOV #200,R0 ;SET FD.
21793 112066 170100 LDFPS R0
21794 112070 012737 112172 000004 MOV #MMB10,#MERRVECT ;SET UP VECTOR 4 IN CASE OF AN ERROR.
21795
21796 112076 170677 000060 MMB2: ABSD #MMBBF1 ;TEST INSTRUCTION.
21797
21798 112102 170205 STFPS R5 ;GET FPS.
21799 112104 012701 112152 MOV #MMBBF0,R1 ;CHECK RESULT.
21800 112110 012702 000004 MOV #4,R2
21801 112114 005721 11: TST (R1).
21802 112116 001025 BNE MMB10 ;BRANCH IF INCORRECT
21803 112120 077203 SOB R2,1#
21804 112122 022705 000204 CMP #204,R5 ;IS THE FPS CORRECT?
21805 112126 001021 BNE MMB10 ;BRANCH IF INCORRECT.
21806 112130 000421 BR MMBDONE
21807
21808 ;THESE ARE TEST DATA TABLES AND DATA BUFFER.
21809 112132 000137 MMBTP1: 137
21810 112134 045607 045607
21811 112136 101230 101230
21812 112140 045607 112152 177777 45607,MMBBF0, 1,-1, 1
112146 177777 177777
21813 112152 177777 MMBBF0: -1
21814 112154 177777 -1
21815 112156 177777 -1
21816 112160 177777 -1
21817 112162 177777 MMBBF1: -1
21818 112164 177777 -1
21819 112166 177777 -1
21820 112170 177777 -1
21821
21822 112172 MMB10:
(2) 112172 104000 EMT
21823 112174 LLBDONE:
(1) 112174 004767 012352 JSR PC..RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
```


614

(1)
(1)
(1)
21830
(2)
(3)
(2) 112200
21831
21832 112200 012700 000200
21833 112204 170100
21834 112206 012700 112266
21835 112212 172410
21836 112214 170700
21837
21838 112216 170205
21839 112220 012700 000200
21840 112224 170100
21841 112226 012700 112306
21842 112232 174010
21843 112234 012700 112306
21844 112240 012701 112276
21845 112244 012702 000004
21846 112250 022021
21847 112252 001021
21848 112254 077203
21849 112256 022705 000210
21850 112262 001015
21851 112264 000415
21852
21853
21854 112266 013572
21855 112270 046013
21856 112272 057246
21857 112274 013570
21858 112276 113572
21859 112300 046013
21860 112302 057246
21861 112304 013570
21862 112306 000000
21863 112310 000000
21864 112312 000000
21865 112314 000000
21866
21867 112316
(2) 112316 104000
21868 112320
(1) 112320 004767 012226
(1)
(1)
(1)
(1)
21869
(2)
(3)
(2) 112324
21870

```

;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 543      SPECIAL DEST, MODE 0, TEST
;*****
T5543:
      MOV      #200,R0      ;SET FD.
      LDFPS   R0
      MOV      #NNBTP1,R0      ;SET UP ACO.
      LDO      (R0),ACO
      NNB2:    NEG0      ACO      ;TEST INSTRUCTION.
      STFPS   R5      ;GET FPS.
      MOV      #200,R0      ;SET FD.
      LDFPS   R0
      MOV      #NNBBF0,R0      ;GET THE RESULT.
      STD      ACO,(R0)
      MOV      #NNBBF0,R0      ;IS THE RESULT CORRECT?
      MOV      #NNBTP2,R1
      MOV      #4,R2
1$:    CMP      (R0),.(R1)+
      BNE     NNB10      ;BRANCH IF INCORRECT.
      SOB     R2,1$
      CMP      #210,R5      ;IS THE FPS CORRECT?
      BNE     NNB10      ;BRANCH IF INCORRECT.
      BR      NNBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
NNBTP1: 013572
        46013
        57246
        013570
NNBTP2: 113572
        46013
        57246
        013570
NNBBF0: 0
        0
        0
        0
NNB10:
NNBDONE: EMT
        JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                                ;SEE IF THE USER HAS EXPRESSED
                                ;THE DESIRE TO CHANGE THE SOFTWARE
                                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                                ;THE USER TYPED CONTROL G?).
;*****
;TEST 544      SPECIAL DEST, MODE 1, TEST
;*****
T5544:

```

21871 112324 012701 112426
 21872 112330 012700 112436
 21873 112334 012702 000004
 21874 112340 C12021
 21875 112342 077202
 21876 112344 012700 112426
 21877 112350 042710 100000
 21878 112354 012701 000200
 21879 112360 170101
 21880
 21881 112362 170710
 21882 112364 170205
 21883 112366 012701 112426
 21884 112372 012702 112436
 21885 112376 012703 000004
 21886 112402 022122
 21887 112404 001020
 21888 112406 077303
 21889 112410 022700 112426
 21890 112414 001014
 21891 112416 022705 000210
 21892 112422 001011
 21893 112424 000411
 21894
 21895
 21896 112426 023245
 21897 112430 026720
 21898 112432 122324
 21899 112434 052672
 21900 112436 123245
 21901 112440 026720
 21902 112442 122324
 21903 112444 052672
 21904
 21905 112446
 (2) 112446 104000
 21906 112450
 (1) 112450 004767 012076
 (1)
 (1)
 (1)
 (1)
 21907
 (2)
 (3)
 (2) 112454
 21908
 21909 112454 012701 112556
 21910 112460 012700 112566
 21911 112464 012702 000004
 21912 112470 012021
 21913 112472 077202
 21914 112474 012700 112556
 21915 112500 042710 100000
 21916 112504 012701 000200
 21917 112510 170101

```

MOV #008TP1,R1 ;SET UP THE DATA BUFFER.
MOV #008TP2,R0
MOV #4,R2
14: MOV (R0)+,(R1)+
SOB R2,14
MOV #008TP1,R0
BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

0082: NEG0 (R0) ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #008TP1,R1 ;IS THE RESULT CORRECT.
MOV #008TP2,R2
MOV #4,R3
14: CMP (R1)+,(R2)+
BNE 00810 ;BRANCH IF INCORRECT.
SOB R3,14
CMP #008TP1,R0 ;IS R0 CORRECT.
BNE 00810 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE 00810 ;BRANCH IF INCORRECT.
BR 008DONE

```

;THESE ARE DATA TABLES AND A DATA BUFFER.

```

008TP1: 023245
        26720
        122324
        52672
008TP2: 123245
        26720
        122324
        52672

```

00810:

EMT ;

008DONE:

JSR PC,.RSET

```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

```

```

;*****
;TEST 545 SPECIAL DEST, MODE 2, TEST
;*****
TSS45:

```

```

MOV #PPBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #PPBTP2,R0
MOV #4,R2
14: MOV (R0)+,(R1)+
SOB R2,14
MOV #PPBTP1,R0
BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1

```

21918
21919 112512 170720
21920
21921 112514 170205
21922 112516 012701 112556
21923 112522 012702 112566
21924 112526 012703 000004
21925 112532 022122
21926 112534 001020
21927 112536 077303
21928 112540 022700 112566
21929 112544 001014
21930 112546 022705 000210
21931 112552 001011
21932 112554 000411
21933
21934
21935 112556 023245
21936 112560 026720
21937 112562 122324
21938 112564 052672
21939 112566 123245
21940 112570 026720
21941 112572 122324
21942 112574 052672
21943
21944 112576
(2) 112576 104000
21945 112600
(1) 112600 004767 011746
(1)
(1)
(1)
(1)
(1)
21946
(2)
(3)
(2) 112604
21947 112604 012701 112710
21948 112610 012700 112730
21949 112614 012702 000004
21950 112620 012021
21951 112622 077202
21952 112624 012700 112720
21953 112630 042760 100000 177770
21954 112636 012701 000200
21955 112642 170101
21956
21957 112644 170740
21958
21959 112646 170205
21960 112650 012701 112710
21961 112654 012702 112730
21962 112660 012703 000004
21963 112664 022122
21964 112666 001024

PPB2: NEG0 (R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.
MOV @PPBTP1,R1 ;IS THE RESULT CORRECT.
MOV @PPBTP2,R2
MOV #4,R3
14: CMP (R1)+,(R2)+
BNE PPB10 ;BRANCH IF INCORRECT.
SOB R3,14
CMP @PPBTP1+10,R0 ;IS R0 CORRECT.
BNE PPB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE PPB10 ;BRANCH IF INCORRECT.
BR PPBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.

PPBTP1: 023245
26720
122324
52672
PPBTP2: 123245
26720
122324
52672

PPB10:

EMT ;
PPBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;TEST 546 SPECIAL DEST. MODE 4. TEST

T5546:

MOV @QOBTP1,R1 ;SET UP THE DATA BUFFER.
MOV @QOBTP2,R0
MOV #4,R2
14: MOV (R0)+,(R1)+
SOB R2,14
MOV @QOBTP1+10,R0
BIC #100000,-10(R0) ;MAKE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1
QOB2: NEG0 -(R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS.
MOV @QOBTP1,R1 ;IS THE RESULT CORRECT.
MOV @QOBTP2,R2
MOV #4,R3
14: CMP (R1)+,(R2)+
BNE QOB10 ;BRANCH IF INCORRECT.

21965 112670 077303
21966 112672 022700 112710
21967 112676 001020
21968 112700 022705 000210
21969 112704 001015
21970 112706 000415
21971
21972
21973 112710 023245
21974 112712 026720
21975 112714 122324
21976 112716 052672
21977 112720 177777 177777 177777
112726 177777
21978 112730 123245
21979 112732 026720
21980 112734 122324
21981 112736 052672
21982
21983 112740
(2) 112740 104000
21984 112742
(1) 112742 004767 011604
(1)
(1)
(1)
(1)
(1)
21985
21986
(2)
(3)
(2) 112746
21987
21988 112746 012701 113056
21989 112752 012700 113066
21990 112756 012702 000004
21991 112762 012021
21992 112764 077202
21993 112766 012700 113076
21994 112772 012710 113056
21995 112776 042737 100000 113056
21996 113004 012701 000200
21997 113010 170101
21998
21999 113012 170730
22000
22001 113014 170205
22002 113016 012701 113056
22003 113022 012702 113066
22004 113026 012703 000004
22005 113032 022122
22006 113034 001021
22007 113036 077303
22008 113040 022700 113100
22009 113044 001015
22010 113046 022705 000210

SOB R3,1#
CMP #QQBTP1,R0 ;IS R0 CORRECT
BNE QQB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE QQB10 ;BRANCH IF INCORRECT.
BR QQBDONE

;THESE ARE DATA TABLES AND A DATA BUFFER.
QQBTP1: 023245
26720
122324
52672
.WORD -1, 1, 1, -1
QQBTP2: 123245
26720
122324
52672
QQB10:
EMT ;
QQBDONE: JSR PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 547 SPECIAL DEST. MODE 3, TEST
;*****
TS547:
MOV #RRBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #RRBTP2,R0
MOV #4,R2
1#: MOV (R0)+,(R1)+
SOB R2,1#
MOV #RRBTP3,R0
MOV #RRBTP1,(R0)
BIC #100000,#RRBTP1 ;MAKE THE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1
RRB2: NEGD @R0+ ;TEST INSTRUCTION.
STFPS R5 ;GET FPS.
MOV #RRBTP1,R1 ;IS THE RESULT CORRECT.
MOV #RRBTP2,R2
MOV #4,R3
1#: CMP (R1)+,(R2)+
BNE RRB10 ;BRANCH IF INCORRECT.
SOB R3,1#
CMP #RRBTP3+2,R0 ;IS R0 CORRECT.
BNE RRB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?

22011 113052 001012 BNE RRB10 ;BRANCH IF INCORRECT
22012 113054 000412 BR RRB DONE
22013
22014 ;THESE ARE DATA TABLES AND A DATA BUFFER.
22015 113056 023245 RRBTP1: 023245
22016 113060 026720 26720
22017 113062 122324 122324
22018 113064 052672 52672
22019 113066 123245 RRBTP2: 123245
22020 113070 026720 26720
22021 113072 123324 123324
22022 113074 052672 52672
22023 113076 113056 RRBTP3: RRBTP1
22024
22025 113100 RRB10:
(2) 113100 104000 EMT ;
22026 113102 RRB DONE:
(1) 113102 004767 011444 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
22027
22028 ;*****
(2) ;TEST 550 SPECIAL DEST. MODE 5, TEST
(3) ;*****
(2) T550:
22029 113106 012701 113220 MOV #SSBTP1,R1 ;SET UP THE DATA BUFFER.
22030 113112 012700 113230 MOV #SSBTP2,R0
22031 113116 012702 000004 MOV #4,R2
22032 113122 012021 14: MOV (R0)+,(R1)+
22033 113124 077202 SOB R2,14
22034 113126 012700 113242 MOV #SSBTP3+2,R0
22035 113132 012760 113220 177776 MOV #SSBTP1,-2(R0)
22036 113140 042737 100000 113220 BIC #100000,#SSBTP1 ;MAKE THE OPERAND POSITIVE.
22037 113146 012701 000200 MOV #200,R1 ;SET FD.
22038 113152 170101 LDFPS R1
22039
22040 113154 170750 SSB2: NEGD @-(R0) ;TEST INSTRUCTION.
22041
22042 113156 170205 STFPS R5 ;GET FPS.
22043 113160 012701 113220 MOV #SSBTP1,R1 ;IS THE RESULT CORRECT.
22044 113164 012702 113230 MOV #SSBTP2,R2
22045 113170 012703 000004 MOV #4,R3
22046 113174 022122 14: CMP (R1)+,(R2)+
22047 113176 001021 BNE SSB10 ;BRANCH IF INCORRECT.
22048 113200 077303 SOB R3,14
22049 113202 022700 113240 CMP #SSBTP3,R0 ;IS R0 CORRECT.
22050 113206 001015 BNE SSB10 ;BRANCH IF INCORRECT.
22051 113210 022705 000210 CMP #210,R5 ;IS THE FPS CORRECT?
22052 113214 001012 BNE SSB10 ;BRANCH IF INCORRECT.
22053 113216 000412 BR SSB DONE
22054
22055 ;THESE ARE DATA TABLES AND A DATA BUFFER.
22056 113220 023245 SSBTP1: 023245
22057 113222 026720 26720

22058 113224 122324
 22059 113226 052672
 22060 113230 123245
 22061 113232 026270
 22062 113234 122324
 22063 113236 052672
 22064 113240 113220
 22065
 22066 113242
 (2) 113242 104000
 22067 113244
 (1) 113244 004767 011302
 (1)
 (1)
 (1)
 (1)
 22068
 (2)
 (3)
 (2) 113250
 22069 113250 012701 113352
 22070 113254 012700 113362
 22071 113260 012702 000004
 22072 113264 012021
 22073 113266 077202
 22074 113270 012700 113352
 22075 113274 042710 100000
 22076 113300 012701 000000
 22077 113304 170101
 22078
 22079 113306 170720
 22080
 22081 113310 170205
 22082 113312 012701 113352
 22083 113316 012702 113362
 22084 113322 012703 000004
 22085 113326 022122
 22086 113330 001020
 22087 113332 077303
 22088 113334 022700 113356
 22089 113340 001014
 22090 113342 022705 000010
 22091 113346 001011
 22092 113350 000411
 22093
 22094
 22095 113352 023245
 22096 113354 026720
 22097 113356 122324
 22098 113360 052672
 22099 113362 123245
 22100 113364 026720
 22101 113366 122324
 22102 113370 052672
 22103
 22104 113372

122324
 52672
 SSBTP2: 123245
 26270
 122324
 52672
 SSBTP3: SSBTP1
 SSB10:
 EMT ;
 SSBDONE: JSR PC, RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).
 ;*****
 ;TEST 551 SPECIAL DEST, FLOATING MODE 2, TEST
 ;*****
 T551:
 MOV #TTBTP1,R1 ;SET UP THE DATA BUFFER.
 MOV #TTBTP2,R0
 MOV #4,R2
 1\$: MOV (R0)+,(R1)+
 SOB R2,1\$
 MOV #TTBTP1,R0
 BIC #100000,(R0) ;MAKE OPERAND POSITIVE.
 MOV #000,R1 ;SET FD.
 LDFPS R1
 TTB2: NEG# (R0)+ ;TEST INSTRUCTION.
 STFPS R5 ;GET FPS.
 MOV #TTBTP1,R1 ;IS THE RESULT CORRECT.
 MOV #TTBTP2,R2
 MOV #4,R3
 1\$: CMP (R1)+,(R2)+
 BNE TTB10 ;BRANCH IF INCORRECT.
 SOB R3,1\$
 CMP #TTBTP1+4,R0 ;IS R0 CORRECT.
 BNE TTB10 ;BRANCH IF INCORRECT.
 CMP #010,R5 ;IS THE FPS CORRECT?
 BNE TTB10 ;BRANCH IF INCORRECT.
 BR TTBDONE
 ;THESE ARE DATA TABLES AND A DATA BUFFER.
 TTBTP1: 023245
 26720
 122324
 52672
 TTBTP2: 123245
 26720
 122324
 52672
 TTB10:

```

(2) 113372 104000
22105 113374
(1) 113374 004767 011152
(1)
(1)
(1)
(1)
22106
(2)
(3)
(2) 113400
22107 113400 012700 113516
22108 113404 012701 113444
22109 113410 012702 000004
22110 113414 012021
22111 113416 077202
22112 113420 012700 113444
22113 113424 042737 100000 113444
22114 113432 012701 000200
22115 113436 170101
22116 113440 005001
22117
22118 113442 170727
22119 113444 005201 005201 005201
113452 005201
22120
22121 113454 170205
22122 113456 012703 113444
22123 113462 012702 113516
22124 113466 012704 000004
22125 113472 022322
22126 113474 001014
22127 113476 077403
22128 113520 022701 000003
22129 113504 001010
22130 113506 022705 000210
22131 113512 001005
22132 113514 000405
22133
22134
22135 113516 105201
22136 113520 005201
22137 113522 005201
22138 113524 005201
22139
22140 113526
(2) 113526 104000
22141 113530
(1) 113530 004767 011016
(1)
(1)
(1)
(1)
22142
(2)
(3)

```

```

EMT
;
TTBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 552 SPECIAL DEST, MODE2, GR7 (IMMEDIATE), TEST
;*****
T552:
MOV #UUBTP2,R0
MOV #UUBTP1,R1 ;SET UP THE DATA BUFFER.
MOV #4,R2
1$: MOV (R0)+,(R1)+
SOB R2,1$
MOV #UUBTP1,R0
BIC #100000,#UUBTP1 ;MAKE THE OPERAND POSITIVE.
MOV #200,R1 ;SET FD.
LDFPS R1
CLR R1
UUB2: NEG0 (R7)+ ;TEST INSTRUCTION.
UUBTP1: 5201,5201,5201,5201
;NOTE THAT AFTER EXECUTING THIS INSTRUCTION R1 SHOULD CONTAIN 3.
STFPS R5 ;GET FPS.
MOV #UUBTP1,R3 ;IS THE RESULT CORRECT.
MOV #UUBTP2,R2
MOV #4,R4
1$: CMP (R3)+,(R2)+
BNE UUB10 ;BRANCH IF INCORRECT.
SOB R4,1$
CMP #3,R1 ;WAS R1 INCREMENTED CORRECTLY.
BNE UUB10 ;BRANCH IF INCORRECT.
CMP #210,R5 ;IS THE FPS CORRECT?
BNE UUB10 ;BRANCH IF INCORRECT.
BR UUBDONE
;THESE ARE DATA TABLE.
UUBTP2: 105201
5201
5201
5201
UUB10:
EMT
;
UUBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
;*****
;TEST 553 SPECIAL DEST, MODE 6, TEST
;*****

```

(2) 113534
 2214 113534 012701 113650
 22144 113540 012700 113660
 22145 113544 012702 000004
 22146 113550 012021
 22147 113552 077202
 22148 113554 012700 106447
 22149 113560 042737 100000 113650
 22150 113566 012701 000200
 22151 113572 170101
 22152
 22153 113574 005001
 22154 113576 170760 005201
 22155
 22156 113602 170205
 22157 113604 005701
 22158 113606 001030
 22159 113610 012701 113650
 22160 113614 012702 113660
 22161 113620 012703 000004
 22162 113624 022122
 22163 113626 001020
 22164 113630 077303
 22165 113632 022700 106447
 22166 113636 001014
 22167 113640 022705 000210
 22168 113644 001011
 22169 113646 000411
 22170
 22171
 22172 113650 023245
 22173 113652 026720
 22174 113654 122324
 22175 113656 052672
 22176 113660 123245
 22177 113662 026720
 22178 113664 122324
 22179 113666 052672
 22180
 22181
 22182 113670
 (2) 113670 104000
 22183 113672
 (1) 113672 004767 010654
 (1)
 (1)
 (1)
 (1)
 22184
 22185
 (2)
 (3)
 (2) 113676
 22186
 22187 113676 012701 114020
 22188 113702 012700 114030

TS53:
 MOV @XXBTP1,R1 ;SET UP THE DATA BUFFER.
 MOV @XXBTP2,R0
 MOV #4,R2
 1\$: MOV (R0)+,(R1)+
 SOB R2,1\$
 MOV @XXBTP1-5201,R0
 BIC #100000,@XXBTP1;MAKE OPERAND POSITIVE.
 MOV @200,R1 ;SET FD.
 LDFPS R1
 CLR R1
 XXB2: NEG0 5201(R0) ;TEST INSTRUCTION.
 STFPS R5 ;GET FPS.
 TST R1
 BNE XXB10 ;WAS THE PC CORRECT AFTER EXECUTION?
 MOV @XXBTP1,R1 ;IS THE RESULT CORRECT.
 MOV @XXBTP2,R2
 MOV #4,R3
 1\$: CMP (R1)+,(R2)+
 BNE XXB10 ;BRANCH IF INCORRECT.
 SOB R3,1\$
 CMP @XXBTP1-5201,R0 ;IS R0 CORRECT.
 BNE XXB10 ;BRANCH IF INCORRECT.
 CMP #210,R5 ;IS THE FPS CORRECT?
 BNE XXB10 ;BRANCH IF INCORRECT.
 BR XXBDONE
 ;THESE ARE DATA TABLES AND A DATA BUFFER.
 XXBTP1: 023245
 26720
 122324
 52672
 XXBTP2: 123245
 26720
 122324
 52672
 XXB10:
 EMT ;
 XXBDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).
 ;*****
 ;TEST 554 SPECIAL DEST, MODE 7, TEST
 ;*****
 TS554:
 MOV @YYBTP1,R1 ;SET UP THE DATA BUFFER.
 MOV @YYBTP2,R0


```

22189 113706 012702 000004      MOV      #4,R2
22190 113712 012021      14:     MOV      (R0),.(R1).
22191 113714 077202      SOB      R2,14
22192 113716 012700 106637      MOV      @YYBTP3-5201,R0
22193 113722 012760 114020 005201      MOV      @YYBTP1,5201(R0)
22194 113730 042737 100000 114020      BIC      #100000,@YYBTP1 ;MAKE THE OPERAND POSITIVE.
22195 113736 012701 000200      MOV      #200,R1 ;SET FD.
22196 113742 170101      LDFPS   R1
22197
22198 113744 005001      CLR      R1
22199 113746 170770 005201  YYB2-    NEGD     #5201(R0) ;TEST INSTRUCTION.
22200
22201 113752 170205      STFPS   R5 ;GET FPS.
22202 113754 005701      TST     R1 ;WAS THE PC CORRECT AFTER EXECUTION?
22203 113756 001031      BNE     YYB10
22204 113760 012701 114020      MOV      @YYBTP1,R1 ;IS THE RESULT CORRECT.
22205 113764 012702 114030      MOV      @YYBTP2,R2
22206 113770 012703 000004      MOV      #4,R3
22207 113774 022122      14:     CMP      (R1),.(R2).
22208 113776 001021      BNE     YYB10 ;BRANCH IF INCORRECT.
22209 114000 077303      SOB      R3,14
22210 114002 022700 106637      CMP      @YYBTP3-5201,R0 ;IS R0 CORRECT.
22211 114006 001015      BNE     YYB10 ;BRANCH IF INCORRECT.
22212 114010 022705 000210      CMP      #210,R5 ;IS THE FPS CORRECT?
22213 114014 001012      BNE     YYB10 ;BRANCH IF INCORRECT.
22214 114016 000412      BR      YYBDONE
22215
22216 ;THESE ARE DATA TABLES AND A DATA BUFFER.
22217 114020 023245      YYBTP1: 023245
22218 114022 026720      26720
22219 114024 122324      122324
22220 114026 052672      52672
22221 114030 123245      YYBTP2: 123245
22222 114032 026720      26720
22223 114034 123324      123324
22224 114036 052672      52672
22225 114040 114020      YYBTP3: YYBTP1
22226 114042      YYB10:
(2) 114042 104000      EMT ;
22227
22228 114044      YYBDONE:
(1) 114044 004767 010502      JSR      PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
22234 ;*****
(2) ;TEST 555 NEG0, ABS0 AND TSTD TEST
(3) ;*****
(2) 114050      T555:
22235 ;TEST NEG0 WITH POS NONZERO OPERAND
22236 114050 004767 000526      WMB1:   JSR      PC,NATSUB
22237 114054 000000      14:     0 ;FLAG=NEG0.
22238 114056 016341      24:     16341 ;OPERAND.
22239 114060 055772      55772
22240 114062 021133      21133

```

22241	114064	055447			55447	
22242	114066	116341		3:	116341	;RESULT.
22243	114070	055772			55772	
22244	114072	021133			21133	
22245	114074	055447			55447	
22246	114076	016341		4:	16341	;ERROR RES.
22247	114100	055772			55772	
22248	114102	021133			21133	
22249	114104	055447			55447	
22250	114106	000207		5:	207	;FPS BEFORE EXECUTION.
22251	114110	000210			210	;FPS AFTER EXECUTION.
22252	114112	000200			200	;ERROR FPS.
22253	114114	177777			-1	;FEC
22254						;TEST NEG. WITH NEG. OPERAND.
22255	114116	004767	000460	WMB2:	JSR PC,NATSUB	
22256	114122	000000		1:	0	;FLAG=NEG.
22257	114124	152525		2:	152525	;OPERAND.
22258	114126	053545			53545	
22259	114130	055565			55565	
22260	114132	057505			57505	
22261	114134	052525		3:	52525	;RESULT.
22262	114136	053545			53545	
22263	114140	055565			55565	
22264	114142	057505			57505	
22265	114144	152525		4:	152525	;ERROR RES.
22266	114146	053545			53545	
22267	114150	055565			55565	
22268	114152	057505			57505	
22269	114154	000217		5:	217	;FPS BEFORE EXECUTION.
22270	114156	000200			200	;FPS AFTER EXECUTION.
22271	114160	000210			210	;ERROR FPS.
22272	114162	177777			-1	;FEC
22273						;TEST ABSD WITH POSITIVE OPERAND
22274	114164	004767	000412	WMB3:	JSR PC,NATSUB	
22275	114170	000001		1:	1	;FLAG=ABSD.
22276	114172	060705		2:	60705	;OPERAND.
22277	114174	124735			124735	
22278	114176	060124			60124	
22279	114200	073560			73560	
22280	114202	060705		3:	60705	;RESULT.
22281	114204	124735			124735	
22282	114206	060124			60124	
22283	114210	073560			73560	
22284	114212	160705		4:	160705	;ERROR RES.
22285	114214	124735			124735	
22286	114216	060124			60124	
22287	114220	073560			73560	
22288	114222	000217		5:	217	;FPS BEFORE EXECUTION.
22289	114224	000200			200	;FPS AFTER EXECUTION.
22290	114226	000210			210	;ERROR FPS.
22291	114230	177777			-1	;EITHER BUT OP18
22292						;TEST ABSD WITH NEG. OPERAND
22293	114232	004767	000344	WMB4:	JSR PC,NATSUB	
22294	114236	000001		1:	1	;FLAG=ABSD.
22295	114240	154345		2:	154345	;OPERAND.
22296	114242	076567			76567	

22297	114244	032123		32123	
22298	114246	043234		43234	
22299	114250	054345		54345	;RESULT.
22300	114252	076567		76567	
22301	114254	032123		32123	
22302	114256	043234		43234	
22303	114260	154345		154345	;ERROR RES.
22304	114262	076567		76567	
22305	114264	032123		32123	
22306	114266	043234		43234	
22307	114270	000217		217	;FPS BEFORE EXECUTION.
22308	114272	000200		200	;FPS AFTER EXECUTION.
22309	114274	177777		-1	;ERROR FPS.
22310	114276	177777		-1	
22311					;TEST WITH POSITIVE OP
22312	114300	004767	000276	WMB5: JSR PC,NATSUB	
22313	114304	000002		1#: 2	;FLAG=TSTD.
22314	114306	012321		2#: 12321	;OPERAND.
22315	114310	045654			
22316	114312	070107			
22317	114314	034543			
22318	114316	012321		3#: 12321	;RESULT.
22319	114320	045654			
22320	114322	070107			
22321	114324	034543			
22322	114326	112321		4#: 112321	;ERROR RES.
22323	114330	045654			
22324	114332	070107			
22325	114334	034543			
22326	114336	000217		5#: 217	;FPS BEFORE EXECUTION.
22327	114340	000200		200	;FPS AFTER EXECUTION
22328	114342	000210		210	;ERROR FPS.
22329	114344	177777		-1	
22330					;TEST TSTD WITH NEG OP
22331	114346	004767	000230	WMB6: JSR PC,NATSUB	
22332	114352	000002		1#: 2	;FLAG=TSTD.
22333	114354	123765		2#: 123765	;OPERAND.
22334	114356	023407			
22335	114360	034510			
22336	114362	045621			
22337	114364	123765		3#: 123765	;RESULT.
22338	114366	023407			
22339	114370	034510			
22340	114372	045621			
22341	114374	023765		4#: 23765	;ERROR RES.
22342	114376	023407			
22343	114400	034510			
22344	114402	045621			
22345	114404	000207		5#: 207	;FPS BEFORE EXECUTION.
22346	114406	000210		210	;FPS AFTER EXECUTION.
22347	114410	000200		200	;ERROR FPS.
22348	114412	177777		-1	
22349					;TEST TSTD 0 OP
22350	114414	004767	000162	WMB7: JSR PC,NATSUB	
22351	114420	000002		1#: 2	;FLAG=TSTD
22352	114422	000175		2#: 175	;OPERAND.

```

22353 114424 176737 176737
22354 114426 071727 71727
22355 114430 037574 37574
22356 114432 000175 38: 175 ;RESULT.
22357 114434 176737 176737
22358 114436 071727 71727
22359 114440 037574 37574
22360 114442 000000 48: 0 ;ERROR RES.
22361 114444 000000 0
22362 114446 000000 0
22363 114450 000000 0
22364 114452 000200 58: 200 ;FPS BEFORE EXECUTION.
22365 114454 000204 ;FPS AFTER EXECUTION.
22366 114456 000214 ;ERROR FPS.
22367 114460 177777 -1
22368 ;TEST TSTD -0 OP FIUV=0
22369 114462 004767 000114 WMB10: JSR PC,NATSUB
22370 114466 000002 18: 2 ;FLAG=TSTD.
22371 114470 100123 28: 100123 ;OPERAND.
22372 114472 021012 21012
22373 114474 034565 34565
22374 114476 043210 43210
22375 114500 100123 38: 100123 ;RESULT.
22376 114502 021012 21012
22377 114504 034565 34565
22378 114506 043210 43210
22379 114510 000000 48: 0 ;ERROR RES.
22380 114512 000000 0
22381 114514 000000 0
22382 114516 000000 0
22383 114520 040203 58: 40203 ;FPS BEFORE EXECUTION.
22384 114522 040214 ;FPS AFTER EXECUTION.
22385 114524 140214 ;ERROR FPS.
22386 114526 177777 -1
22387 ;TEST TSTD -0 OP FIUV=1
22388 114530 004767 000046 WMB11: JSR PC,NATSUB
22389 114534 000002 18: 2 ;FLAG=TSTD.
22390 114536 100137 28: 100137 ;OPERAND.
22391 114540 024613 24613
22392 114542 057024 57024
22393 114544 060137 60137
22394 114546 100137 38: 100137 ;RESULT.
22395 114550 024613 24613
22396 114552 057024 57024
22397 114554 060137 60137
22398 114556 000000 48: 0 ;ERROR RES.
22399 114560 000000 0
22400 114562 000000 0
22401 114564 000000 0
22402 114566 044200 58: 44200 ;FPS BEFORE EXECUTION.
22403 114570 144214 ;FPS AFTER EXECUTION.
22404 114572 044214 ;ERROR FPS.
22405 114574 000014 14
22406 114576 000167 000162 JMP WMBDONE
22407
22408 ;THIS SUBROUTINE, NATSUB, IS USED TO SET UP THE OPERANDS, EXECUTE

```

```

22409 ;THE EITHER A TSTD, AN ABS0 OR A NEG0 INSTRUCTION AND CHECK THE RESULTS. A CALL
22410 ;TO IT IS MADE THUS:
22411 ;
22412 ;           JSR      PC,#NATSUB
22413 ;           FLAG:   .WORD  X           ;INSTRUCTION TYPE FLAG.
22414 ;           ACARG:  .WORD  X,X,X,X     ;OPERAND
22415 ;           RES:    .WORD  X,X,X,X     ;EXPECTED RESULT
22416 ;           ERRES:  .WORD  X,X,X,X     ;ERROR RESULT
22417 ;           FPSB:   .WORD  X           ;FPS BEFORE EXECUTION
22418 ;           FPSA:   .WORD  X           ;FPS AFTER EXECUTION
22419 ;           FEC:    .WORD  X           ;EXPECTED FEC
22420 ;           ERFPS:  .WORD  X           ;ERROR FPS.
22421 ;           ERR1:   ERROR  X           ;DATA ERROR.
22422 ;           BR      BR      CONT
22423 ;           ERR2:   ERROR  X           ;FPS ERROR.
22424 ;           CONT:   ;RETURN ADDRESS
22425 ;

```

```

22426 ;THE OPERAND IS SET UP IN NATBF1. THEN
22427 ;THE EITHER THE TSTD, NEG0 OR ABS0 INSTRUCTION IS EXECUTED.
22428 ;NATSUB USES THE FIRST OPERAND AS A FLAG TO DETERMINE WHICH INSTRUCTION
22429 ;IS TO BE EXECUTED: 0 = NEG0, 1 = ABS0, 2 = TSTD.
22430 ;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
22431 ;COMPARED WITH FPSA. IF THIS TOO IS CORRECT NATSUB RETURNS CONTROL
22432 ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD NATSUB
22433 ;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN NATSUB WILL RETURN
22434 ;TO THE ERROR CALL AT ERR2, OTHERWISE NATSUB ITSELF
22435 ;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
22436 ;INSTRUCTION IS INCORRECT. THE INCORRECT RESULT IS COMPARED WITH THE
22437 ;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
22438 ;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN NATSUB
22439 ;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
22440 ;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND NATSUB WILL
22441 ;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
22442
22443

```

```

22444 114602 012601
22445 114604 010102
22446 114606 062702 000002
22447 114612 012703 114752
22448 114616 012704 000004
22449 114622 012223
22450 114624 077402
22451 114626 016100 000032
22452 114632 170100
22453 114634 012700 114752
22454 114640 011102
22455 114642 006302
22456 114644 006302
22457 114646 012703 114656
22458 114652 060203
22459 114654 000113
22460 114656 170710
22461 114660 000403
22462 114662 170610
22463 114664 000401
22464 114666 170510

```

```

NATSUB: MOV      (SP)+,R1           ;GET A POINTER TO THE ARGUMENTS.
        MOV      R1,R2           ;COPY THE OPERAND.
        ADD      #2,R2
        MOV      #NATBF1,R3
        MOV      #4,R4
14:     MOV      (R2)+,(R3)+
        SOB      R4,14
        MOV      32(R1),R0       ;LOAD THE FPS.
        LDFPS   R0
        MOV      #NATBF1,R0     ;SET UP THE OPERAND ADDRESS.
        MOV      (R1),R2       ;GET THE FLAG TO DETERMINE WHICH
        ASL      R2           ;INSTRUCTION TO EXECUTE.
        ASL      R2           ;0 = NEG0, 1 = ABS0, 2 = TSTD
        MOV      #NATINS,R3
        ADD      R2,R3
        JMP      (R3)         ;GO EXECUTE THE INSTRUCTION.
NATINS: NEG0   (R0)
        BR      24
        ABS0  (R0)
        BR      24
        TSTD  (R0)

```

```

22465
22466 114670 170204
22467 114672 170305
22468 114674 010100
22469 114676 062700 000012
22470 114702 012702 114752
22471 114706 012703 000004
22472 114712 022022
22473 114714 001014
22474 114716 077303
22475 114720 026104 000034
22476 114724 001010
22477 114726 005761 000034
22478 114732 100003
22479 114734 026105 000040
22480 114740 001002
22481 114742 000161 000042
22482
22483 114746
(2) 114746 104000
22484
22485 114750 177777
22486 114752 177777 177777 177777
114760 177777 177777
22487
22488 114764
(1) 114764 004767 007562
(1)
(1)
(1)
(1)
22489
22490
22497
22498
(2)
(3)
(2) 114770
22499
22500
22501
22502 114770 012700 115040
22503 114774 012710 147517
22504 115000 012737 115014 037576
22505 115006 012737 115044 000004
22506 115014 170110
22507
22508 115016 170205
22509
22510 115020 020027 115040
22511 115024 001007
22512 115026 022705 147517
22513 115032 001004
22514 115034 000404
22515
22516

28: STFPS R4 ;GET THE FPS.
STST R5 ;GET THE FEC.
MOV R1,R0 ;WAS THE RESULT CORRECT?
ADD #12,R0
MOV #NATBF1,R2
MOV #4,R3
38: CMP (R0)+,(R2)+
BNE 108 ;BRANCH IF INCORRECT.
SOB R3,38
CMP 34(R1),R4 ;WAS THE FPS CORRECT?
BNE 108 ;BRANCH IF INCORRECT.
TST 34(R1) ;IF THE EXPECTED FPS WAS NEGATIVE CHECK THE FEC.
BPL 48
CMP 40(R1),R5 ;WAS THE FEC CORRECT.
BNE 108 ;BRANCH IF INCORRECT.
48: JMP 42(R1) ;RETURN.

108: EMT ;

NATBF1: .WORD .1
.WORD -1,-1,1,-1,1

WABDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 556 SOURCE MODES, MODE 1 (FL=0), TEST
;*****
T5556:

MOV #AACTP1,R0 ;SET UP TEST DATA IN BUFFER.
MOV #147517,(R0)
MOV #AAC2,#TMP2
MOV #AAC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
AAC2: LDFPS (R0) ;TEST INSTRUCTION.

STFPS R5 ;GET FPS

CMP R0,#AACTP1 ;IS R0 CORRECT?
BNE AAC10 ;BR IF NOT.
CMP #147517,R5 ;IS FPS CORRECT?
BNE AAC10 ;BR IF NOT.
BR AACDONE

;TEST BUFFER AND DATA:

```

22517 115036 177777
 22518 115040 147517
 22519 115042 177777
 22520 115044
 (2) 115044 104000
 22521
 22522 115046
 (1) 115046 004767 007500
 (1)
 (1)
 (1)
 (1)
 22523
 22524
 22525
 (2)
 (3)
 (2) 115052
 22526
 22527
 22528 115052 012700 115114
 22529 115056 012710 145212
 22530 115062 012737 115120 000004
 22531
 22532 115070 170120
 22533
 22534 115072 170205
 22535
 22536 115074 026027 115116
 22537 115100 001007
 22538 115102 022705 145212
 22539 115106 001004
 22540 115110 000404
 22541
 22542
 22543
 22544 115112 177777
 22545 115114 177777
 22546 115116 177777
 22547

1
 AACTP1: 147517
 1
 AAC10:
 ENT
 ,
 AACDONE:
 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
 ;SEE IF THE USER HAS EXPRESSED
 ;THE DESIRE TO CHANGE THE SOFTWARE
 ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
 ;THE USER TYPED CONTROL G?).

;*****
 ;TEST 557 SOURCE MODES, MODE 2 (FL=0), TEST
 ;*****
 T557:

MOV #BBCTP1,R0 ;SET UP TEST DATA IN BUFFER.
 MOV #145212,(R0)
 MOV #BBC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
 BBC2: LDFPS (R0)+ ;TEST INSTRUCTION.
 STFPS R5 ;GET FPS
 CMP R0,#BBCTP1+2 ;IS R0 CORRECT?
 BNE BBC10 ;BR IF NOT.
 CMP #145212,R5 ;IS THE FPS CORRECT?
 BNE BBC10 ;BR IF NOT.
 BR BBCDONE

;TEST BUFFER AND DATA:
 -1
 BBCTP1: .WORD -1
 -1

22549
22550 115120
(2) 115120 104000
22551 115122
(1) 115122 004767 007424
(1)
(1)
(1)
(1)

BBC10: EMT ;
BBCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

22552
22553
22554
(2)
(3)
(2) 115126

;TEST 560 SOURCE MODES, MODE 4 (FL=0), TEST

TS560:

22555
22556
22557 115126 012700 115210
22558 115132 012760 105252 177776
22559 115140 012737 115154 037576
22560 115146 012737 115220 000004
22561 115154 170140
22562 115156 170205
22563 115160 020027 115206
22564 115164 001015
22565 115166 022705 105252
22566 115172 001012
22567 115174 000412

MOV #DDCTP1+2,R0 ;SET UP THE TEST DATA BUFFER.
MOV #105252,-2(R0)
MOV #DCC2,#TMP2
MOV #DDC10,#ERRVEC
DDC2: LDFPS -(R0)
STFPS R5
CMP R0,#DDCTP1
BNE DDC10
CMP #105252,R5
BNE DDC10
BR DDCDONE

22568
22569 115176 177777 177777 177777
115204 177777
22570 115206 177777
22571 115210 177777 177777 177777
115216 177777

-1,-1,-1,1
DDCTP1: -1
-1,-1,-1,-1

22572 115220
(2) 115220 104000
22573 115222
(1) 115222 004767 007324
(1)
(1)
(1)
(1)

DDC10: EMT ;
DDCDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

22574
(2)
(3)
(2) 115226

;TEST 561 SOURCE MODES, MODE 3 (FL=0), TEST

TS561:

22575 115226 012700 115314
22576 115232 012710 115304
22577 115236 012767 103456 000040
22578 115244 012737 115326 000004
22579 115252 170130
22580 115254 170205
22581 115256 020027 115316
22582 115262 001021
22583 115264 022705 103456
22584 115270 001016

MOV #EECTP2,R0
MOV #EECTP1,(R0)
MOV #103456,EECTP1
MOV #EEC10,#ERRVEC ;SET UP FOR TRAPS TO 4.
EEC2: LDFPS B(R0); ;TEST INSTRUCTION.
STFPS R5 ;GET THE FPS.
CMP R0,#EECTP2+2 ;IS R0 CORRECT?
BNE EEC10 ;BR IF NOT.
CMP #103456,R5 ;IS THE FPS CORRECT?
BNE EEC10 ;BR IF NOT.

22585 115272 000416
 22586
 22587
 22588
 22589 115274 177777 177777 177777
 115302 177777
 22590 115304 177777
 22591 115306 177777 177777 177777
 22592 115317 115304 177777 177777
 115322 177777 000000

 22593
 22594 115326
 (2) 115326 104000
 22595 115330
 (1) 115330 004767 007216
 (1)
 (1)
 (1)
 (1)
 22596
 (2)
 (3)
 (2) 115334
 22597 115334 012700 115420
 22598 115340 012760 115406 177776
 22599 115346 012737 045412 115406
 22600 115354 012737 115426 000004
 22601 115362 170150
 22602 115364 170205
 22603 115366 020027 115416
 22604 115372 001015
 22605 115374 022705 045412
 22606 115400 001012
 22607 115402 000412
 22608
 22609
 22610
 22611 115404 177777
 22612 115406 177777
 22613 115410 177777 177777 177777
 22614 115416 115406 177777 177777
 115424 177777

 22615
 22616 115426
 (2) 115426 104000
 22617 115430
 (1) 115430 004767 007116
 (1)
 (1)
 (1)
 (1)
 22618
 (2)
 (3)
 (2) 115434
 22619 115434 012700 110307

```

BR      EECDONE

;TEST BUFFER AND DATA:
      1. 1.-1. 1

EECTP1: -1
      -1. 1. 1
EECTP2: EECTP1. 1. 1. 1.

EEC10:
EECDONE: EMT
          JSR      PC..RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 562      SOURCE MODES, MODE 5 (FL=0), TEST
;*****
TS562:
      MOV      #FFCTP2+2,R0      ;SET UP THE TEST DATA BUFFER.
      MOV      #FFCTP1,-2(R0)
      MOV      #45412,#FFCTP1
      MOV      #FFC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
FFC2:  LDFPS   @-(R0)           ;TEST INSTRUCTION.
      STFPS   R5              ;GET THE FPS.
      CMP     R0,#FFCTP2      ;IS R0 CORRECT?
      BNE    FFC10           ;BR IF NOT.
      CMP     #45412,R5      ;IS THE FPS CORRECT?
      BNE    FFC10           ;BR IF NOT.
      BR     FFCDONE

;TEST BUFFER AND DATA:
      -1
FFCTP1: -1
      -1,-1,-1
FFCTP2: FFCTP1,-1,-1,-1

FFC10:
FFCDONE: EMT
          JSR      PC..RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 563      SOURCE MODES, MODE 6 (FL=0), TEST
;*****
TS563:
      MOV      #GGCTP1-5201,R0 ;SET UP THE TEST DATA BUFFER.

```

```

22620 115440 012737 046543 115510      MOV      #46543,#GGCTP1
22621 115446 005001                CLR      R1
22622 115450 012737 115522 000004      MOV      #GGC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
22623 115456 170160 005201      GGC2:   LDFPS  5201(R0)                ;TEST INSTRUCTION.
22624 115462 170204                STFPS   R4                ;GET THE FPS.
22625 115464 005701                TST     R1                ;WAS PC CORRECT AFTER EXECUTION?
22626 115466 001015                BNE    GGC10              ;BR IF NOT.
22627 115470 020027 110307      CMP     R0,#GGCTP1-5201 ;IS R0 CORRECT?
22628 115474 001012                BNE    GGC10              ;BR IF NOT.
22629 115476 022704 046543      CMP     #46543,R4        ;IS THE FPS CORRECT?
22630 115502 001007                BNE    GGC10              ;BR IF NOT.
22631 115504 000407                BR     GGCDONE
22632
22633
22634                ;TEST BUFFER AND DATA:
22635 115506 177777                -1
22636 115510 177777 177777 177777  GGCTP1: -1,-1,-1,-1
22637 115520 177777                -1
22638 115522                GGC10:
(2) 115522 104000                EMT
22639 115524                GGCDONE:
(1) 115524 004767 007022      JSR     PC,.RSET        ;GO INITIALIZE THE FPS AND STACK; AND
(1)                ;SEE IF THE USER HAS EXPRESSED
(1)                ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                ;THE USER TYPED CONTROL G?).
22640                ;*****
(2)                ;TEST 564 SOURCE MODES, MODE 7 (FL=0), TEST
(3)                ;*****
(2) 115530      TS564:
22641 115530 012700 110421      MOV     #MMCTP2-5201,R0 ;SET UP THE TEST DATA BUFFER.
22642 115534 012760 115612 005201      MOV     #MMCTP1,5201(R0)
22643 115542 012737 004547 115612      MOV     #4547,#MMCTP1
22644 115550 005001                CLR     R1
22645 115552 012737 115632 000004      MOV     #MHC10,#ERRVECT ;SET UP FOR TRAPS TO 4.
22646 115560 170170 005201      MHC2:  LDFPS  5201(R0)                ;TEST INSTRUCTION.
22647 115564 170204                STFPS  R4                ;GET THE FPS.
22648 115566 005701                TST    R1                ;WAS PC CORRECT AFTER EXECUTION?
22649 115570 001020                BNE   MHC10              ;BR IF NOT.
22650 115572 020027 110421      CMP    R0,#MMCTP2-5201 ;IS R0 CORRECT?
22651 115576 001015                BNE   MHC10              ;BR IF NOT.
22652 115600 022704 004547      CMP    #4547,R4        ;IS THE FPS CORRECT?
22653 115604 001012                BNE   MHC10              ;BR IF NOT.
22654 115606 000412                BR     MHCDONE
22655
22656
22657                ;TEST BUFFER AND DATA:
22658 115610 177777                -1
22659 115612 177777 177777 177777  MHC1P1: .WORD -1,-1,-1,-1
22660 115620 177777
22660 115622 177777 177777 177777  MHC2P2: .WORD -1,-1,-1,-1
22661 115630 177777
22661 115632                MHC10:
(2) 115632 104000                EMT
22662 115634                MHCDONE:
    
```

(1) 115634 004767 006712
(1)
(1)
(1)
(1)

JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

22663
22664
22671
22672

;*****
;TEST 565 SOURCE MODES, MODE 2 GR7 (FL=1), TEST
;*****
TS565:

(2) 115640
22673
22674 115640 012737 115676 000004
22675 115646 012700 000300
22676 115652 170100
22677 115654 005001

MOV #IIC20,BERRVECT ;SET UP FOR TRAPS TO 4.
MOV #300,R0
LDFPS R0
CLR R1

22678
22679 115656 177027
22680 115660 005201
22681 115662 005201
22682 115664 005201
22683 115666 005201

IIC2: LDCLD (R7)+,ACO ;TEST INSTRUCTION.
5201
5201
5201
5201

22684
22685 115670 020127 000003
22686 115674 001401
(1) 115676
(2) 115676 104000

CMP R1,#3 ;WAS PC CORRECT AFTER EXECUTION?
BEQ IICDONE
IIC20: EMT ;

22687
22688 115700
(1) 115700 004767 006646
(1)
(1)
(1)
(1)

IICDONE: JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

22689
22696
22697
(2)

;*****
;TEST 566 SOURCE MODES, MODE 2 (FL=1), TEST
;*****
TS566:

(2) 115704
22698
22699 115704 012700 000300
22700 115710 170100
22701 115712 012700 115756
22702 115716 177020

MOV #300,R0
LDFPS R0
MOV #TCCBF0,R0 ;SET UP THE TEST DATA BUFFER.
TCC2: LDCLD (R0)+,ACO ;TEST INSTRUCTION.

22703
22704 115720 170204
22705 115722 012701 115766
22706 115726 012702 000200
22707 115732 170102
22708 115734 174011
22709 115736 020027 115762
22710 115742 001401
(2) 115744 104000

STFPS R4 ;GET THE FPS.
MOV #TCCBF1,R1 ;GET THE RESULT.
MOV #200,R2
LDFPS R2
STD ACO,(R1)
CMP R0,#TCCBF0+4 ;IS R0 CORRECT?
BEQ TCC3
EMT ;

22711

```
22712 115746 022704 000300 TCC3: CMP #300,R4 ;IS THE FPS CORRECT?
22713 115752 001411 BEQ TCCDONE
(2) 115754 104000 EMT ;
22714
22715
22716 ;TEST BUFFER AND DATA:
22717 115756 001234 067076 054321 TCCBF0: .WORD 01234,67076,54321,012345
115764 012345
22718 115766 177777 177777 177777 TCCBF1: -1,-1,-1,-1
115774 177777
22719
22720 115776 TCCDONE:
(1) 115776 004767 006550 JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
22721
22722
22729
22730 ;*****
(2) ;TEST 567 LDCIF AND LDCLF TEST
(3) ;*****
(2) 116002 TS567:
22731
22732 ;ZERO OPERAND FL=0
22733
22734
22735 116002 004737 116674 KKC1: JSR PC,@LDCFSUB ;GO EXECUTE INSTRUCTION.
22736
22737 116006 000000 000000 1#: .WORD 0,0 ;FSRC OPERAND.
22738 116012 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
22739 116016 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
22740 116022 000000 4#: 0 ;FPS BEFORE EXECUTION.
22741 116024 000004 4 4 ;FPS AFTER EXECUTION.
22742 116026 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
22743
22744 ;ZERO OPERAND FL=0
22745 116030 004737 116674 KKC2: JSR PC,@LDCFSUB ;GO EXECUTE THE INSTRUCTION.
22746
22747 116034 000000 177777 1#: .WORD 0,-1 ;FSRC OPERAND.
22748 116040 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
22749 116044 004177 177400 3#: 4177,177400 ;ANTICIPATED ERRONEOUS RESULT.
22750 116050 000000 4#: 0 ;FPS BEFORE EXECUTION.
22751 116052 000004 4 ;FPS AFTER EXECUTION.
22752 116054 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
22753
22754 ;ZERO OPERAND FL=1
22755 116056 004737 116674 KKC3: JSR PC,@LDCFSUB ;GO EXECUTE THE INSTRUCTION.
22756
22757 116062 000000 000000 1#: .WORD 0,0 ;FSRC OPERAND.
22758 116066 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
22759 116072 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
22760 116076 000100 4#: 100 ;FPS BEFORE EXECUTION.
22761 116100 000104 104 ;FPS AFTER EXECUTION.
22762 116102 000004 4 ;ANTICIPATED ERRONEOUS FPS.
```


Address	Instruction	Operand	FL	FSRC	Expected Result	Anticipated Result
22819						
22820	116335 004737 116674	OPERAND=1	FL=1			
22821	116342 000000 000001	KKC13: JSR PC, BMLDCFSUB				
22822	116346 040200 000000	1#: .WORD 0.1				
22823	116352 034200 000000	2#: .WORD 40200.0				
22824	116356 000100	3#: .WORD 34200.0				
22825	116360 000100	4#: 100				
22826	116362 177777					
22827						
22828	116364 004737 116674	OPERAND=	PATTERN FL=1			
22829	116370 000000 000252	KKC14: JSR PC, BMLDCFSUB				
22830	116374 042052 000000	1#: .WORD 0.252				
22831	116400 036052 000000	2#: .WORD 42052.0				
22832	116404 000111	3#: .WORD 36052.0				
22833	116406 000100	4#: 111				
22834	116410 177777					
22835						
22836	116412 004737 116674	OPERAND=-40000.0	FL=1			
22837	116416 140000 000000	KKC15: JSR PC, BMLDCFSUB				
22838	116422 147600 000000	1#: .WORD -40000.0				
22839	116426 047600 000000	2#: .WORD 147600.0				
22840	116432 000107	3#: .WORD 47600.0				
22841	116434 000110	4#: 107				
22842	116436 177777					
22843						
22844	116440 004737 116674	OPERAND=-1. -1	FL=1			
22845	116444 177777 177777	KKC16: JSR PC, BMLDCFSUB				
22846	116450 140200 000000	1#: .WORD -1. -1				
22847	116454 150000 000000	2#: .WORD 140200.0				
22848	116460 000100	3#: .WORD 150000.0				
22849	116462 000110	4#: 100				
22850	116464 177777					
22851						
22852	116466 004737 116674	OPERAND=-PATTERN	FL=1,			
22853	116472 125252 125252	KKC17: JSR PC, BMLDCFSUB				
22854	116476 147652 125253	1#: .WORD 125252, 125252				
22855	116502 047652 125253	2#: .WORD 147652, 125253				
22856	116506 000105	3#: .WORD 47652, 125253				
22857	116510 000110	4#: 105				
22858	116512 177777					
22859						
22860	116514 004737 116674	OPERAND=77777, 177500	FL=1,			
22861	116520 077777 177500	KKC20: JSR PC, BMLDCFSUB				
22862	116524 047777 177777	1#: .WORD 77777, 177500				
22863	116530 047777 177776	2#: .WORD 47777, 177777				
22864	116534 000117	3#: .WORD 47777, 177776				
22865	116536 000100	4#: 117				
22866	116540 177777					
22867						
22868	116542 004737 116674	OPERAND=40000, 000100	FL=1,			
22869	116546 040000 000100	KKC21: JSR PC, BMLDCFSUB				
22870	116552 047600 000001	1#: .WORD 40000, 100				
22871	116556 047600 000000	2#: .WORD 47600, 1				
22872	116562 000102	3#: .WORD 47600, 0				
22873	116564 000100	4#: 102				
22874	116566 177777					

22875
 22876 116570 004737 116674
 22877 116574 040000 000100
 22878 116600 047600 000000
 22879 116604 047600 000001
 22880 116610 000157
 22881 116612 000140
 22882 116614 177777
 22883
 22884 116616 004737 116674
 22885 116622 100000 000000
 22886 116626 144000 000000
 22887 116632 143600 000000
 22888 116636 000007
 22889 116640 000010
 22890 116642 177777
 22891
 22892 116644 004737 116674
 22893 116650 100000 000000
 22894 116654 150000 000000
 22895 116660 147600 000000
 22896 116664 000107
 22897 116666 000110
 22898 116670 177777
 22899 116672 000441
 22900
 22901
 22902
 22903
 22904
 22905
 22906
 22907
 22908
 22909
 22910
 22911
 22912
 22913
 22914
 22915
 22916
 22917
 22918
 22919
 22920
 22921
 22922
 22923
 22924
 22925
 22926
 22927
 22928
 22929
 22930

```

;OPERAND=40000,000100 FL=1, TRUNC MODE
KKC22: JSR PC,@LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1#: .WORD 40000,100 ;FSRC OPERAND.
2#: .WORD 47600,0 ;EXPECTED RESULT.
3#: .WORD 47600,1 ;ANTICIPATED ERRONEOUS RESULT.
4#: 157 ;FPS BEFORE EXECUTION.
 140 ;FPS AFTER EXECUTION.
 -1 ;ANTICIPATED ERRONEOUS FPS.
;OPERAND=100000,0 (MOST NEG #) FL=0
KKC23: JSR PC,@LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1#: .WORD 100000,0 ;FSRC OPERAND.
2#: .WORD 144000,0 ;EXPECTED RESULT.
3#: .WORD 143600,0 ;ANTICIPATED ERRONEOUS RESULT.
4#: 7 ;FPS BEFORE EXECUTION.
 10 ;FPS AFTER EXECUTION.
 -1 ;ANTICIPATED ERRONEOUS FPS.
;OPERAND=100000,0 FL=1
KKC24: JSR PC,@LDCFSUB ;GO EXECUTE THE INSTRUCTION.
1#: .WORD 100000,0 ;FSRC OPERAND.
2#: .WORD 150000,0 ;EXPECTED RESULT.
3#: .WORD 147600,0 ;ANTICIPATED ERRONEOUS RESULT.
4#: 107 ;FPS BEFORE EXECUTION.
 110 ;FPS AFTER EXECUTION.
 -1 ;ANTICIPATED ERRONEOUS FPS.
6#: BR KKCDONE

```

; THIS SUBROUTINE, LDCFSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
 ; THE LDCIF OR LDCLF INSTRUCTION AND CHECK THE RESULTS. A CALL
 ; TO IT IS MADE THUS:

```

;
;          JSR      PC,@LDCFSUB
;          ACARG:  .WORD  X,X          ;AC OPERAND
;          RES:    .WORD  X,X          ;EXPECTED RESULT
;          ERRES:  .WORD  X,X          ;ERROR RESULT
;          FPSB:   .WORD  X           ;FPS BEFORE EXECUTION
;          FPSA:   .WORD  X           ;FPS AFTER EXECUTION
;          ERFPS:  .WORD  X           ;ERROR FPS
;          ERR1:   ERROR X           ;DATA ERROR
;          BR      CONT
;          ERR2:   ERROR X           ;FPS ERROR
;          CONT:   ;RETURN ADDRESS
;

```

; THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 ; THE LDCIF OR LDCLF INSTRUCTION IS EXECUTED.
 ; THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 ; COMPARED WITH FPSA IF THIS TOO IS CORRECT LDCFSUB RETURNS CONTROL
 ; TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDCFSUB WILL
 ; COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDCFSUB WILL RETURN
 ; TO THE ERROR CALL AT ERR2, OTHERWISE LDCFSUB ITSELF
 ; REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 ; LDCIF OR LDCLF IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 ; ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 ; THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN LDCFSUB
 ; WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 ; RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDCFSUB
 ; REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

22931
22932 116674 012601          LDCFSUB:  MOV      (SP),R1      ;GET A POINTER TO THE ARGUMENTS.
22933 116676 016100 000014  MOV      14(R1),R0      ;SET THE FPS.
22934 116702 170100          LDFPS   RC
22935 116704 010100          MOV      R1,R0
22936
22937 116706 177010          16:     LDCIF   (R0),AC0      ;TEST INSTRUCTION LDCIF OR LDCLF.
22938
22939 116710 170204          STFPS   R4              ;GET FPS.
22940 116712 012700 116766  MOV      @LDCT,R0      ;GET THE RESULT.
22941 116716 012702 000200  MOV      @200,R2
22942 116722 170102          LDFPS   R2
22943 116724 174010          STD     AC0,(R0)
22944
22945 116726 012702 116766  MOV      @LDCT,R2      ;SEE IF THE RESULT WAS CORRECT.
22946 116732 010100          MOV      R1,R0
22947 116734 062700 000004  ADD     @4,RC
22948 116740 012703 000002  MOV     @2,R3
22949 116744 022022          24:     CMP     (R0),R2      ;BR IF INCORRECT.
22950 116746 001006          BNE     10$
22951 116750 077303          SOB     R3,24$
22952
22953 116752 026104 000016  CMP     16(R1),R4      ;SEE IF THE FPS WAS CORRECT.
22954 116756 001002          BNE     10$            ;BR IF INCORRECT.
22955 116760 000161 000022  34:     JMP     22(R1)      ;RETURN.
22956 116764          104:
(2) 116764 104000          EMT
22957
22958          ;DATA BUFFER:
22959 116766 000000 000000 000000 LDCT:   .WORD  0,0,0,0
116774 000000
22960
22961 116776          KKCDONE:
(1) 115776 004767 005550          JSR     PC,,RSET      ;GO INITIALIZE THE FPS AND STACK; AND
(1)          ;SEE IF THE USER HAS EXPRESSED
(1)          ;THE DESIRE TO CHANGE THE SOFTWARE
(1)          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)          ;THE USER TYPED CONTROL G?).
22962
22968
22969          ;*****
(2)          ;TEST 570      LDCID AND LDCLD TEST
(3)          ;*****
(2) 117002          TS570:
22970          ;OPERAND=0      FL=0,  FD=1
22971 117002 004737 117460          LLC1:   JSR     PC,@LDCCDSUB ;GO EXECUTE THE INSTRUCTION.
22972 117006 000000 000000          14:     .WORD  0,0      ;FSRC OPERAND.
22973 117012 000000 000000 000000          24:     .WORD  0,0,0,0 ;EXPECTED RESULT.
117020 000000
22974 117022 177777 177777 177777          34:     .WORD  -1,-1,-1,-1 ;ANTICIPATED ERRONEOUS RESULT.
117030 177777
22975 117032 000213          44:     213      ;FPS BEFORE EXECUTION.
22976 117034 000204          ;FPS AFTER EXECUTION.
22977 117036 177777          -1      ;ANTICIPATED ERRONEOUS FPS.
22978
22979 117040 004737 117460          ;OPERAND=0      FL=0,  FD=1
          LLC2:   JSR     PC,@LDCCDSUB ;GO EXECUTE THE INSTRUCTION.

```


22980	117044	000000	177777		18:	.WORD	0, 1		;FSRC OPERAND.
22981	117050	000000	000000	000000	28:	.WORD	0,0,0,0		;EXPECTED RESULT.
	117056	000000							
22982	117060	004177	177400	000000	38:	.WORD	4177,177400,0,0		;ANTICIPATED ERRONEOUS RESULT.
	117066	000000							
22983	117070	000200			48:	200			;FPS BEFORE EXECUTION.
22984	117072	000204				204			;FPS AFTER EXECUTION.
22985	117074	177777				-1			;ANTICIPATED ERRONEOUS FPS.
22986						;OPERAND=0	FL=1 FD=1		
22987	117076	004737	117460		LLC3:	JSR	PC,B#LDCCDSUB		;GO EXECUTE THE INSTRUCTION.
22988	117102	000000	000000		18:	.WORD	0,0		;FSRC OPERAND.
22989	117106	000000	000000	000000	28:	.WORD	0,0,0,0		;EXPECTED RESULT.
	117114	000000							
22990	117116	177777	177777	177777	38:	.WORD	-1,-1,-1,-1		;ANTICIPATED ERRONEOUS RESULT.
	117124	177777							
22991	117126	000211			48:	211			;FPS BEFORE EXECUTION.
22992	117130	000204				204			;FPS AFTER EXECUTION.
22993	117132	177777				-1			;ANTICIPATED ERRONEOUS FPS.
22994						;OPERAND=40000	FL=0 FD=1		
22995	117134	004737	117460		LLC4:	JSR	PC,B#LDCCDSUB		;GO EXECUTE THE INSTRUCTION.
22996	117140	040000	000000		18:	.WORD	40000,0		;FSRC OPERAND.
22997	117144	043600	000000	000000	28:	.WORD	43600,0,0,0		;EXPECTED RESULT.
	117152	000000							
22998	117154	047600	000000	000000	38:	.WORD	47600,0,0,0		;ANTICIPATED ERRONEOUS RESULT.
	117162	000000							
22999	117164	000217			48:	217			;FPS BEFORE EXECUTION.
23000	117166	000200				200			;FPS AFTER EXECUTION.
23001	117170	177777				-1			;ANTICIPATED ERRONEOUS FPS.
23002						;OPERAND=-40000	FL=0 FD=1		
23003	117172	004737	117460		LLC5:	JSR	PC,B#LDCCDSUB		;GO EXECUTE THE INSTRUCTION.
23004	117176	140000	000000		18:	.WORD	-40000,0		;FSRC OPERAND.
23005	117202	143600	000000	000000	28:	.WORD	143600,0,0,0		;EXPECTED RESULT.
	117210	000000							
23006	117212	043600	000000	000000	38:	.WORD	43600,0,0,0		;ANTICIPATED ERRONEOUS RESULT.
	117220	000000							
23007	117222	000200			48:	200			;FPS BEFORE EXECUTION.
23008	117224	000210				210			;FPS AFTER EXECUTION.
23009	117226	177777				-1			;ANTICIPATED ERRONEOUS FPS.
23010						;OPERAND=40000,0	FL=1 FD=1		
23011	117230	004737	117460		LLC6:	JSR	PC,B#LDCCDSUB		;GO EXECUTE THE INSTRUCTION.
23012	117234	040000	000000		18:	.WORD	40000,0		;FSRC OPERAND.
23013	117240	047600	000000	000000	28:	.WORD	47600,0,0,0		;EXPECTED RESULT.
	117246	000000							
23014	117250	043600	000000	000000	38:	.WORD	43600,0,0,0		;ANTICIPATED ERRONEOUS RESULT.
	117256	000000							
23015	117260	000317			48:	317			;FPS BEFORE EXECUTION.
23016	117262	000300				300			;FPS AFTER EXECUTION.
23017	117264	177777				-1			;ANTICIPATED ERRONEOUS FPS.
23018						;OPERAND=0,1	FL=1 FD=1		
23019	117266	004737	117460		LLC7:	JSR	PC,B#LDCCDSUB		;GO EXECUTE THE INSTRUCTION.
23020	117272	000000	000001		18:	.WORD	0,1		;FSRC OPERAND.
23021	117276	040200	000000	000000	28:	.WORD	40200,0,0,0		;EXPECTED RESULT.
	117304	000000							
23022	117306	034200	000000	000000	38:	.WORD	34200,0,0,0		;ANTICIPATED ERRONEOUS RESULT.
	117314	000000							
23023	117316	000300			48:	300			;FPS BEFORE EXECUTION.

```

23024 117320 000300          300          ;FPS AFTER EXECUTION.
23025 117322 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
23026          ;OPERAND=77777,177777 FL=1 FD=1
23027 117324 004737 117460 LLC10: JSR PC,BMLDCDSUB ;GO EXECUTE THE INSTRUCTION.
23028 117330 077777 177777 14: .WORD 77777,177777 ;FSRC OPERAND.
23029 117334 047777 177777 177000 24: .WORD 47777,177777,177000.0 ;EXPECTED RESULT.
      117342 000000
23030 117344 177777 177777 177777 34: .WORD -1,-1, 1, 1 ;ANTICIPATED ERRONEOUS RESULT.
      117352 177777
23031 117354 000317          44: 317 ;FPS BEFORE EXECUTION.
23032 117356 000300          300 ;FPS AFTER EXECUTION.
23033 117360 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
23034          ;OPERAND=-PATTERN FL=1 FD=1
23035
23036 117362 004767 000072 LLC11: JSR PC,LDCDSUB ;GO EXECUTE THE INSTRUCTION.
23037 117366 177777 177526 14: .WORD -1,-252 ;FSRC OPERAND.
23038 117372 142052 000000 000000 24: .WORD 142052.0,0.0 ;EXPECTED RESULT.
      117400 000000
23039 117402 136052 000000 000000 34: .WORD 136052.0,0.0 ;ANTICIPATED ERRONEOUS RESULT.
      117410 000000
23040 117412 000307          44: 307 ;FPS BEFORE EXECUTION.
23041 117414 000310          310 ;FPS AFTER EXECUTION.
23042 117416 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
23043          ;OPERAND=PATTERN FL=1 FD=1 FT=1
23044 117420 004767 000034 LLC12: JSR PC,LDCDSUB ;GO EXECUTE THE INSTRUCTION.
23045 117424 012345 067012 14: .WORD 12345,67012 ;FSRC OPERAND.
23046 117430 047247 025560 050000 24: .WORD 47247,025560,050000,0 ;EXPECTED RESULT.
      117436 000000
23047 117440 177777 177777 177777 34: .WORD -1,-1,-1, 1 ;ANTICIPATED ERRONEOUS RESULT.
      117446 177777
23048 117450 000352          44: 352 ;FPS BEFORE EXECUTION.
23049 117452 000340          340 ;FPS AFTER EXECUTION.
23050 117454 177777          -1          ;ANTICIPATED ERRONEOUS FPS.
23051 117456 000435          64: BR LLCDONE
23052
23053          ;THIS SUBROUTINE, LDCDSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
23054          ;THE LDCID OR LDCLD INSTRUCTION AND CHECK THE RESULTS. A CALL
23055          ;TO IT IS MADE THUS:
23056          ;
23057          ; JSR PC,BMLDCDSUB
23058          ; ACARG: .WORD X,X ;AC OPERAND
23059          ; RES: .WORD X,X,X,X ;EXPECTED RESULT
23060          ; ERRES: .WORD X,X,X,X ;ERROR RESULT
23061          ; FPSB: .WORD X ;FPS BEFORE EXECUTION
23062          ; FPSA: .WORD X ;FPS AFTER EXECUTION
23063          ; ERFPS: .WORD X ;ERROR FPS.
23064          ; ERR1: ERROR X ;DATA ERROR.
23065          ; BR CONT
23066          ; ERR2: ERROR X ;FPS ERROR.
23067          ; CONT: ;RETURN ADDRESS
23068
23069          ;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
23070          ;THE LDCID OR LDCLD INSTRUCTION IS EXECUTED.
23071          ;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
23072          ;COMPARED WITH FPSA IF THIS TOO IS CORRECT LDCDSUB RETURNS CONTROL
23073          ;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDCDSUB

```

23074
23075
23076
23077
23078
23079
23080
23081
23082
23083
23084 117460 012601
23085 117462 016100 000024
23086 117466 170100
23087 117470 010100
23088 117472 177010
23089
23090 117474 170204
23091 117476 012700 116766
23092 117502 012702 000200
23093 117506 170102
23094 117510 174010
23095
23096
23097 117512 012702 116766
23098 117516 010100
23099 117520 062700 000004
23100 117524 012703 000002
23101 117530 022022
23102 117532 001006
23103 117534 077303
23104
23105 117536 026104 000026
23106 117542 001002
23107 117544 000161 000032
23108 117550
(2) 117550 104000
23109
23110 117552
(1) 117552 004767 004774
(1)
(1)
(1)
(1)
(1)
23111
23120
23121
(2)
(3)
(2) 117556
23122
23123
23124 117536 004767 001136
23125 117562 012345 067012 034567
117570 012345
23126 117572 000010
23127 117574 042145 067012 034567

```

;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDCDSUB WILL RETURN
;TO THE ERROR CALL AT ERR2. OTHERWISE LDCDSUB ITSELF
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
;LDCID OR LDCLD IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRS. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRS THEN LDCDSUB
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDCDSUB WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

LDCDSUB:      MOV      (SP),R1      ;GET A POINTER TO THE ARGUMENTS.
              MOV      24(R1),R0  ;SET THE FPS.
              LDFPS   R0
              MOV      R1,R0
1$:           LDCID   (R0),AC0     ;TEST INSTRUCTION, LDCID OR LDCLD.

              STFPS   R4          ;GET FPS.
              MOV      @LDCT,R0   ;GET THE RESULT.
              MOV      @200,R2
              LDFPS   R2
              STD     AC0,(R0)

;SEE IF THE RESULT IS CORRECT.
              MOV      @LDCT,R2
              MOV      R1,R0
              ADD     @4,R0
              MOV      @2,R3
2$:           CMP     (R0),R3      ;BR IF INCORRECT.
              BNE    10$
              SOB    R3,2$

              CMP     26(R1),R4   ;IS THE FPS CORRECT?
              BNE    10$         ;BR IF INCORRECT.
3$:           JMP     32(R1)      ;RETURN.
10$:          EMT

LLCDONE:     JSR     PC,.RSET     ;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

;*****
;TEST 571      LDEXP TEST
;*****
TS571:
; NON-ZERO RES. VALID EXPON=210 (EXCESS 200)=10
MPC1:       JSR     PC,LDXSUB    ;GO EXECUTE THE INSTRUCTION.
1$:         .WORD  12345,67012,34567,012345 ;ACO OPERAND.

2$:         .WORD  10 ;EXPONENT OPERAND.
3$:         .WORD  42145,67012,34567,012345 ;EXPECTED RESULT.

```

23128	117602	012345	067012	034567	4:	.WORD	2145.67012,34567,012345	;ANTICIPATED ERRONEOUS RESULT.
	117604	002145						
	117612	012345						
23129	117614	047217			5:	47217	;FPS BEFORE EXECUTION.	
23130	117616	047200				47200	;FPS AFTER EXECUTION.	
23131	117620	147200				147200	;ANTICIPATED ERRONEOUS FPS.	
23132	117622	177777				-1	;EXPECTED FEC.	
23133						;NON-ZERO	RES NEG.	
23134	117624	004737	120720		MHC2:	JSR	PC,04L0XSUB ;EXPON=377	
23135	117630	123456	070123	045670	1:	.WORD	123456,70123,45670,123456	;ACO OPERAND.
	117636	123456						
23136	117640	000177			2:	.WORD	177	;EXPONENT OPERAND.
23137	117642	177656	070123	045670	3:	.WORD	177656,70123,45670,123456	;EXPECTED RESULT.
	117650	123456						
23138	117652	137656	070123	045670	4:	.WORD	137656,70123,45670,123456	;ANTICIPATED ERRONEOUS RESULT.
	117660	123456						
23139	117662	047207			5:	47207	;FPS BEFORE EXECUTION.	
23140	117664	047210				47210	;FPS AFTER EXECUTION.	
23141	117666	147210				147210	;ANTICIPATED ERRONEOUS FPS.	
23142	117670	177777				-1	;EXPECTED FEC.	
23143						;NON-ZERO	RES, EXP=256=(56)REAL	
23144	117672	004737	120720		MHC3:	JSR	PC,04L0XSUB ;GO EXECUTE THE INSTRUCTION.	
23145	117676	073261	057645	043323	1:	.WORD	73261,057645,43323,101760	;ACO OPERAND.
	117704	101760						
23146	117706	000056			2:	.WORD	56	;EXPONENT OPERAND.
23147	117710	053461	057645	043323	3:	.WORD	53461,057645,43323,101760	;EXPECTED RESULT.
	117716	101760						
23148	117720	177777	177777	177777	4:	.WORD	-1,-1,-1,-1	;ANTICIPATED ERRONEOUS RESULT.
	117726	177777						
23149	117730	047200			5:	47200	;FPS BEFORE EXECUTION.	
23150	117732	047200				47200	;FPS AFTER EXECUTION.	
23151	117734	147200				147200	;ANTICIPATED ERRONEOUS FPS.	
23152	117736	177777				-1	;EXPECTED FEC.	
23153						;EXP=27 (EXCESS 200)=-151 (OCT)		
23154	117740	004737	120720		MHC4:	JSR	PC,04L0XSUB ;GO EXECUTE THE INSTRUCTION.	
23155	117744	012223	024252	062720	1:	.WORD	12223,24252,62720,21222	;ACO OPERAND.
	117752	021222						
23156	117754	177627			2:	.WORD	-151	;EXPONENT OPERAND.
23157	117756	005623	024252	062720	3:	.WORD	5623,24252,62720,21222	;EXPECTED RESULT.
	117764	021222						
23158	117766	177777	177777	177777	4:	.WORD	-1,-1,-1,-1	;ANTICIPATED ERRONEOUS RESULT.
	117774	177777						
23159	117776	047200			5:	47200	;FPS BEFORE EXECUTION.	
23160	120000	047200				47200	;FPS AFTER EXECUTION.	
23161	120002	147200				147200	;ANTICIPATED ERRONEOUS FPS.	
23162	120004	177777				-1	;EXPECTED FEC.	
23163						;EXP=0 (EXCESS 200)=-200 (OCT), POSITIVE FRAC		
23164						;FIV=1		
23165	120006	004737	120720		MHC5:	JSR	PC,04L0XSUB ;GO EXECUTE THE INSTRUCTION.	
23166	120012	030131	032334	035363	1:	.WORD	30131,32334,35363,73031	;ACO OPERAND.
	120020	073031						
23167	120022	177600			2:	.WORD	-200	;EXPONENT OPERAND.
23168	120024	000131	032334	035363	3:	.WORD	00131,32334,35363,73031	;EXPECTED RESULT.
	120032	073031						
23169	120034	000000	000000	000000	4:	.WORD	0,0,0,0	;ANTICIPATED ERRONEOUS RESULT.
	120042	000000						

23170	120044	042200			5:	42200		;FPS BEFORE EXECUTION.
23171	120046	142204				142204		;FPS AFTER EXECUTION.
23172	120050	042202				42202		;ANTICIPATED ERRONEOUS FPS.
23173	120052	000012				12		;EXPECTED FEC.
23174						;EXP=0	(EXCESS 200)=-200 (OCT), NEG FRACT,FIU=1	
23175	120054	004737	120720		MHC6:	JSR	PC,BMLDXSUB	;GO EXECUTE THE INSTRUCTION.
23176	120060	140414	024344	045464	1:	.WORD	140414,24344,45464,74045	;ACO OPERAND.
	120066	074045						
23177	120070	177600			2:	.WORD	-200	;EXPONENT OPERAND.
23178	120072	100014	024344	045464	3:	.WORD	100014,24344,45464,74045	;EXPECTED RESULT.
	120100	074045						
23179	120102	000000	000000	000000	4:	.WORD	0,0,0,0	;ANTICIPATED ERRONEOUS RESULT.
	120110	000000						
23180	120112	042200			5:	42200		;FPS BEFORE EXECUTION.
23181	120114	142214				142214		;FPS AFTER EXECUTION.
23182	120116	042214				42214		;ANTICIPATED ERRONEOUS FPS.
23183	120120	000012				12		;EXPECTED FEC.
23184						;EXP=0	(EXCESS 200)=-200 (OCT), POS FRAC, FIU=0	
23185								
23186	120122	004737	120720		MHC7:	JSR	PC,BMLDXSUB	;GO EXECUTE THE INSTRUCTION.
23187	120126	051525	035455	005675	1:	.WORD	51525,35455,5675,05152	;ACO OPERAND.
	120134	005152						
23188	120136	177600			2:	.WORD	-200	;EXPONENT OPERAND.
23189	120140	000000	000000	000000	3:	.WORD	0,0,0,0	;EXPECTED RESULT.
	120146	000000						
23190	120150	000125	035455	005675	4:	.WORD	00125,35455,5675,05152	;ANTICIPATED ERRONEOUS RESULT.
	120156	005152						
23191	120160	045200				45200		;FPS BEFORE EXECUTION.
23192	120162	045204				45204		;FPS AFTER EXECUTION.
23193	120164	145204				145204		;ANTICIPATED ERRONEOUS FPS.
23194	120166	177777				-1		;EXPECTED FEC.
23195						;EXP=-1405	(EXCESS 200)=-1605 (OCT), FIU=1	
23196	120170	004737	120720		MHC10:	JSR	PC,BMLDXSUB	;GO EXECUTE THE INSTRUCTION.
23197	120174	061526	062636	046566	1:	.WORD	61626,62636,46566,67606	;ACO OPERAND.
	120202	067606						
23198	120204	176173			2:	.WORD	-1605	;EXPONENT OPERAND.
23199	120206	076626	062636	046566	3:	.WORD	76626,62636,46566,67606	;EXPECTED RESULT.
	120214	067606						
23200	120216	000000	000000	000000	4:	.WORD	0,0,0,0	;ANTICIPATED ERRONEOUS RESULT.
	120224	000000						
23201	120226	042200			5:	42200		;FPS BEFORE EXECUTION.
23202	120230	142200				142200		;FPS AFTER EXECUTION.
23203	120232	042204				42204		;ANTICIPATED ERRONEOUS FPS.
23204	120234	000012				12		;EXPECTED FEC.
23205						;EXP=-17416	(EXCESS 200)=-17616 (OCT), FIU=0	
23206	120236	004737	120720		MHC11:	JSR	PC,BMLDXSUB	;GO EXECUTE THE INSTRUCTION.
23207	120242	071727	037475	076777	1:	.WORD	71727,37475,76777,17273	;ACO OPERAND.
	120250	017273						
23208	120252	160162			2:	.WORD	-17616	;EXPONENT OPERAND.
23209	120254	000000	000000	000000	3:	.WORD	0,0,0,0	;EXPECTED RESULT.
	120262	000000						
23210	120264	074527	037475	076777	4:	.WORD	74527,37475,76777,17273	;ANTICIPATED ERRONEOUS RESULT.
	120272	017273						
23211	120274	045200			5:	45200		;FPS BEFORE EXECUTION.
23212	120276	045204				45204		;FPS AFTER EXECUTION.
23213	120300	145200				145200		;ANTICIPATED ERRONEOUS FPS.

```
23214 120302 177777 1 ;EXPECTED FEC.
23215 ;EXP= 1601 (EXCESS 200)= 2001 (OCT), FIV=1
23216 120304 004737 120720 MNC12: JSR PC, @LDXSUB ;GO EXECUTE THE INSTRUCTION.
23217 120310 001020 030405 006070 1#: .WORD 01020,30405,06070,00102 ;ACO OPERAND.
120316 000102
23218 120320 175777 2#: .WORD -2001 ;EXPONENT OPERAND.
23219 120322 037620 030405 006070 3#: .WORD 37620,30405,06070,00102 ;EXPECTED RESULT
120330 000102
23220 120332 000000 000000 000000 4#: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
120340 000000
23221 120342 042200 5#: 42200 ;FPS BEFORE EXECUTION.
23222 120344 142200 142200 ;FPS AFTER EXECUTION.
23223 120346 042204 42204 ;ANTICIPATED ERRONEOUS FPS.
23224 120350 000012 12 ;EXPECTED FEC.
23225 ;EXP=1206 (EXCESS 200)=1006 (OCT) FIV =1
23226 120352 004737 120720 MNC13: JSR PC, @LDXSUB ;GO EXECUTE THE INSTRUCTION
23227 120356 012131 014151 016171 1#: .WORD 12131,14151,16171,10111 ;ACO OPERAND.
120364 010111
23228 120366 001006 2#: .WORD 1006 ;EXPONENT OPERAND.
23229 120370 041531 014151 016171 3#: .WORD 41531,14151,16171,10111 ;EXPECTED RESULT.
120376 010111
23230 120400 000000 000000 000000 4#: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
120406 000000
23231 120410 041200 5#: 41200 ;FPS BEFORE EXECUTION.
23232 120412 141202 141202 ;FPS AFTER EXECUTION.
23233 120414 041204 41204 ;ANTICIPATED ERRONEOUS FPS.
23234 120416 000010 10 ;EXPECTED F. . .
23235 ;EXP=16315 (EXCESS 200)=16115 (OCT) FIV=0
23236 120420 004737 120720 MNC14: JSR PC, @LDXSUB ;GO EXECUTE THE INSTRUCTION.
23237 120424 027262 025242 023222 1#: .WORD 27262,25242,23222,21202 ;ACO OPERAND.
120432 021202
23238 120434 016115 2#: .WORD 16115 ;EXPONENT OPERAND.
23239 120436 000000 000000 000000 3#: .WORD 0,0,0,0 ;EXPECTED RESULT.
120444 000000
23240 120446 063262 025242 023222 4#: .WORD 63262,25242,23222,21202 ;ANTICIPATED ERRONEOUS RESULT.
120454 021202
23241 120456 046200 5#: 46200 ;FPS BEFORE EXECUTION.
23242 120460 046206 46206 ;FPS AFTER EXECUTION.
23243 120462 146202 146202 ;ANTICIPATED ERRONEOUS FPS.
23244 120464 177777 -1 ;EXPECTED FEC.
23245 ;EXP=11011 (EXCESS 200)=10611 (OCT) FIV=1
23246
23247 120466 004737 120720 MNC15: JSR PC, @LDXSUB ;GO EXECUTE THE INSTRUCTION.
23248 120472 030313 032333 034353 1#: .WORD 30313,32333,34353,36373 ;ACO OPERAND.
120500 036373
23249 120502 010611 2#: .WORD 10611 ;EXPONENT OPERAND.
23250 120504 002313 032333 034353 3#: .WORD 2313,32333,34353,36373 ;EXPECTED RESULT.
120512 036373
23251 120514 000000 000000 000000 4#: .WORD 0,0,0,0 ;ANTICIPATED ERRONEOUS RESULT.
120522 000000
23252 120524 041200 5#: 41200 ;FPS BEFORE EXECUTION.
23253 120526 141202 141202 ;FPS AFTER EXECUTION.
23254 120530 041204 41204 ;ANTICIPATED ERRONEOUS FPS.
23255 120532 000010 10 ;EXPECTED FEC.
23256 ;EXP=17123 (EXCESS 200)=16723 (OCT) FIV=0
23257
```

```

23258 120534 004737 120720      MMC16: JSR      PC,B@LDXSUB      ;GO EXECUTE THE INSTRUCTION.
23259 120540 040414 042434 044454 1#:      .WORD    40414,42434,44454,46474 ;ACO OPERAND.
      120546 046474
23260 120550 016723                2#:      .WORD    16723          ;EXPONENT OPERAND.
23261 120552 000000 000000 000000 3#:      .WORD    0,0,0,0          ;EXPECTED RESULT.
      120560 000000
23262 120562 024614 042434 044454 4#:      .WORD    24614,42434,44454,46474 ;ANTICIPATED ERRONEOUS RESULT.
      120570 046474
23263 120572 046200                5#:      46200          ;FPS BEFORE EXECUTION.
23264 120574 046206                46206          ;FPS AFTER EXECUTION.
23265 120576 146202                146202         ;ANTICIPATED ERRONEOUS FPS.
23266 120600 177777                1          ;EXPECTED FEC.
23267                ;EXP= 254 (OCT)= 454 (EXCESS 200) FIV=1
23268
23269 120602 004737 120720      MMC17: JSR      PC,B@LDXSUB      ;GO EXECUTE THE INSTRUCTION.
23270 120606 050515 052535 054555 1#:      .WORD    50515,52535,54555,56575 ;ACO OPERAND.
      120614 056575
23271 120616 000254                2#:      .WORD    254          ;EXPONENT OPERAND.
23272 120620 013115 052535 054555 3#:      .WORD    13115,52535,54555,56575 ;EXPECTED RESULT.
      120626 056575
23273 120630 000000 000000 000000 4#:      .WORD    0,0,0,0          ;ANTICIPATED ERRONEOUS RESULT.
      120636 000000
23274 120640 041200                5#:      41200          ;FPS BEFORE EXECUTION.
23275 120642 141202                141202         ;FPS AFTER EXECUTION.
23276 120644 041204                41204          ;ANTICIPATED ERRONEOUS FPS.
23277 120646 000010                10          ;EXPECTED FEC.
23278                ;EXP= 313 (OCT)= 513(EXCESS 200) FIV=0
23279
23280 120650 004737 120720      MMC20: JSR      PC,B@LDXSUB      ;GO EXECUTE THE INSTRUCTION.
23281 120654 060616 062636 064656 1#:      .WORD    60616,62636,64656,66676 ;ACO OPERAND.
      120662 066676
23282 120664 000313                2#:      .WORD    313          ;EXPONENT OPERAND.
23283 120666 000000 000000 000000 3#:      .WORD    0,0,0,0          ;EXPECTED RESULT.
      120674 000000
23284 120676 022616 062636 064656 4#:      .WORD    22616,62636,64656,66676 ;ANTICIPATED ERRONEOUS RESULT.
      120704 066676
23285 120706 046200                5#:      46200          ;FPS BEFORE EXECUTION.
23286 120710 046206                46206          ;FPS AFTER EXECUTION.
23287 120712 146202                146202         ;ANTICIPATED ERRONEOUS FPS.
23288 120714 177777                -1          ;EXPECTED FEC.
23289 120716 000457                BR      MMCDONE
  
```

```

;THIS SUBROUTINE, LDXSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE LDEXP INSTRUCTION AND CHECK THE RESULTS. A CALL
;TO IT IS MADE THUS:
  
```

```

23291 ;
23292 ;
23293 ;
23294 ;
23295 ;
23296 ;
23297 ;
23298 ;
23299 ;
23300 ;
23301 ;
23302 ;
23303 ;
23304 ;
      JSR      PC,B@LDXSUB
      ACARG:  .WORD    X,X,X,X          ;AC OPERAND
      EXP:    .WORD    X          ;EXPONENT
      RES:    .WORD    X,X,X,X        ;EXPECTED RESULT
      ERRES:  .WORD    X,X,X,X        ;ERROR RESULT
      FPSB:   .WORD    X          ;FPS BEFORE EXECUTION
      FPSA:   .WORD    X          ;FPS AFTER EXECUTION
      ERFPS:  .WORD    X          ;ERROR FPS.
      FEC:    .WORD    X          ;EXPECTED FEC
      ERR1:   ERROR  X          ;DATA ERROR.
  
```

```

23305      :          BR      CONT
23306      :          ERR2.  ERROR  X          ,FPS ERROR
23307      :          CONT:          ;RETURN ADDRESS
23308      :
23309      ; THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
23310      ; THE LDEXP INSTRUCTION IS EXECUTED
23311      ; THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
23312      ; COMPARED WITH FPSA IF THIS TOO IS CORRECT LDXSUB RETURNS CONTROL
23313      ; TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD LDXSUB
23314      ; COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN LDXSUB WILL RETURN
23315      ; TO THE ERROR CALL AT ERR2, OTHERWISE LDXSUB ITSELF
23316      ; REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
23317      ; LDEXP IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
23318      ; ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
23319      ; THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN LDXSUB
23320      ; WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
23321      ; RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND LDXSUB WILL
23322      ; REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.
23323
23324 120720 012601      LDXSUB: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
23325 120722 012700 000200      MOV      #200,R0      ;LOAD THE ACO OPERAND.
23326 120726 170100      LDFPS   R0
23327 120730 010100      MOV      R1,R0
23328 120732 172410      LDD     (R0),ACO
23329 120734 016100 000032      MOV      32(R1),R0      ;SET UP THE FPS.
23330 120740 170100      LDFPS   R0
23331 120742 010100      MOV      R1,R0
23332 120744 062700 000010      ADD     #10,R0
23333
23334 120750 176410      1$:  LDEXP  (R0),ACO      ;TEST INSTRUCTION.
23335
23336 120752 170204      STFPS   R4          ;GET THE FPS.
23337 120754 170305      STST   R5          ;GET THE FEC
23338 120756 012700 000200      MOV      #200,R0      ;GET THE RESULT.
23339 120762 170100      LDFPS   R0
23340 120764 012700 121046      MOV      #LDXT,R0
23341 120770 174010      STD     ACO,(R0)
23342 120772 012702 121045      MOV      #LDXT,R2      ;SEE IF THE RESULT WAS CORRECT.
23343 120776 010103      MOV      R1,R3
23344 121000 062703 000012      ADD     #12,R3
23345 121004 012700 000004      MOV      #4,R0
23346 121010 022223      2$:  CMP     (R2)+,(R3)+
23347 121012 001014      BNE     10$          ;BRANCH IF NOT CORRECT.
23348 121014 077003      SOB     R0,2$
23349 121016 020461 000034      CMP     R4,34(R1)      ;SEE IF THE FPS WAS CORRECT.
23350 121022 001010      BNE     10$          ;BRANCH IF NOT CORRECT.
23351 121024 005761 000034      TST     34(R1)
23352 121030 100003      BPL     3$
23353 121032 020561 000040      CMP     R5,40(R1)      ;SEE IF THE FEC WAS CORRECT.
23354 121036 001002      BNE     10$          ;BRANCH IF NOT CORRECT.
23355
23356 121040 000161 000042      3$:  JMP     42(R1)      ;RETURN.
23357 121044      10$:
(2) 121044 104000      EMT
23358
23359      ;DATA BUFFER:

```


23360 121046 000000 000000 000000
121054 000000

23361
23362 121056
(1) 121056 004767 003470

(1)
(1)
(1)
(1)

23363
23364
23371
23372

(2)
(3)
(2) 121062

23373
23374

23375 121062 012700 121152
23376 121066 012701 000006
23377 121072 012720 177777

23378 121076 077103
23379 121100 012700 102345

23380 121104 012737 121166 000004
23381 121112 170100
23382 121114 012700 121156

23383
23384 121120 170210
23385 121122 020027 121156

23386 121126 001017
23387 121130 023727 121156 102345
23388 121136 001013

23389 121140 023727 121160 177777
23390 121146 001007
23391 121150 000407

23392
23393
23394 121152 177777 177777
23395 121156 177777 177777 177777
121164 177777

23396 121166
(2) 121166 104000

23397
23398 121170
(1) 121170 004767 003356

(1)
(1)
(1)
(1)

23399
23400
23401

(2)
(3)
(2) 121174

23402

```
MMCDONE:
      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                          ;SEE IF THE USER HAS EXPRESSED
                          ;THE DESIRE TO CHANGE THE SOFTWARE
                          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                          ;THE USER TYPED CONTROL G?).
```

```
;*****
;TEST 572      DESTINATION MODES, MODE 1 (FL=0), TEST
;*****
TS572:
```

```
      MOV      @NNCTB0,R0      ;SET UP THE DATA BUFFER.
      MOV      @6,R1
1$:   MOV      @-1,(R0)+
      SOB      R1,1$
      MOV      @102345,R0
      MOV      @NNC10,@ERRVECT ;SET UP FOR TRAPS TO 4.
      LDFPS   R0              ;SET UP FPS.
      MOV      @NNCTB1,R0
NNC2: STFPS   (R0)           ;TEST INSTRUCTION.
      CMP      R0,@NNCTB1     ;IS R0 CORRECT?
      BNE      NNC10          ;BRANCH IF NOT CORRECT.
      CMP      @NNCTB1,@102345 ;IS RESULT CORRECT?
      BNE      NNC10          ;BRANCH IF NOT CORRECT.
      CMP      @NNCTB1+2,@-1  ;IS THE RESULT CORRECT?
      BNE      NNC10          ;BRANCH IF NOT CORRECT.
      BR      NNCDONE
```

```
;TEST DATA BUFFER:
NNCTB0: .WORD  -1,-1
NNCTB1: .WORD  -1,-1,-1,-1
```

```
NNC10:
      EMT
```

```
NNCDONE:
      JSR      PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND
                          ;SEE IF THE USER HAS EXPRESSED
                          ;THE DESIRE TO CHANGE THE SOFTWARE
                          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
                          ;THE USER TYPED CONTROL G?).
```

```
;*****
;TEST 573      DESTINATION MODES, MODE 2 (FL=0), TEST
;*****
TS573:
```

```
23403
23404 121174 012700 121264      MOV      #OOC10,RO      ;SET UP THE DATA BUFFER.
23405 121200 012701 000006      MOV      #6,R1
23406 121204 012720 177777      1$:     MOV      #1,(RO)+
23407 121210 077103              SOB      R1,1$
23408 121212 012700 105412      MOV      #105412,RO
23409 121216 012737 121300 000004  MOV      #OOC10,#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
23410 121224 170100              LDFPS   RO              ;SET UP FPS.
23411 121226 012700 121270      MOV      #OOC10,RO
23412
23413 121232 170220      OOC2:   STFPS   (RO)+    ;TEST INSTRUCTION.
23414 121234 020027 121272      CMP      RO,#OOC10+2    ;IS RO CORRECT?
23415 121240 001017              BNE     OOC10           ;BRANCH IF NOT CORRECT.
23416 121242 023727 121270 105412  CMP      #OOC10+2,#105412 ;IS THE RESULT CORRECT?
23417 121250 001013              BNE     OOC10           ;BRANCH IF NOT CORRECT.
23418 121252 023727 121272 177777  CMP      #OOC10+2,#-1   ;IS THE RESULT CORRECT?
23419 121260 001007              BNE     OOC10           ;BRANCH IF NOT CORRECT.
23420 121262 000407
23421
23422              ;TEST DATA BUFFER:
23423 121264 177777 177777      OOC10:  .WORD   -1,-1
23424 121270 177777 177777 177777  OOC11:  .WORD   -1,-1,-1,-1
23425 121300
(2) 121300 104000
23426
23427 121302
(1) 121302 004767 003244      OOCDONE: JSR      PC,,RSET ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
23428
23429
23430
23431
(2)
(3)
(2) 121306
23432
23433 121306 012700 121376      MOV      #PPC10,RO     ;SET UP THE DATA BUFFER.
23434 121312 012701 000006      MOV      #6,R1
23435 121316 012720 177777      1$:     MOV      #-1,(RO)+
23436 121322 077103              SOB      R1,1$
23437 121324 012700 105555      MOV      #105555,RO
23438 121330 012737 121412 000004  MOV      #PPC10,#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
23439 121336 170100              LDFPS   RO              ;SET UP FPS.
23440 121340 012700 121404      MOV      #PPC10+2,RO
23441
23442 121344 170240      PPC2:   STFPS   -(RO)    ;TEST INSTRUCTION.
23443 121346 020027 121402      CMP      RO,#PPC10     ;IS RO CORRECT?
23444 121352 001017              BNE     PPC10          ;BRANCH IF NOT CORRECT.
23445 121354 023727 121402 105555  CMP      #PPC10+2,#105555 ;IS THE RESULT CORRECT?
23446 121362 001013              BNE     PPC10          ;BRANCH IF NOT CORRECT.
23447 121364 023727 121404 177777  CMP      #PPC10+2,#-1   ;IS THE RESULT CORRECT?
23448 121372 001007              BNE     PPC10          ;BRANCH IF NOT CORRECT.
```

23449 121374 000407
23450
23451
23452 121376 177777 177777
23453 121402 177777 177777 177777
121410 177777
23454 121412
(2) 121412 104000
23455 121414
(1) 121414 004767 003132
(1)
(1)
(1)
(1)
(1)

BR PPCDONE
;TEST DATA BUFFER:
PPCTB0: .WORD 1, 1
PPCTB1: .WORD -1, 1, 1, 1
PPC10:
EMT
PPCDONE:
JSR PC, .RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

23456
23457
23458
23459
(2)
(3)
(2) 121420

;*****
;TEST 575 DESTINATION MODES, MODE 3 (FL=0), TEST
;*****
TS575:

23460
23461 121420 012700 121514
23462 121424 012701 000010
23463 121430 012720 177777
23464 121434 077103
23465 121436 012700 106653
23466 121442 012737 121534 000004
23467 121450 170100
23468 121452 012700 121530
23469 121456 012710 121520
23470
23471 121462 170230
23472 121464 020027 121532
23473 121470 001021
23474 121472 023727 121520 106653
23475 121500 001015
23476 121502 023727 121530 121520
23477 121510 001011
23478 121512 000411
23479

MOV @QCTB0, R0 ;SET UP THE DATA BUFFER.
MOV @10, R1
18: MOV @-1, (R0).
SOB R1, 18
MOV @106653, R0
MOV @QCT10, @ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
LDFPS R0 ;SET UP FPS.
MOV @QCTB2, R0
MOV @QCTB1, (R0)
QOC2: STFPS @R0). ;TEST INSTRUCTION.
CMP R0, @QCTB2+2 ;IS R0 CORRECT?
BNE QOC10 ;BRANCH IF NOT CORRECT.
CMP @QCTB1, @106653 ;IS THE RESULT CORRECT?
BNE QOC10 ;BRANCH IF NOT CORRECT.
CMP @QCTB2, @QCTB1 ;IS THE RESULT CORRECT?
BNE QOC10 ;BRANCH IF NOT CORRECT.
BR QOCDONE

23480
23481 121514 177777 177777
23482 121520 177777 177777 177777
121526 177777
23483 121530 177777 177777
23484 121534
(2) 121534 104000
23485 121536
(1) 121536 004767 003010
(1)
(1)
(1)
(1)

;TEST DATA BUFFER:
QOCTB0: .WORD -1, -1
QOCTB1: .WORD -1, -1, -1, -1
QOCTB2: .WORD -1, -1
QOC10:
EMT
QOCDONE:
JSR PC, .RSET

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

23486
23487

```

23488
23489
(2)
(3)
(2) 121542
23490
23491
23492 121542 012700 121640
23493 121546 012701 000006
23494 121552 012720 177777
23495 121556 077103
23496 121560 012700 004301
23497 121564 012737 121660 000004
23498 121572 170100
23499 121574 012700 121656
23500 121600 012760 121644 177776
23501
23502 121606 170250
23503 121610 020027 121654
23504 121614 001021
23505 121616 023727 121644 004301
23506 121624 001015
23507 121626 023727 121654 121644
23508 121634 001011
23509 121636 000411
23510
23511
23512 121640 177777 177777
23513 121644 177777 177777 177777
121652 177777
23514 121654 177777 177777
23515 121660
(2) 121660 104000
23516 121662
(1) 121662 004767 002664
(1)
(1)
(1)
(1)
(1)
23517
23518
23519
(2)
(3)
(2) 121666
23520
23521
23522 121666 012700 121770
23523 121672 012701 000006
23524 121676 012720 177777
23525 121702 077103
23526 121704 012700 102514
23527 121710 012737 122004 000004
23528 121716 170100
23529 121720 005001
23530 121722 012700 114573

```

```

;*****
;TEST 576 DESTINATION MODES, MODE 5 (FL=0), TEST
;*****
TS576:

```

```

MOV #RRC10,R0 ;SET UP THE DATA BUFFER.
MOV #6,R1
MOV #-1,(R0).
1$: SOB R1,1$
MOV #004301,R0
MOV #RRC10,#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
LDFPS R0 ;SET UP FPS.
MOV #RRC10+2,R0
MOV #RRC10,-2(R0)
RRC2: STFPS #-1(R0) ;TEST INSTRUCTION.
CMP R0,#RRC10 ;IS R0 CORRECT?
BNE RRC10 ;BRANCH IF NOT CORRECT.
CMP #RRC10,#004301 ;IS THE RESULT CORRECT?
BNE RRC10 ;BRANCH IF NOT CORRECT.
CMP #RRC10+2,#RRC10 ;IS THE RESULT CORRECT?
BNE RRC10 ;BRANCH IF NOT CORRECT.
BR RRCDONE

```

```

;TEST DATA BUFFER:
RRC10: .WORD -1, 1
RRC11: .WORD -1,-1,-1, 1
RRC12: .WORD -1,-1
RRC13: EMT
RRCDONE: JSR PC,.RSET

```

```

;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

```

```

;*****
;TEST 577 DESTINATION MODES, MODE 6 (FL=0), TEST
;*****
TS577:

```

```

MOV #SSCT10,R0 ;SET UP THE DATA BUFFER.
MOV #6,R1
MOV #-1,(R0).
1$: SOB R1,1$
MOV #102514,R0
MOV #SSCT10,#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.
LDFPS R0 ;SET UP FPS.
CLR R1
MOV #SSCT10+201,R0

```

E I

```
23531
23532 121726 170260 005201          SSC2:  STFPS  5201(R0)          ;TEST INSTRUCTION.
23533 121732 020127 000000          CMP      R1,#0          ;WAS PC CORRECT AFTER EXECUTION?
23534 121736 001022                   BNE      SSC10          ;BRANCH IF NOT CORRECT.
23535 121740 020027 114573          CMP      R0,#SSCTB1 5201 ;IS R0 CORRECT?
23536 121744 001017                   BNE      SSC10          ;BRANCH IF NOT CORRECT.
23537 121746 023727 121774 102514  CMP      @#SSCTB1,@102514 ;IS THE RESULT CORRECT?
23538 121754 001013                   BNE      SSC10          ;BRANCH IF NOT CORRECT.
23539 121756 023727 121776 177777  CMP      @#SSCTB1+2,@-1  ;IS THE RESULT CORRECT?
23540 121764 001007                   BNE      SSC10          ;BRANCH IF NOT CORRECT.
23541 121766 000407                   BR       SSCDONE
23542
23543                                     ;TEST DATA BUFFER:
23544 121770 177777 177777          SSCTB0: .WORD  -1, 1
23545 121774 177777 177777 177777  SSCTB1: .WORD  -1,-1, 1,-1
23546 122002 177777
23546 122004                                     SSC10:
(2) 122004 104000                                     EMT
23547 122006                                     SSCDONE:
(1) 122006 004767 002540                                     JSR      PC,.RSET
(1)
(1)
(1)
(1)
```

```
;GO INITIALIZE THE FPS AND STACK; AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).
```

23549
23550
23551
 (2)
 (3)
 (2) 122012
23552
23553 122012 012700 122122
23554 122016 012701 000010
23555 122022 012720 177777
23556 122026 077103
23557 122030 012700 103747
23558 122034 012737 122142 000004
23559 122042 170100
23560 122044 005001
23561 122046 012700 114735
23562 122052 012760 122126 005201
23563
23564 122060 170270 005201
23565 122064 022701 000000
23566 122070 001024
23567 122072 020027 114735
23568 122076 001021
23569 122100 023727 122126 103747
23570 122106 001015
23571 122110 023727 122130 177777
23572 122116 001011
23573 122120 000411
23574
23575
23576 122122 177777 177777
23577 122126 177777 177777 177777
 122134 177777
23578 122136 177777 177777
23579 122142
 (2) 122142 104000
23580 122144
 (1) 122144 004767 002402
 (1)
 (1)
 (1)
 (1)
 (1)
23581
23588
 (2)
 (3)
 (2) 122150
23589 122150 012700 006300
23590 122154 170100
23591 122156 012700 122202
23592 122162 172410
23593 122164 012700 122214
23594
23595 122170 175420
23596
23597 122172 020027 122220

```
*****  
;TEST 600 DESTINATION MODES, MODE 7 (FL=0), TEST  
*****  
TS600:  
      MOV     #TTCTB0,R0      ;SET UP THE DATA BUFFER.  
      MOV     #10,R1  
1$:   MOV     #-1,(R0)+  
      SOB     R1,1$  
      MOV     #103747,R0  
      MOV     #TTC10,#ERRVECT ;SET UP FOR TRAPS TO VECTOR 4.  
      LDFPS   RO              ;SET UP FPS.  
      CLR     R1  
      MOV     #TTCTB2-5201,R0  
      MOV     #TTCTB1,5201(R0)  
  
TTC2: STFPS   #5201(R0)      ;TEST INSTRUCTION.  
      CMP     #0,R1          ;WAS PC CORRECT AFTER EXECUTION?  
      BNE     TTC10          ;BRANCH IF NOT CORRECT.  
      CMP     RO,#TTCTB2-5201 ;IS RO CORRECT?  
      BNE     TTC10          ;BRANCH IF NOT CORRECT.  
      CMP     #TTCTB1,#103747 ;IS THE RESULT CORRECT?  
      BNE     TTC10          ;BRANCH IF NOT CORRECT.  
      CMP     #TTCTB1+2,#-1   ;IS THE RESULT CORRECT?  
      BNE     TTC10          ;BRANCH IF NOT CORRECT.  
      BR      TTCDONE  
  
;TEST DATA BUFFER:  
TTCTB0: .WORD  -1,-1  
TTCTB1: .WORD  -1,-1,-1,-1  
  
TTCTB2: .WORD  -1,-1  
TTC10:  
TTCDONE: EMT  
      JSR     PC,.RSET      ;GO INITIALIZE THE FPS AND STACK; AND  
                          ;SEE IF THE USER HAS EXPRESSED  
                          ;THE DESIRE TO CHANGE THE SOFTWARE  
                          ;VIRTUAL CONSOLE SWITCH REGISTER (HAS  
                          ;THE USER TYPED CONTROL G?).
```

```
*****  
;TEST 601 DESTINATION MODES, MODE 2 (FL=1), TEST  
*****  
TS601:  
      MOV     #300,R0        ;SET UP FPS.  
      LDFPS   RO  
      MOV     #UUCTP1,R0     ;SET UP THE ACO OPERAND.  
      LDD     (R0),ACO  
      MOV     #UUCBFO,R0  
  
UUC2: STCDL   ACO,(R0)+     ;TEST INSTRUCTION.  
      CMP     RO,#UUCBFO+4   ;IS RO CORRECT?
```

```
23598 122176 001411          BEQ      UUCDONE
      (2) 122200 104000          EMT
23599          ;TEST DATA BUFFER:
23600 122202 000000 000000 000000 UUCTP1: .WORD 0,0,0,0
      122210 000000
23601 122212 177777          1
23602 122214 177777 177777 177777 UUCBFO: .WORD -1,-1, 1
23603
23604 122222          UUCDONE:
      (1) 122222 004767 002324      JSR      PC,.RSET          ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                                     ;SEE IF THE USER HAS EXPRESSED
      (1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                                     ;THE USER TYPED CONTROL G?).
23605
23612          ;*****
      (2)          ;TEST 602          DESTINATION MODES, MODE 4 (FL-1), TEST
      (3)          ;*****
      (2) 122226          TS602:
23613
23614 122226 012700 000300          MOV      #300,RO          ;SET UP FPS.
23615 122232 170100          LDFPS   RO
23616 122234 012700 122260          MOV      #VVCTP1,RO      ;SET UP THE ACO OPERAND.
23617 122240 172410          LDD     (RO),ACO
23618 122242 012700 122276          MOV      #VVCBFO+4,RO
23619
23620 122246 175440          VVC2:  STCDL   ACO,-(RO)          ;TEST INSTRUCTION.
23621
23622 122250 020027 122272          CMP     RO,#VVCBFO      ;IS RO CORRECT?
23623 122254 001411          BEQ     VVCDONE
      (2) 122256 104000          EMT
23624          ;TEST DATA BUFFER:
23625 122260 000000 000000 000000 VVCTP1: .WORD 0,0,0,0
      122266 000000
23626 122270 177777          -1
23627 122272 177777 177777 177777 VVCBFO: .WORD -1,-1,-1
23628
23629 122300          VVCDONE:
      (1) 122300 004767 002246      JSR     PC,.RSET          ;GO INITIALIZE THE FPS AND STACK; AND
      (1)                                     ;SEE IF THE USER HAS EXPRESSED
      (1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
      (1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
      (1)                                     ;THE USER TYPED CONTROL G?).
23630
23640          ;*****
      (2)          ;TEST 603          STCDI AND STCDL TEST
      (3)          ;*****
      (2) 122304          TS603:
23641
23642          ;FIRST TEST STC WITH EXP=100 (EXCESS 200)
23643 122304 004737 123242          WWC1:  JSR     PC,#STCSUB      ;GO EXECUTE THE INSTRUCTION.
23644 122310 020000 000000 000000 1$: .WORD 20000,0,0,0      ;ACO OPERAND.
      122316 000000
23645 122320 000000 000000          2$: .WORD 0,0          ;EXPECTED RESULT.
23646 122324 177777 177777          3$: .WORD -1,-1          ;ERROR RES.
23647 122330 040300          4$: 40300          ;FPS BEFORE EXECUTION.
```

```

23648 122332 040304 40304 ;FPS AFTER EXECUTION.
23649 122334 140304 140304 ;ANTICIPATED ERRONEOUS FPS.
23650 122336 177777 -1 ;REPORT RESULT INCORRECT.
23651 ;EXP=0 (OCT) FL=1 FIC=0
23652 122340 004737 123242 WMC2: JSR PC,@STCS JB ;GO EXECUTE THE INSTRUCTION.
23653 122344 040000 000000 000000 1#: .WORD 40000.0,0,0 ;ACO OPERAND.
122352 000000
23654 122354 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
23655 122360 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
23656 122364 040313 4#: 40313 ;FPS BEFORE EXECUTION.
23657 122366 040304 40304 ;FPS AFTER EXECUTION.
23658 122370 140304 140304 ;ANTICIPATED ERRONEOUS FPS.
23659 122372 177777 -1 ;EXPECTED FEC.
23660 ;EXP=37 (OCT) FL=1 FIC=1
23661 122374 004737 123242 WMC3: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23662 122400 047667 075757 157737 1#: .WORD 47667,75757,157737,167773 ;ACO OPERAND.
122406 167773
23663 122410 055675 173757 2#: .WORD 55675,173757 ;EXPECTED RESULT.
23664 122414 122102 004021 3#: .WORD 122102,004021 ;ANTICIPATED ERRONEOUS RESULT.
23665 122420 040717 4#: 40717 ;FPS BEFORE EXECUTION.
23666 122422 040700 40700 ;FPS AFTER EXECUTION.
23667 122424 140705 140705 ;ANTICIPATED ERRONEOUS FPS.
23668 122426 177777 -1 ;EXPECTED FEC.
23669 ;EXP=40 (OCT) FL=1 FIC=1
23670 122430 004737 123242 WMC4: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23671 122434 050000 000000 000000 1#: .WORD 50000.0,0,0 ;ACO OPERAND.
122442 000000
23672 122444 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
23673 122450 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
23674 122454 040700 4#: 40700 ;FPS BEFORE EXECUTION.
23675 122456 140705 140705 ;FPS AFTER EXECUTION.
23676 122460 040705 40705 ;ANTICIPATED ERRONEOUS FPS.
23677 122462 000006 6 ;EXPECTED FEC.
23678
23679 ;EXP=40 (OCT) FL=1 FIC=0
23680 122464 004737 123242 WMC5: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23681 122470 050000 000000 000000 1#: .WORD 50000.0,0,0 ;ACO OPERAND.
122476 000000
23682 122500 000000 000000 2#: .WORD 0,0 ;EXPECTED RESULT.
23683 122504 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
23684 122510 040312 4#: 40312 ;FPS BEFORE EXECUTION.
23685 122512 040305 40305 ;FPS AFTER EXECUTION.
23686 122514 140305 140305 ;ANTICIPATED ERRONEOUS FPS.
23687 122516 177777 -1 ;EXPECTED FEC.
23688 ;EXP=30 (OCT) FL=1 FIC=1
23689 122520 004737 123242 WMC6: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23690 122524 046000 000001 000000 1#: .WORD 46000.1,0,0 ;ACO OPERAND.
122532 000000
23691 122534 000200 000001 2#: .WORD 200,1 ;EXPECTED RESULT.
23692 122540 177777 177777 3#: .WORD -1,-1 ;ANTICIPATED ERRONEOUS RESULT.
23693 122544 040700 4#: 40700 ;FPS BEFORE EXECUTION.
23694 122546 040700 40700 ;FPS AFTER EXECUTION.
23695 122550 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
23696 122552 177777 -1 ;EXPECTED FEC.
23697 ;EXP=27 (OCT) FL=1 FIC=1
23698 122554 004737 123242 WMC7: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
```


23699	122560	045600	000001	000000	1#:	.WORD	45600,1,0,0	;ACO OPERAND.
	122566	000000						
23700	122570	000100	000000		2#:	.WORD	100,0	;EXPECTED RESULT.
23701	122574	177777	177777		3#:	.WORD	1,1	;ANTICIPATED ERRONEOUS RESULT.
23702	122600	040707			4#:	40707		;FPS BEFORE EXECUTION.
23703	122602	040700				40700		;FPS AFTER EXECUTION.
23704	122604	177777				-1		;ANTICIPATED ERRONEOUS FPS.
23705	122606	177777				-1		;EXPECTED FEC.
23706								
23707	122610	004737	123242					
23708	122614	043600	000000	000000	1#:	.WORD	43600,0,0,0	;GO EXECUTE THE INSTRUCTION ;ACO OPERAND.
	122622	000000						
23709	122624	040000	177777		2#:	.WORD	40000,-1	;EXPECTED RESULT.
23710	122630	000000	177777		3#:	.WORD	0,-1	;ANTICIPATED ERRONEOUS RESULT.
23711	122634	040600			4#:	40600		;FPS BEFORE EXECUTION.
23712	122636	040600				40600		;FPS AFTER EXECUTION.
23713	122640	140604				140604		;ANTICIPATED ERRONEOUS FPS.
23714	122642	177777				-1		;EXPECTED FEC.
23715								
23716								
23717	122644	004737	123242					
23718	122650	044000	000000	000000	1#:	.WORD	44000,0,0,0	;GO EXECUTE THE INSTRUCTION. ;ACO OPERAND.
	122656	000000						
23719	122660	000000	177777		2#:	.WORD	0,-1	;EXPECTED RESULT.
23720	122664	177777	177777		3#:	.WORD	-1,-1	;ANTICIPATED ERRONEOUS RESULT.
23721	122670	040600			4#:	40600		;FPS BEFORE EXECUTION.
23722	122672	140605				140605		;FPS AFTER EXECUTION.
23723	122674	040600				40600		;ANTICIPATED ERRONEOUS FPS.
23724	122676	000006				6		;EXPECTED FEC.
23725								
23726	122700	004737	123242					
23727	122704	142000	000000	000000	1#:	.WORD	142000,0,0,0	;GO EXECUTE THE INSTRUCTION. ;ACO OPERAND.
	122712	000000						
23728	122714	177600	177777		2#:	.WORD	177600,-1	;EXPECTED RESULT.
23729	122720	000200	000000		3#:	.WORD	200,0	;ANTICIPATED ERRONEOUS RESULT.
23730	122724	040600			4#:	40600		;FPS BEFORE EXECUTION.
23731	122726	040610				40610		;FPS AFTER EXECUTION.
23732	122730	040600				40600		;ANTICIPATED ERRONEOUS FPS.
23733	122732	177777				-1		;EXPECTED FEC.
23734								
23735	122734	004737	123242					
23736	122740	147600	000000	000000	1#:	.WORD	147600,0,0,0	;GO EXECUTE THE INSTRUCTION. ;ACO OPERAND.
	122746	000000						
23737	122750	140000	000000		2#:	.WORD	140000,0	;EXPECTED RESULT.
23738	122754	137777	000000		3#:	.WORD	137777,0	;ANTICIPATED ERRONEOUS RESULT.
23739	122760	040700			4#:	40700		;FPS BEFORE EXECUTION.
23740	122762	040710				40710		;FPS AFTER EXECUTION.
23741	122764	177777				-1		;ANTICIPATED ERRONEOUS FPS.
23742	122766	177777				-1		;EXPECTED FEC.
23743								
23744	122770	004737	123242					
23745	122774	147600	000000	001000	1#:	.WORD	147600,0,1000,0	;GO EXECUTE THE INSTRUCTION. ;ACO OPERAND.
	123002	000000						
23746	123004	137777	177777		2#:	.WORD	137777,177777	;EXPECTED RESULT.
23747	123010	140000	177777		3#:	.WORD	140000,177777	;ANTICIPATED ERRONEOUS RESULT.
23748	123014	040707			4#:	40707		;FPS BEFORE EXECUTION.

J1

```
23749 123016 040710 40710 ;FPS AFTER EXECUTION.
23750 123020 177777 1 ;ANTICIPATED ERRONEOUS FPS.
23751 123022 177777 -1 ;EXPECTED FEC.
23752 ;EXP=41 (OCT), AC NEG, FL=1, FIC=1
23753 123024 004737 123242 WWC15: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23754 123030 150200 000000 000000 1$: .WORD 150200,0,0,0 ;ACO OPERAND.
123036 000000
23755 123040 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
23756 123044 177777 177777 3$: .WORD -1,1 ;ANTICIPATED ERRONEOUS RESULT.
23757 123050 040700 4$: 40700 ;FPS BEFORE EXECUTION.
23758 123052 140705 140705 ;FPS AFTER EXECUTION.
23759 123054 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
23760 123056 000006 6 ;EXPECTED FEC.
23761 ;EXP=40 (OCT), AC NEG, FL=1, FIC=1
23762 123060 004737 123242 WWC16: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23763 123064 150000 000001 000000 1$: .WORD 150000,1,0,0 ;ACO OPERAND.
123072 000000
23764 123074 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
23765 123100 100000 177600 3$: .WORD 100000,-200 ;ANTICIPATED ERRONEOUS RESULT.
23766 123104 040700 4$: 40700 ;FPS BEFORE EXECUTION.
23767 123106 140705 140705 ;FPS AFTER EXECUTION.
23768 123110 040700 40700 ;ANTICIPATED ERRONEOUS FPS.
23769 123112 000006 6 ;EXPECTED FEC.
23770 ;EXP=40, AC NEGATIVE, FL=1, FIC=1
23771 123114 004737 123242 WWC17: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23772 123120 150001 000000 000000 1$: .WORD 150001,0,0,0 ;ACO OPERAND.
123126 000000
23773 123130 000000 000000 2$: .WORD 0,0 ;EXPECTED RESULT.
23774 123134 077400 000000 3$: .WORD 77400,0 ;ANTICIPATED ERRONEOUS RESULT.
23775 123140 040700 4$: 40700 ;FPS BEFORE EXECUTION.
23776 123142 140705 140705 ;FPS AFTER EXECUTION.
23777 123144 177777 -1 ;ANTICIPATED ERRONEOUS FPS.
23778 123146 000006 6 ;EXPECTED FEC.
23779 ;EXP 40 (OCT), AC MOST NEG LONG INT, FL=1
23780 ;FIC=1
23781 123150 004737 123242 WWC20: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23782 123154 150000 000000 000000 1$: .WORD 150000,0,0,0 ;ACO OPERAND.
123162 000000
23783 123164 100000 000000 2$: .WORD 100000,0 ;EXPECTED RESULT.
23784 123170 000000 000000 3$: .WORD 0,0 ;ANTICIPATED ERRONEOUS RESULT.
23785 123174 040700 4$: 40700 ;FPS BEFORE EXECUTION.
23786 123176 040710 40710 ;FPS AFTER EXECUTION.
23787 123200 140705 140705 ;ANTICIPATED ERRONEOUS FPS.
23788 123202 177777 -1 ;EXPECTED FEC.
23789 ;EXP=20, AC = MOST NEG INTEGER, FL=0, FIC=1
23790
23791 123204 004737 123242 WWC21: JSR PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23792 123210 144000 000001 000000 1$: .WORD 144000,1,0,0 ;ACO OPERAND.
123216 000000
23793 123220 100000 177777 2$: .WORD 100000,-1 ;EXPECTED RESULT.
23794 123224 100000 177400 3$: .WORD 100000,177400 ;ANTICIPATED ERRONEOUS RESULT.
23795 123230 040600 4$: 40600 ;FPS BEFORE EXECUTION.
23796 123232 040610 40610 ;FPS AFTER EXECUTION.
23797 123234 140605 140605 ;ANTICIPATED ERRONEOUS FPS.
23798 123236 177777 -1 ;EXPECTED FEC.
23799 123240 000457 6$: BR WWC DONE
```

K1

23800
 23801
 23802
 23803
 23804
 23805
 23806
 23807
 23808
 23809
 23810
 23811
 23812
 23813
 23814
 23815
 23816
 23817
 23818
 23819
 23820
 23821
 23822
 23823
 23824
 23825
 23826
 23827
 23828
 23829
 23830
 23831
 23832
 23833
 23834
 23835
 23836
 23837
 23838
 23839
 23840
 23841
 23842
 23843
 23844
 23845
 23846
 23847
 23848
 23849
 23850
 23851
 23852
 23853
 23854
 23855

; THIS SUBROUTINE, STCSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
 ; THE STCDI OR STCDL INSTRUCTION AND CHECK THE RESULTS A CALL
 ; TO IT IS MADE THUS:

```

:
:      JSR      PC,#STCSUB
:      ACARG:  .WORD  X,X,X,X      ;AC OPERAND
:      RES:    .WORD  X,X          ;EXPECTED RESULT
:      ERRES:  .WORD  X,X          ;ERROR RESULT
:      FPSB:   .WORD  X            ;FPS BEFORE EXECUTION
:      FPSA:   .WORD  X            ;FPS AFTER EXECUTION
:      ERFPS:  .WORD  X            ;ERROR FPS.
:      FEC:    .WORD  X            ;EXPECTED FEC
:      ERR1:   ERROR X            ;DATA ERROR.
:      BR      BR      CONT
:      ERR2:   ERROR X            ;FPS ERROR.
:      CONT:   CONT              ;RETURN ADDRESS
  
```

; THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
 ; THE STCDI OR STCDL INSTRUCTION IS EXECUTED.
 ; THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
 ; COMPARED WITH FPSA IF THIS TOO IS CORRECT STCSUB RETURNS CONTROL
 ; TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STCSUB
 ; COMPARE IT TO ERROR FPS, IF THIS MATCHES THEN STCSUB WILL RETURN
 ; TO THE ERROR CALL AT ERR2, OTHERWISE STCSUB ITSELF
 ; REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
 ; STCDI OR STCDL IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
 ; ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
 ; THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STCSUB
 ; WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
 ; RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STCSUB WILL
 ; REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

STCSUB: MOV      (SP)+,R1      ;GET A POINTER TO THE ARGUMENTS.
        MOV      #200,R0     ;SET UP THE ACO OPERAND.
        LDFPS   R0
        MOV      R1,R0
        LDD     (R0),ACO
        MOV      #STCIBF,R2  ;INITIALIZE THE OUT PUT BUFFER.
        MOV      #4,R0
1$:     MOV      #-1,(R2)+
        SOB     R0,1$
        MOV      20(R1),R0   ;SET THE FPS.
        LDFPS   R0
        MOV      #STCIBF,R0
2$:     STCDL   ACO,(R0)     ;TEST INSTRUCTION.

        STFPS   R4          ;GET THE FPS.
        STST   R5          ;GET THE FEC.
        MOV      R1,R2
        ADD     #10,R2
        MOV      #STCIBF,R0  ;SEE IF THE RESULT IS CORRECT.
        MOV      #2,R3
3$:     CMP     (R0)+,(R2)+
        BNE    10$
        SOB     R3,3$
  
```

```

123242 012601
123244 012700 000200
123250 170100
123252 010100
123254 172410
123256 012702 123370
123262 012700 000004
123266 012722 177777
123272 077003
123274 016100 000020
123300 170100
123302 012700 123370
123306 175410
123310 170204
123312 170305
123314 010102
123316 062702 000010
123322 012700 123370
123326 012703 000002
123332 022022
123334 001014
123336 077303
  
```

L1

```
23856 123340 016102 000022      MOV      22(R1),R2
23857 123344 020204              CMP      R2,R4      ;SEE IF THE FPS IS CORRECT
23858 123346 001007              BNE     10$         ;BRANCH IF INCORRECT
23859 123350 005702              TST     R2
23860 123352 100003              BPL     4$
23861 123354 026105 000026      CMP      26(R1),R5  ;SEE IF THE FEC IS CORRECT.
23862 123360 001002              BNE     10$         ;BRANCH IF INCORRECT.
23863
23864 123362 000161 000030      4$:     JMP      30(R1) ;RETURN.
23865 123366              10$:
(2) 123366 104000              EMT
23866
23867 ;DATA BUFFER:
23868 123370 177777 177777 177777 STCIBF: .WORD 1, 1, 1, -1
123376 177777
23869
23870 123400              WWC DONE:
(1) 123400 004767 001146              JSR     PC,.RSET   ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
23871
23872
23880 ;*****
(2) ;TEST 604      STCFI AND STCFI TEST
(3) ;*****
(2) 123404      TS604:
23881
23882
23883 ;EXPONENT=37, FL=1
23884 123404 004737 123242              JSR     PC,@STCSUB ;GO EXECUTE THE INSTRUCTION.
23885 123410 047777 177777 177777 1$: .WORD 47777,-1,-1,-1 ;ACO OPERAND.
123416 177777
23886 123420 077777 177600              2$: .WORD 77777,177600 ;EXPECTED RESULT.
23887 123424 077777 177777              3$: .WORD 77777,177777 ;ANTICIPATED ERRONEOUS RESULT.
23888 123430 040100              4$: 40100 ;FPS BEFORE EXECUTION.
23889 123432 040100              40100 ;FPS AFTER EXECUTION.
23890 123434 177777              -1 ;ANTICIPATED ERRONEOUS FPS.
23891 123436 177777              -1 ;EXPECTED FEC.
23892 123440              XXC DONE:
(1) 123440 004767 001106              JSR     PC,.RSET   ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
23893
23894
23901 ;*****
(2) ;TEST 605      STEXP TEST
(3) ;*****
(2) 123444      TS605:
23902
23903 ; EXP = 100 (EXCESS 200)
23904 123444 004737 123652              YYC1: JSR     PC,@STXSUB
23905 123450 020000 0000.0 000000 1$: .WORD 20000,0,0,0 ;AC
```

```

23906 123456 000000
23907 123460 177700
23908 123462 052525
23909 123464 040000
23910 123466 040010
23911 123470 040000
23912 123472 004737 123652
23913 123476 040000 000000 000000
23914 123504 000000
23915 123506 000000
23916 123510 052525
23917 123512 040000
23918 123514 040004
23919 123516 040000
23920 123520 004737 123652
23921 123524 040200 000000 000000
23922 123532 000000
23923 123534 000001
23924 123536 052525
23925 123540 040000
23926 123542 040000
23927 123544 040004
23928
23929 123546 004737 123652
23930 123552 077200 000000 000000
23931 123560 000000
23932 123562 000175
23933 123564 052525
23934 123566 040000
23935 123570 040000
23936 123572 040010
23937
23938 123574 004737 123652
23939 123600 000200 000000 000000
23940 123606 000000
23941 123610 177601
23942 123612 052525
23943 123614 040000
23944 123616 040010
23945 123620 040000
23946
23947 123622 004737 123652
23948 123626 033400 000000 000000
23949 123634 000000
23950 123636 177756
23951 123640 052525
23952 123642 047707
23953 123644 047710
23954 123646 177777
23955 123650 000452

```

2#: 100 ; EXP RES
3#: 52525 ; ERROR EXP.
4#: 40000 ; FPSB
40010 ; FPSA
40000 ; ERROR FPS
; EXP = 200 (EXCESS 200)
YYC2: JSR PC, @STXSUB ; GO EXECUTE THE INSTRUCTION.
1#: .WORD 40000,0,0,0 ; ACO OPERAND.
2#: 0 ; EXPECTED EXPONENT RESULT.
3#: 52525 ; ANTICIPATED ERRONEOUS RESULT.
4#: 40000 ; FPS BEFORE EXECUTION.
40004 ; FPS AFTER EXECUTION.
40000 ; ANTICIPATED ERRONEOUS FPS.
; EXP = 201 (EXCESS 200)
YYC3: JSR PC, @STXSUB ; GO EXECUTE THE INSTRUCTION.
1#: .WORD 40200,0,0,0 ; ACO OPERAND.
2#: 1 ; EXPECTED EXPONENT RESULT.
3#: 52525 ; ANTICIPATED ERRONEOUS RESULT.
4#: 40000 ; FPS BEFORE EXECUTION.
40000 ; FPS AFTER EXECUTION.
40004 ; ANTICIPATED ERRONEOUS FPS.
; EXP = 375 (EXCESS 200)
YYC4: JSR PC, @STXSUB ; GO EXECUTE THE INSTRUCTION.
1#: .WORD 77200,0,0,0 ; ACO OPERAND.
2#: 175 ; EXPECTED EXPONENT RESULT.
3#: 52525 ; ANTICIPATED ERRONEOUS RESULT.
4#: 40000 ; FPS BEFORE EXECUTION.
40000 ; FPS AFTER EXECUTION.
40010 ; ANTICIPATED ERRONEOUS FPS.
; EXP = 1 (EXCESS 200)
YYC5: JSR PC, @STXSUB ; GO EXECUTE THE INSTRUCTION.
1#: .WORD 200,0,0,0 ; ACO OPERAND.
2#: -177 ; EXPECTED EXPONENT RESULT.
3#: 52525 ; ANTICIPATED ERRONEOUS RESULT.
4#: 40000 ; FPS BEFORE EXECUTION.
40010 ; FPS AFTER EXECUTION.
40000 ; ANTICIPATED ERRONEOUS FPS.
; EXP = 156 (EXCESS 200)
YYC6: JSR PC, @STXSUB ; GO EXECUTE THE INSTRUCTION.
1#: .WORD 33400,0,0,0 ; ACO OPERAND.
2#: -22 ; EXPECTED EXPONENT RESULT.
3#: 52525 ; ANTICIPATED ERRONEOUS RESULT.
4#: 47707 ; FPS BEFORE EXECUTION.
47710 ; FPS AFTER EXECUTION.
-1 ; ANTICIPATED ERRONEOUS FPS.
BR YYCDONE

;THIS SUBROUTINE, STXSUB, IS USED TO SET UP THE OPERANDS, EXECUTE
;THE STEXP INSTRUCTION AND CHECK THE RESULTS. A CALL
;TO IT IS MADE THUS:

```

:
:          JSR      PC,@STXSUB
:          ACARG:  .WORD  X,X,X,X          ;AC OPERAND
:          RES:    .WORD  X                ;EXPECTED RESULT
:          ERRES:  .WORD  X                ;ERROR RESULT
:          FPSB:   .WORD  X                ;FPS BEFORE EXECUTION
:          FPSA:   .WORD  X                ;FPS AFTER EXECUTION
:          ERFPS:  .WORD  X                ;ERROR FPS.
:          ERR1:   ERROR X                ;DATA ERROR.
:          BR      CONT
:          ERR2:   ERROR X                ;FPS ERROR.
:          CONT:   ;RETURN ADDRESS

```

```

;THE OPERANDS ARE SET UP (USING ACO AS THE ACCUMULATOR). THEN
;THE STEXP INSTRUCTION IS EXECUTED.
;THE RESULT IS CHECKED AGAINST RES. IF THE RESULT IS CORRECT THEN THE FPS IS
;COMPARED WITH FPSA IF THIS TOO IS CORRECT STXSUB RETURNS CONTROL
;TO THE CALLING ROUTINE AT CONT. IF THE FPS IS BAD STXSUB
;COMPARE IT TO ERROR FPS. IF THIS MATCHES THEN STXSUB WILL RETURN
;TO THE ERROR CALL AT ERR2, OTHERWISE STXSUB ITSELF
;REPORTS THIS FAILURE AND THEN RETURNS TO CONT. IF THE RESULT OF THE
;STEXP IS INCORRECT, THE INCORRECT RESULT IS COMPARED WITH THE
;ANTICIPATED FAILING DATA PATTERN, ERRES. IF THE FAILURE IN
;THE RESULT WAS ANTICIPATED CORRECTLY TO BE ERRES THEN STXSUB
;WILL TRANSFER CONTROL TO THE ERROR CALL AT ERR1. OTHERWISE THE
;RESULT WAS INCORRECT BUT WAS NOT ANTICIPATED AND STXSUB WILL
;REPORT THE FAILURE AFTER WHICH CONTROL WILL BE PASSED TO CONT.

```

```

STXSUB: MOV      (SP)+,R1          ;GET A POINTER TO THE ARGUMENTS.
        MOV      R1,R2
        MOV      @123456,@STXBF
        MOV      @76543,@STXBF+2
        MOV      @200,R0
        LDFPS    R0
        MOV      R1,R0          ;SET UP THE ACO OPERAND.
        LDD      (R0),ACO
        MOV      16(R1),R0      ;SET THE FPS.
        LDFPS    R0
        MOV      @STXBF,R0
1$:     STEXP    ACO,(R0)        ;TEST INSTRUCTION.
        STFPS    R4            ;GET FPS.
        CMP      10(R1),@STXBF ;WAS RESULT CORRECT?
        BEQ      5$
        EMT
        CMP      R4,16(R1)     ;SEE IF THE FPS IS CORRECT.
        BEQ      10$
        EMT
;SEE IF MORE THAN ONE WORD WAS WRITTEN IN THE OUTPUT BUFFER.
10$:    CMP      @76543,@STXBF+2
        BEQ      4$
        EMT
        JMP      22(R1)
4$:

```

```

23956
23957
23958
23959
23960
23961
23962
23963
23964
23965
23966
23967
23968
23969
23970
23971
23972
23973
23974
23975
23976
23977
23978
23979
23980
23981
23982
23983
23984
23985
23986
23987 123652 012601
23988 123654 010102
23989 123656 012737 123456 123764
23990 123664 012737 076543 123766
23991 123672 012700 000200
23992 123676 170100
23993 123700 010100
23994 123702 172410
23995 123704 016100 000016
23996 123710 170100
23997 123712 012700 123764
23998 123716 175010
23999 123720 170204
24000 123722 026137 000010 123764
24001 123730 001401
      (2) 123732 104000
24002 123734 020461 000016
24003 123740 001401
      (2) 123742 104000
24004
24005 123744 022737 076543 123766
24006 123752 001401
      (2) 123754 104000
24007 123756 000161 000022
24008

```

24009 123762 177777
24010 123764 177777 177777
123772 177777 177777

177777 STXBF: .WORD 1
1. 1. 1. 1. 1

24011
24012 123776
(1) 123776 004767 000550
(1)
(1)
(1)
(1)

YYCDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK, AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

24013
24024
(2)
(3)
(2)

;.....
;TEST 606 STST TEST
;.....
TS606:

24025
24026 124002 012700 040000
24027 124006 170100
24028
24029 124010 170003
24030
24031 124012 012700 124072
24032 124016 012710 177777
24033 124022 012760 177777 000002
24034 124030 170310
24035
24036 124032 170204
24037 124034 012700 124072
24038 124040 022710 000002
24039 124044 001010
24040 124046 022760 124010 000002
24041 124054 001004
24042 124056 022704 140000
24043 124062 001001
24044 124064 000407
24045 124066
(2) 124066 104000

MOV #40000,R0 ;SET FPS. FID=1.
LDFPS R0
ZZC2: .WORD 170003 ;ILLEGAL FPP
;OP CODE
MOV #ZZCBF,R0 ;SET UP THE OUTPUT BUFFER.
MOV #-1,(R0)
MOV #-1,2(R0)
ZZC3: STST (R0) ;GET FEC AND
;FEA
STFPS R4 ;GET FPS.
MOV #ZZCBF,R0
CMP #2,(R0) ;SEE IF FEC IS CORRECT.
BNE ZZC10 ;BRANCH IF INCORRECT.
CMP #ZZC2,2(R0) ;SEE IF FEA, ADDRESS, IS CORRECT.
BNE ZZC10 ;BRANCH IF INCORRECT.
CMP #140000,R4 ;SEE IF FPS IS CORRECT.
BNE ZZC10 ;BRANCH IF INCORRECT.
BR ZZCDONE
ZZC10: EMT ;

24046
24047
24048 124070 177777
24049 124072 177777 177777 177777
124100 177777
24050 124102 177777
24051

;DATA BUFFER:
-1
ZZCBF: .WORD -1,-1,-1,-1
-1

24052 124104
(1) 124104 004767 000442
(1)
(1)
(1)
(1)

ZZCDONE:
JSR PC,.RSET ;GO INITIALIZE THE FPS AND STACK, AND
;SEE IF THE USER HAS EXPRESSED
;THE DESIRE TO CHANGE THE SOFTWARE
;VIRTUAL CONSOLE SWITCH REGISTER (HAS
;THE USER TYPED CONTROL G?).

24053
24054
24055
(2)
(3)
(2)

;.....
;TEST 607 SPECIAL CASE TEST
;.....
TS607:

124110

```

24056 124110 012746 144724      AAD1:  MOV    #144724, -(SP)  ;PUT FRACTION ON STACK
24057 124114 012746 040600      MOV    #040600, -(SP)  ;PUT EXPONENT ON STACK
24058 124120 005046                CLR    -(SP)           ;PUT SUBTRAHEND FRACTION ON STACK
24059 124122 012746 040600      MOV    #040600, -(SP)  ;PUT SUBTRAHEND EXPONENT ON STACK
24060 124126 172466 000004      LDF   4(SP), ACO      ;LOAD FP ACCUMULATORS
24061 124132 173026                SUBF  (SP), ACO      ;DO SUBTRACTION
24062 124134 174037 124164      STF   ACO, B#AADBF    ;GET AND STORE ANSWER
24063 124140 022737 036711 124164  CVM   #36711, B#AADBF ;IS EXPONENT CORRECT
24064 124146 001401                BEQ   1#              ;
(2) 124150 104000                EMT                       ;BAD EXPONENT FROM SUBTRACTION
24065 124152 022737 152000 124166 1#:  CMP   #152000, B#AADBF*2 ;IS FRACTION CORRECT
24066 124160 001403                BEQ   AADDONE          ;FRACTION INCORRECT
(2) 124162 104000                EMT                       ;
24067
24068 124164 000000      AADBF: .WORD 0
24069 124166 000000      .WORD 0
24070
24071 124170 012706 001000      AADDONE: MOV #STBOT, SP ;RESTORE STACK POINTER
24072 124174 004767 000352      JSR   PC, .RSET        ;GO INITIALIZE THE FPS AND STACK; AND
(1)                                     ;SEE IF THE USER HAS EXPRESSED
(1)                                     ;THE DESIRE TO CHANGE THE SOFTWARE
(1)                                     ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1)                                     ;THE USER TYPED CONTROL G?).
24073
24074
24075
24076
(2)
(3)
(2) 124200
24077 124200 004567 124416      BDD1:  JSR   R5,CHKAPT ;SKIP TEST IF ON APT AND NOT FIRST PASS
24078 124204 000520      BR    R5,CHKAPT ;
24079
24080 124206 005001                CLR   R1              ;INITIALIZE A COUPLE OF COUNTERS
24081 124210 005000                CLR   R0              ;
24082 124212 013767 000064 113352  MOV   B#64, 1TMP0     ;SAVE INTERRUPT VECTOR
24083 124220 013767 000066 113346  MOV   B#66, 1TMP1     ;SAVE INTERRUPT PRIORITY
24084 124226 012737 124300 000064  MOV   #31, B#64       ;SET UP INTERRUPT PRIORITY FOR THIS TEST
24085 124234 005037 000066                CLR   B#66           ;AND PRIORITY
24086 124240 005067 053532                CLR   PS              ;PUT PROCESSOR PRIORITY AT 0
24087 124244 005067 053316                CLR   TPB             ;SEND A NULL CHARACTER
24088 124250 105767 053310      1#:  TSTB  TTCSR        ;WAIT FOR DONE TO SET
24089 124254 100375                BPL   1#              ;
24090 124256 005067 053304                CLR   TPB             ;SEND A SECOND CHARACTER
24091 124262 052767 000100 053274  BIS   #BIT6, TTCSR    ;SET INTERRUPT ENABLE
24092 124270 005200                INC   R0              ;INCREMENT COUNTER TO GET BASE TIME
24093 124272 001376                BNE  2#              ;CONTINUE LOOPING UNLESS COUNTER GOES TO 0
24094 124274 000005                RESET                ;IF NO INTERRUPT YET KILL IT
24095 124276 104000                EMT                       ;NO INTERRUPT OCCURRED IN ALLOTTED TIME
24096 124300 166700 000110      3#:  SUB   Y, R0        ;SUBTRACT TIME FOR FP INSTRUCTION
24097 124304 010067 000106                MOV   R0, Z          ;SAVE FIRST TIME
24098 124310 012737 124366 000064  MOV   #7, B#64       ;SET UP FOR NEXT INTERRUPT
24099 124316 005100                COM   R0              ;MAKE PRE LOOP COUNTER NEGATIVE
24100 124320 005067 053240                CLR   TTCSR          ;MAKE SURE NO INTERRUPT YET
24101 124324 005067 053236                CLR   TPB            ;SEND A CHARACTER
24102 124330 105767 053230      5#:  TSTB  TTCSR        ;WAIT FOR READY BIT TO SET

```

.....
 ;TEST 610 INTERRUPTABILITY TEST
 ;.....

```

TS610:
BDD1:  JSR   R5,CHKAPT ;SKIP TEST IF ON APT AND NOT FIRST PASS
      BR    R5,CHKAPT ;
      CLR   R1              ;INITIALIZE A COUPLE OF COUNTERS
      CLR   R0              ;
      MOV   B#64, 1TMP0     ;SAVE INTERRUPT VECTOR
      MOV   B#66, 1TMP1     ;SAVE INTERRUPT PRIORITY
      MOV   #31, B#64       ;SET UP INTERRUPT PRIORITY FOR THIS TEST
      CLR   B#66           ;AND PRIORITY
      CLR   PS              ;PUT PROCESSOR PRIORITY AT 0
      CLR   TPB             ;SEND A NULL CHARACTER
      1#:  TSTB  TTCSR        ;WAIT FOR DONE TO SET
      BPL   1#              ;
      CLR   TPB             ;SEND A SECOND CHARACTER
      BIS   #BIT6, TTCSR    ;SET INTERRUPT ENABLE
      2#:  INC   R0              ;INCREMENT COUNTER TO GET BASE TIME
      BNE  2#              ;CONTINUE LOOPING UNLESS COUNTER GOES TO 0
      RESET                ;IF NO INTERRUPT YET KILL IT
      EMT                       ;NO INTERRUPT OCCURRED IN ALLOTTED TIME
      3#:  SUB   Y, R0        ;SUBTRACT TIME FOR FP INSTRUCTION
      MOV   R0, Z          ;SAVE FIRST TIME
      MOV   #7, B#64       ;SET UP FOR NEXT INTERRUPT
      4#:  COM   R0              ;MAKE PRE LOOP COUNTER NEGATIVE
      CLR   TTCSR          ;MAKE SURE NO INTERRUPT YET
      CLR   TPB            ;SEND A CHARACTER
      5#:  TSTB  TTCSR        ;WAIT FOR READY BIT TO SET

```



```

24103 124334 100375          BPL      5:
24104 124336 005067 053224  CLR      TPB
24105 124342 052767 000100 053214  BIS      #BIT6, TTCSR ;SEND SECOND CHARACTER
24106 124350 005200          INC      R0          ;SET INTERRUPT ENABLE
24107 124352 001376          BNE      6:          ;DO PRE LOOP
24108 124354 171227 040400          Mulf    #2, AC2     ;DO FLOATING POINT INSTRUCTION
24109 124360 000240          NOP
24110 124362 000005          RESET
24111 124364 104000          EMT
24112 124366 005201          7:      INC      R1          ;JUST IN CASE INTERRUPT TAKES TOO LONG
24113 124370 020127 000015          CMP     R1, #15     ;IF NO INTERRUPT CLEAR THE WORLD
24114 124374 001411          BEQ     BBDDONE     ;INCREMENT TIMES THROUGH COUNTER
24115 124376 062767 000002 000012  ADD     #2, Z        ;HAVE WE PASSED HERE 15 TIMES BEFORE
24116 124404 016700 000006          MOV     Z, R0       ;IF YES I MAY NEVER PASS HERE AGAIN
24117 124410 000742          BR      4:          ;IF NO ADD A LITTLE TIME TO PRELOOP
24118
24119 124412 000000          X:      .WORD    0   ;PUT NEW COUNT IN COUNTER
24120 124414 000026          Y:      .WORD    26 ;DO IT ALL AGAIN
24121 124416 000000          Z:      .WORD    0
24122
24123 124420 042767 000100 053136 BBDDONE: BIC     #100, TTCSR ;CLEAR INTERRUPT ENABLE BEFORE EXITING TEST
24124 124426 016737 113140 000064          MOV     $TMP0, #064 ;RESTORE PRINTER VECTOR
24125 124434 016737 113134 000066          MOV     $TMP1, #066 ;RESTORE PRINTER PRIORITY
24126 124442 004767 000104          JSR     PC, .RSET   ;GO INITIALIZE THE FPS AND STACK; AND
(1) ;SEE IF THE USER HAS EXPRESSED
(1) ;THE DESIRE TO CHANGE THE SOFTWARE
(1) ;VIRTUAL CONSOLE SWITCH REGISTER (HAS
(1) ;THE USER TYPED CONTROL G?).
24127 124446 000167 000164          FPEXIT: JMP     SLUIST ;GET OVER SUBROUTINES TO NEXT TEST
24128
24129
24130 124452 004767 006752          ERROR4: JSR     PC, ABORT ;ARE WE UNDER UFG ?
24131 124456 012737 000003 001002          MOV     #3, #FATAL ;SET UP FATAL ERROR NUMBER
24132 124464 012767 000001 054306          MOV     #1, #MSGTY ;SET FATAL ERROR FLAG
24133 124472 032737 000001 001020          BIT     #1, #ENV    ;UNDER APT
24134 124500 001004          BNE     FPHLT      ;YES
24135 124502 012700 124514          MOV     #FMSG, R0
24136 124506 004767 006652          JSR     PC, TYPE
24137 124512 000777          FPHLT: BR      .          ;STAY HERE FOREVER
24138
24139 124514 040506 046111 042105          FMSG:  .ASCIZ  /FAILED DURING THE FPP TESTS/<12><15>
124522 042040 051125 047111
124530 020107 044124 020105
124536 050106 020120 042524
124544 052123 005123 000015

.EVEN

.SBTTL FLAG RESET ROUTINE
;*****
;*THIS ROUTINE WILL BE CALLED AT THE END OF EACH FLOATING POINT TEST
;*TO RESET THE STACK, CLEAR THE FPS AND REINITIALIZE TRAP VECTORS
24147 124552 012737 124452 000244 .RSET: MOV     #ERROR4, #FPVECT
24148 124560 012737 021526 000004          MOV     #T04, #ERRVECT
24149 124566 012737 021530 000010          MOV     #T010, #010
24150 124574 011600          MOV     (SP), R0

```

E.2

```

24151 124576 012706 001000      MOV     #STBOT,SP
24152 124602 005004      CLR     R4
24153 124604 170104      LDFPS  R4
24154 124606 000110      JMP     (R0)
24155
24156      ;THESE ARE SOME EQUATES USED IN THE PROGRAM
24157      BIT0=000001
24158      BIT1=000002
24159      BIT2=000004
24160      BIT3=000010
24161      BIT4=000020
24162      BIT5=000040
24163      BIT6=000100
24164      BIT7=000200
24165      BIT8=000400
24166      BIT9=001000
24167      BIT10=002000
24168      BIT11=004000
24169      BIT12=010000
24170      BIT13=020000
24171      BIT14=040000
24172      BIT15=100000
24173
24174 124610 177560      RCSR:   177560      ;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
24175 124612 177562      RBUF:   177562      ;ADDRESS OF RECEIVER BUFFER
24176 124614 177564      TCSR:   177564      ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
24177 124616 177566      TBUF:   177566      ;ADDRESS OF TRANSMITTER BUFFER
24178 124620 000060      RVECT:  60          ;RECEIVER INTERRUPT VECTOR
24179 124622 000062      RPSW:   62
24180 124624 000064      TVECT:  64          ;TRANSMITTER INTERRUPT VECTOR
24181 124626 000066      TPSW:   66
24182
24183      ;REAL TIME CLOCK REGISTER AND VECTOR ADDRESSES
24184 124630 177546      LKS:    .WORD      177546
24185 124632 000100      RTCVT:  .WORD      100
24186 124634 000102      RTCPSW: .WORD      102
24187
24188 124636 000244      SLU1ST: CLZ
24189 124640 032777 000004 074730      BIT     #4,BSWR
24190 124646 001402      BEQ
24191 124650 000167 001112      JMP     KWSTRT
24192 124654 012737 000004 001004 1#:     MOV     #4,#TESTN      ;PUT TEST NUMBER IN MAILBOX
24193 124662 012737 125700 000030      MOV     @ERROR5,#30      ;SET UP FOR CORRECT ERROR CALL
24194
24195
24196      ;*****
24197      ;TEST 611 TEST ABILITY TO REFERENCE TCSR
24198      ;*****
24199      TS611:
24197 124670 013703 000004      MOV     @#4,R3          ;SAVE TIMEOUT VECTOR
24198 124674 012737 124710 000004      MOV     #1,#@#4        ;SET UP TIMEOUT VECTOR
24199 124702 005777 177706      TST     @TCSR          ;REFERENCE THE XMIT COMMAND/STATUS REG.
24200 124706 000401      BR     #4
24199 (1) 124710      1#:
24199 (2) 124710 104000      EMT
24201 124712 010337 000004      4#:     MOV     R3,@#4        ;RESTORE TIMEOUT VECTOR

```

24202
24203
24204
24205
(2)
(3)
(2) 124716
24206 124716 013703 000004
24207 124722 012737 124736 000004
24208 124730 005777 177662
24209 124734 000401
(1) 124736
(2) 124736 104000
24210 124740 010337 000004
24211
24212
24213
(2)
(3)
(2) 124744
24214 124744 032737 000001 001020
24215 124752 001405
24216 124754 005737 001006
24217 124760 001402
24218 124762 000167 001000
24219 124766 005077 177624
24220 124772 105777 177616
24221 124776 100006
24222
24223
24224 125000 005077 177612
24225 125004 105777 177604
24226 125010 100001
(2) 125012 104000
24227 125014 005000
24228 125016 105777 177572
24229 125022 100403
24230 125024 005200
24231 125026 001373
(2) 125030 104000
24232 125032
24233
24234
24235
(2)
(3)
(2) 125032
24236 125032 005077 177560
24237 125036 105777 177552
24238 125042 100375
24239 125044 005077 177546
24240 125050 000240
24241 125052 000005
24242 125054 105777 177534
24243 125060 100401
(3) 125062 104000

```
*****  
;TEST 612 TEST ABILITY TO REFERENCE TBUF  
*****  
TS612:  
MOV R3, R3 ;SAVE TIMEOUT VECTOR  
MOV R1, R0 ;SET UP TIMEOUT VECTOR  
TST RBUF ;REFERENCE THE XMIT BUFFER  
BR 4 ;  
  
1 ;  
4 ; EMT ;  
MOV R3, R0 ;RESTORE TIMEOUT VECTOR  
  
*****  
;TEST 613 TEST THAT TCSR BIT7(DONE) CLEARS WHEN XBUF IS LOADED  
*****  
TS613:  
BIT R0, R0+ENV ;ARE WE RUNNING UNDER APT  
BEQ 70 ;IF NO THEN SERIES OF TESTS  
TST R0+PASS ;IS THIS FIRST PASS  
BEQ 70 ;IF YES THEN DO SERIES OF TESTS  
JMP KWSTRT ;IF NO THEN BYPASS SERIES OF TESTS  
70 ; CLR RBUF ;LOAD XBUF  
TSTB R0+TCSR ;CHECK DONE  
BPL 3 ;BR IF CLEAR  
 ;FILL SECOND BUFFER BECUASE REFRESH COULD CAUSE  
 ;FIRST TEST TO FAIL  
 ;FILL DOUBLE BUFFER  
 ;CHECK DONE  
  
3 ; CLR R0 ;  
4 ; TSTB R0+TCSR ;CLEAR TIMER  
 ;CHECK FOR XMIT DONE  
 ;IF DONE SETS, BR TO END OF TEST  
 ;INCREMENT TIMER  
  
5 ;  
  
*****  
;TEST 614 TEST THAT TCSR "DONE" SETS WITH RESET  
*****  
TS614:  
1 ; CLR RBUF ;LOAD TRANSMIT BUFFER  
TSTB R0+TCSR ;WAIT FOR DONE  
BPL 1 ;  
CLR RBUF ;LOAD SECOND BUFFER  
NOP ;  
RESET ;SET DONE WITH RESET  
TSTB R0+TCSR ;CHECK FOR DONE SET  
BMI TS615 ;  
EMT ;
```

```

24244
24245
24246
24247
(2)
(3)
(2) 125064
24248 125064 013703 000004
24249 125070 012737 125104 000004
24250 125076 005777 177506
24251 125102 000401
(1) 125104
(2) 125104 104000
24252 125106 010337 000004
24253
24254
24255
(2)
(3)
(2) 125112
24256 125112 013703 000004
24257 125116 012737 125132 000004
24258 125124 005777 177462
24259 125130 000401
(1) 125132
(2) 125132 104000
24260 125134 010337 000004
24261
24262
24263
24264
24265
(2)
(3)
(2) 125140
24266 125140 017703 177454
24267 125144 012777 125166 177446
24268 125152 106427 000340
24269 125156 032777 000100 177424
24270 125164 001401
(1) 125166
(2) 125166 104000
24271 125170 052777 000100 177412
24272 125176 032777 000100 177404
24273 125204 001001
(2) 125206 104000
24274 125210 042777 000100 177372
24275 125216 032777 000100 177364
24276 125224 001401
(2) 125226 104000
24277 125230
24278 125230 052777 000100 177352
24279 125236 000005
24280 125240 032777 000100 177342
24281 125246 001401
(2) 125250 104000
;*****
;TEST 615 TEST APTLITY TO ACCESS RCSR
;*****
TS615:
MOV R04,R3 ;SAVE TIMEOUT VECTOR
MOV #11,R04 ;SET UP TIMEOUT VECTOR
TST BRCSR ;ACCESS RCSR
BR 21
11:
EMT ;
21: MOV R3,R04 ;RESTORE TIMEOUT VECTOR
;*****
;TEST 616 TEST ABILITY TO ACCESS RBUF
;*****
TS616:
MOV R04,R3 ;SAVE TIMEOUT VECTOR
MOV #11,R04 ;SET UP TIMEOUT VECTOR
TST BRBUF ;ACCESS RBUF
BR 21
11:
EMT ;
21: MOV R3,R04 ;RESTORE TIMEOUT VECTOR
;*****
;TEST 617 TEST THAT BIT6 OF RCSR CAN BE SET & RESET
;*****
TS617:
MOV BRVECT,R3 ;SAVE RECEIVE VECTOR
MOV #11,BRVECT ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
MTPS #340 ;SET PSM TO PRIORITY 7
BIT #BIT6,BRCSR ;TEST BIT6 OF RCSR
BEQ 21
11:
EMT ;
21: BIS #BIT6,BRCSR ;SET BIT6 OF RCSR
BIT #BIT6,BRCSR ;TEST BIT6 OF RCSR
BNE 31
EMT ;
31: BIC #BIT6,BRCSR ;CLEAR BIT6 OF RCSR
BIT #BIT6,BRCSR ;TEST BIT6 OF RCSR
BEQ 41
EMT ;
41:
BIS #BIT6,BRCSR ;SET BIT6 OF RCSR
RESET ;CLEAR BIT6 OF RCSR WITH RESET
BIT #BIT6,BRCSR ;TEST BIT6 OF RCSR
BEQ 51
EMT ;
51:

```

24282 125252 010377 177342

54: MOV R3,@RVECT ;RESTORE RECEIVE VECTOR

24283
24284
24285
24286

;*****
;TEST 620 TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
;*****

(2)
(3)
(2)

TS620:

24287 125256 042777 000100 177330

BIC #BIT6,@TCSR ;CLEAR TRANSMIT INTERRUPT ENABLE

24288 125264 017703 177334

MOV @TVECT,R3 ;SAVE XMIT VECTOR

24289 125270 012777 125312 177326

MOV #24,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT

24290 125276 105777 177312

14: TSTB @TCSR ;WAIT FOR DONE

24291 125302 100375

BPL 14

24292 125304 106427 000140

MTPS #140 ;SET PSW TO PRIORITY 3

24293 125310 000401

BR 34

(1) 125312

24:

(2) 125312 104000

EMT ;

24294 125314 012777 125334 177302

34: MOV #44,@TVECT ;SET XMIT VECTOR TO END OF TEST

24295 125322 052777 000100 177264

BIS #BIT6,@TCSR ;ENABLE INTERRUPTS

24296 125330 000240

NOP

24297

24298 125332 104000

EMT ;XMIT DID NOT INTERRUPT

24299

24300 125334 042777 000100 177252

44: BIC #BIT6,@TCSR ;DISABLE INTERRUPTS

24301 125342 022626

CMPS (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT

24302 125344 010377 177254

MOV R3,@TVECT ;RESTORE XMIT VECTOR

24303

24304

24305

;*****
;TEST 621 TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
;*****

(2)

(3)

(2)

TS621:

24306 125350 042777 000100 177236

BIC #BIT6,@TCSR ;DISABLE INTERRUPTS

24307 125356 106427 000340

MTPS #340 ;SET PSW TO PRIORITY 7

24308 125362 017703 177236

MOV @TVECT,R3 ;SAVE XMIT VECTOR

24309 125366 012777 125414 177230

MOV #24,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT

24310 125374 105777 177214

14: TSTB @TCSR ;WAIT FOR DONE

24311 125400 100375

BPL 14

24312 125402 052777 000100 177204

BIS #BIT6,@TCSR ;ENABLE INTERRUPT

24313 125410 000240

NOP

24314 125412 000401

BR 34

(1) 125414

24:

(2) 125414 104000

EMT ;

24315 125416 042777 000100 177170

34: BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE

24316 125424 012777 125442 177172

MOV #44,@TVECT ;POINT XMIT VECTOR TO ERROR REPORT

24317 125432 106427 000140

MTPS #140 ;SET PSW TO PRIORITY 3

24318 125436 000240

NOP

24319 125440 000401

BR 54

(1) 125442

44:

(2) 125442 104000

EMT ;

24320 125444 010377 177154

54: MOV R3,@TVECT ;RESTORE XMIT VECTOR

24321

24322

24323

;*****
;TEST 622 TEST TRANSMITTER FOR DOUBLE INTERRUPTS
;*****

(2)

(3)

```

(2) 125450
24324 125450 042777 000100 177136 TS622: BIC #BIT6,@TCSR ;CLEAR INTERRUPT ENABLE
24325 125456 017703 177142 MOV @TVECT,R3 ;SAVE XMIT VECTOR
24326 125462 017704 177140 MOV @TPSW,R4 ;SAVE XMIT PSW VECTOR
24327 125466 012777 125526 177130 MOV #2,@TVECT ;SET UP XMIT VECTOR
24328 125474 012777 000340 177124 MOV #340,@TPSW ;SET PIO 7 AFTER INTERRUPT
24329 125502 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
24330 125506 105777 177102 1#: TSTB @TCSR ;WAIT FOR DONE
24331 125512 100375 BPL 1#
24332 125514 052777 000100 177072 BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
24333 125522 000240 NOP
24334
24335 125524 104000 EMT ;
24336 ;XMIT INTERRUPT DID NOT OCCUR
24337 125526 022626 2#: CMP (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
24338 125530 012777 125554 177066 MOV #4,@TVECT ;POINT XMIT VECTOR TO ERROR
24339 125536 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
24340 125542 000240 NOP ;GIVE TIME FOR ANY INTERRUPTS
24341 125544 042777 000100 177042 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
24342 125552 000401 BR 5#
(1) 125554 4#: EMT ;
(2) 125554 104000 5#: MOV R3,@TVECT ;RESTORE XMIT VECTOR
24343 125556 010377 177042 MOV R4,@TPSW ;RESTORE XMIT PSW VECTOR
24344 125562 010477 177040
24345
24346 ;*****
(2) ;TEST 623 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF
(3) ;*****
(2) 125566 TS623:
24347 125566 042777 000100 177020 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
24348 125574 106427 000340 MTPS #340 ;SET PSW TO PRIORITY 7
24349 125600 017703 177020 MOV @TVECT,R3 ;SAVE XMIT VECTOR
24350 125604 012777 125654 177012 MOV #2,@TVECT ;POINT XMIT VECTOR TO ERROR
24351 125612 052777 000100 176774 BIS #BIT6,@TCSR ;ENABLE INTERRUPTS
24352 125620 005077 176772 CLR @TBUF ;LOAD TBUF
24353 125624 105777 176764 1#: TSTB @TCSR ;WAIT FOR DONE (INTERRUPT)
24354 125630 100375 BPL 1#
24355 125632 005077 176760 CLR @TBUF ;FILL SECOND BUFFER TO RESET INT.
24356 125636 106427 000140 MTPS #140 ;SET PSW TO PRIORITY 3
24357 125642 000240 NOP ;GIVE TIME FOR ANY INTERRUPTS
24358 125644 042777 000100 176742 BIC #BIT6,@TCSR ;DISABLE INTERRUPTS
24359 125652 000401 BR 3#
(1) 125654 2#: EMT ;
(2) 125654 104000 3#: MOV R3,@TVECT ;RESTORE XMIT VECTOR
24360 125656 010377 176742 CLR R0 ;INITIALIZE LOOP COUNTER
24361 125662 005000 4#: INC R0 ;INCREMENT LOOP COUNTER
24362 125664 005200 BNE 4# ; UNTIL COUNTER = 0
24363 125666 001376 TST @RBUF ;CLEAR RECEIVER BUFFER
24364 125670 005777 176716 JMP KWSTRT ;GET TO NEXT TEST
24365 125674 000167 000066
24366
24367 125700 004767 005524 ERRORS: JSR PC,ABORT ;ARE WE UNDER UFD ?
24368 125704 012737 000004 001002 MOV #4,@#FATAL ;SET UP FATAL ERROR NUMBER
24369 125712 012767 000001 053060 MOV #1,@MSGTY ;SET FATAL ERROR FLAG
24370 125720 032737 000001 001020 BIT #1,@#ENV ;UNDER APR ?
24371 125726 001004 BNE SLIMLT

```

JL

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07 JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 31-17
T623 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF

SEQ 0436

24372	125730	012700	125742						MOV	#SL1MSG,R0	
24373	125734	004767	005424						JSR	PC,TYPE	
24374	125740	000777						SL1HLT:	BR	.	
24375											
24376	125742	040506	046111	042105				SL1MSG:	.ASCIZ	/FAILED SLU1 TEST/<12><15>	
	125750	051440	052514	020061							
	125756	042524	052123	006412							
	125764	000									
24377		125766						.EVEN			
24378											
24379											
24380	125766	000244						KWSTRT:	CLZ		
24381	125770	032777	000010	073600					BIT	#10,#SWR	
24382	125776	001402							BEQ	1#	
24383	126000	000167	000734						JMP	SLU2ST	
24384	126004	012737	000005	001004			1#:		MOV	#5,#TESTN	;PUT TEST NUMBER IN MAILBOX
24385	126012	012737	126624	000030					MOV	#ERROR6,#30	;SET UP ERROR CALL
24386											
24387	126020							LKSTST:			
24388											
(2)											
(3)											
(2)	126020										
24389	126020	013703	000004						MOV	#4,R3	;SAVE TIMEOUT VECTOR
24390	126024	012737	126040	000004					MOV	#1#,#4	;SET UP TIMEOUT VECTOR
24391	126032	005777	176572						TST	#LKS	;ACCESS LKS
24392	126036	000401							BR	2#	
(1)	126040						1#:				
(2)	126040	104000							EMT		
24393	126042	010337	000004				2#:		MOV	R3,#4	;RESTORE TIMEOUT VECTOR
24394											
24395											
(2)											
(3)											
(2)	126046										
24396	126046	017703	176560						MOV	#RTRCVT,R3	;SAVE LINE CLOCK VECTOR
24397	126052	012777	126074	176552					MOV	#1#,#RTRCVT	;SET UP INTERRUPT VECTOR FOR ERROR REPORT
24398	126060	106427	000340						MTPS	#340	;SET PSM TO PRIORITY 7
24399	126064	032777	000100	176536					BIT	#BIT6,#LKS	;TEST BIT6 OF LKS
24400	126072	001401							BEQ	2#	
(1)	126074						1#:				
(2)	126074	104000							EMT		
24401	126076	052777	000100	176524			2#:		BIS	#BIT6,#LKS	;SET BIT6 OF LKS
24402	126104	032777	000100	176516					BIT	#BIT6,#LKS	;TEST BIT6 OF LKS
24403	126112	001001							BNE	3#	
(2)	126114	104000							EMT		
24404	126116	042777	000100	176504			3#:		BIC	#BIT6,#LKS	;CLEAR BIT6 OF LKS
24405	126124	032777	000100	176476					BIT	#BIT6,#LKS	;TEST BIT6 OF LK
24406	126132	001401							BEQ	4#	
(2)	126134	104000							EMT		
24407	126136	032737	000001	001020			4#:		BIT	#1,#ENV	;ARE WE RUNNING UNDER APT
24408	126144	001403							BEQ	70#	;IF NO THEN DO TEST
24409	126146	005737	001006						TST	#PASS	;IS THIS FIRST PASS
24410	126152	001011							BNE	5#	;IF NO SKIP TO TEST END
24411	126154						70#:				
24412	126154	052777	000100	176446					BIS	#BIT6,#LKS	;SET BIT6 OF LKS

```

24413 126162 000005          RESET          ;CLEAR BIT6 OF LKS WITH RESET
24414 126164 032777 000100 176436 BIT        #BIT6,BLKS ;TEST BIT6 OF LKS
24415 126172 001401          BEQ          5;
(2) 126174 104000          EMT
24416 126176 010377 176430 5;: MOV      R3,RTCVT ;RESTORE LINE CLOCK VECTOR
24417
24418
24419 ;*****
(2) ;TEST 626 TEST THAT THE REAL TIME CLOCK INTERRUPTS PROPERLY
(3) ;*****
(2) 126202 TS626:
24420
24421 126202 106427 000340          MTPS        #340 ;SET PSW TO PRIORITY 7
24422 126206 017703 176420          MOV        RTCVT,R3 ;SAVE LINE CLOCK VECTOR
24423 126212 017704 176416          MOV        RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR
24424 126216 012777 126260 176406          MOV        #ESR51,RTCVT ;SET RTC INTERRUPT VECTOR TO ERROR REPORT
24425 126224 012777 000340 176402          MOV        #340,RTCP SW ;KEEP PRIORITY AT 7
24426 126232 052777 000100 176370          BIS        #BIT6,BLKS ;SET INTERRUPT ENABLE
24427
24428 126240 012701 024000          MOV        #24000,R1 ;SET UP A WAIT LOOP
24429 126244 077101          SOB        R1,ESR36 ;WAIT 30 MILLISEC
24430 126246 012777 126262 176356          MOV        #ESR37,RTCVT ;ALTER VECTOR
24431 126254 106427 000240          MTPS        #240 ;PRIORITY NOW TO FIVE
24432 126260 104000          EMT
24433 126262 012777 126300 176342          ESR37: MOV    #ESR38,RTCVT
24434 126270 012701 024000          MOV        #24000,R1
24435 126274 077101          ESR39: SOB    R1,ESR39 ;WAIT 30 MORE MILLISEC
24436 126276 000401          BR        ESR38+2
24437 126300 104000          ESR38: EMT ;AN ERROR IF WE ARE HERE
24438 126302 005077 176322          CLR        BLKS ;CLR INTERRUPT ENABLE
24439 126306 012701 024000          MOV        #24000,R1
24440 126312 077101          ESR52: SOB    R1,ESR52 ;WAIT LOOP
24441 126314 106427 000240          MTPS        #240 ;ALTER PRIORITY TO FIVE
24442 126320 012701 024000          MOV        #24000,R1
24443 126324 077101          ESR53: SOB    R1,ESR53 ;... AGAIN
24444 126326 012777 126352 176276          MOV        #ESR55,RTCVT
24445 126334 052777 000100 176266          BIS        #BIT6,BLKS ;ENABLE LTC INTERRUPTS
24446 126342 012701 024000          MOV        #24000,R1
24447 126346 077101          ESR54: SOB    R1,ESR54
24448 126350 104000          EMT ;SHOULD HAVE INTERRUPTED
24449
24450 126352 022626          ESR55: CMP    (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
24451 126354 042777 000100 176246          BIC        #BIT6,BLKS ;DISABLE INTERRUPTS
24452 126362 010377 176244          MOV        R3,RTCVT ;RESTORE LINE CLOCK VECTOR
24453 126366 010477 176242          MOV        R4,RTCP SW ;RESTORE LINE CLOCK PSW VECTOR
24454
24455
24456
24457
24458
24459 ;*****
(2) ;TEST 627 TEST RTC FOR DOUBLE INTERRUPTS
(3) ;*****
(2) 126372 TS627:
24460 126372 032737 000001 001020          BIT        #1, #ENV ;ARE WE RUNNING UNDER APT
(1) 126400 001403          BEQ        70; ;IF NO THEN DO TEST

```



```

(1) 126402 005737 001006
(2) 126406 001045
(1) 126410
24461 126410 017703 176216
24462 126414 017704 176214
24463 126420 012777 126462 176204
24464 126426 012777 000340 176200
24465 126434 106427 000240
24466
24467 126440 005000
24468 126442 052777 000100 176160
24469 126450 005200
24470 126452 005700
24471 126454 100375
24472 126456 000240
24473
24474 126460 104000
24475
24476 126462 022626
24477 126464 012777 126502 176140
24478 126472 106427 000240
24479 126476 000240
24480 126500 000401
(1) 126502
(2) 126502 104000
24481 126504 042777 000100 176116
24482 126512 010377 176114
24483 126516 010477 176112
24484
24485
24486
(2)
(3)
(2) 126522
24487 126522 032737 000001 001020
(1) 126530 001403
(1) 126532 005737 001006
(2) 126536 001115
(1) 126540
24488 126540 106427 000340
24489 126544 017703 176062
24490 126550 012777 126614 176054
24491 126556 005000
24492
24493 126560 052777 000100 176042
24494 126566 005200
24495 126570 005700
24496 126572 100375
24497 126574 000005
24498 126576 106427 000240
24499 126602 000240
24500 126604 042777 000100 176016
24501 126612 000401
(1) 126614
(2) 126614 104000
24502 126616 010377 176010

```

```

TST 001PASS ;IS THIS FIRST PASS
BNE TS630 ;IF NO THEN SHIP TO NEXT TEST
70$:
MOV 0RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV 0RTCP SW,R4 ;SAVE LINE CLOCK PSW VECTOR
MOV 021,0RTCVT ;SET UP RTC INTERRUPT VECTOR
MOV 0340,0RTCP SW ;DISALLOW INTERRUPTS AFTER THE INTERRUPT
MTPS 0240 ;SET PSW TO PRIORITY 5
CLR R0
BIS 0BIT6,0LKS ;ENABLE CLOCK INTERRUPTS
1$:
INC R0
TST R0
BPL 1$
NOP ;GIVE TIME FOR ANY INTERRUPT
EMT ;RTC INTERRUPT DID NOT OCCUR
2$:
CMP (SP), (SP) ;RESTORE SP AFTER INTERRUPT
MOV 031,0RTCVT ;POINT RTC VECTOR TO ERROR REPORT
MTPS 0240 ;SET PSW TO PRIORITY 5
NOP ;GIVE SOME TIME FOR AN INTERRUPT
BR 4$
3$:
EMT
4$:
BIC 0BIT6,0LKS ;DISABLE CLOCK INTERRUPTS
MOV R3,0RTCVT ;RESTORE LINE CLOCK VECTOR
MOV R4,0RTCP SW ;RESTORE LINE CLOCK PSW VECTOR

```

```

;*****
;TEST 630 TEST THAT RTC INTERRUPT CLEARS WITH RESET
;*****

```

```

TS630:
BIT 01, 001ENV ;ARE WE RUNNING UNDER APT
BEQ 70$ ;IF NO THEN DO TEST
TST 001PASS ;IS THIS FIRST PASS
BNE TS631 ;IF NO THEN SHIP TO NEXT TEST
70$:
MTPS 0340 ;SET PSW TO PRIORITY 7
MOV 0RTCVT,R3 ;SAVE LINE CLOCK VECTOR
MOV 021,0RTCVT ;POINT RTC VECTOR TO ERROR REPORT
CLR R0
BIS 0BIT6,0LKS ;ENABLE CLOCK INTERRUPTS
1$:
INC R0
TST R0
BPL 1$
RESET ;CLEAR PENDING INTERRUPT WITH RESET
MTPS 0240 ;SET PSW TO PRIORITY 5
NOP ;GIVE TIME FOR ANY INTERRUPT
BIC 0BIT6,0LKS ;DISALLOW INTERRUPTS
BR 3$
2$:
EMT
3$:
MOV R3,0RTCVT ;RESTORE LINE CLOCK VECTOR

```

```

24503 126622 000446 BR SLU2ST
24504
24505
24506
24507 126624 004767 004600 ERROR6: JSR PC,ABORT ;ARE WE UNDER UFD ?
24508 126630 012737 000005 001002 MOV #5,#FATAL ;SET UP FATAL ERROR NUMBER
24509 126636 012767 000001 052134 MOV #1,#MSGTY ;SET FATAL ERROR FLAG
24510 126644 032737 000001 001020 BIT #1,#ENV ;UNDER APT /
24511 126652 001004 BNE LTCHLT
24512 126654 012700 126666 MOV #LTCMSG,R0
24513 126660 004767 004500 JSR PC,TYPE
24514 126664 000777 LTCHLT: BR
24515 126666 040506 046111 051125 LTCMSG: .ASCIZ /FAILURE DURING LTC TEST/<12><15>
126674 020105 052504 044522
126702 043516 046040 041524
126710 052040 051505 005124
126716 000015

24516 .EVEN
24517
24518 ;SERIAL LINE UNIT REGISTER AND VECTOR ADDRESSES FOR SLU2
24519
24520 126720 176500 RCSR2: 176500 ;ADDRESS OF RECEIVER COMMAND/STATUS REGISTER
24521 126722 176502 RBUF2: 176502 ;ADDRESS OF RECEIVER BUFFER
24522 126724 176504 TCSR2: 176504 ;ADDRESS OF TRANSMITTER COMMAND/STATUS REGISTER
24523 126726 176506 TBUF2: 176506 ;ADDRESS OF TRANSMITTER BUFFER
24524 126730 000300 RVECT2: 300 ;RECEIVER INTERRUPT VECTOR
24525 126732 000302 RPSM2: 302
24526 126734 000304 TVECT2: 304 ;TRANSMITTER INTERRUPT VECTOR
24527 126736 000306 TPSM2: 306
24528
24529 126740 000244 SLU2ST: CLZ
24530 126742 032777 000020 072626 BIT #20,#SWR
24531 126750 001402 BEQ 1#
24532 126752 000167 002626 JMP UNIQUE
24533 126756 012737 000006 001004 1#: MOV #6,#TESTN ;PUT TEST NUMBER IN MAILBOX
24534 126764 012737 131474 000030 MOV #ERROR7,#30 ;SET UP FOR CORRECT ERROR CALL
24535
24536
24537 ;*****
(2) ;TEST 631 TEST ABILITY TO REFERENCE TCSR2
(3) ;*****
(2) 126772 TS631:
24538 126772 013703 000004 MOV #04,R3 ;SAVE TIMEOUT VECTOR
24539 126776 012737 127012 000004 MOV #14,#04 ;SET UP TIMEOUT VECTOR
24540 127004 005777 177714 TST #TCSR2 ;REFERENCE THE XMIT COMMAND/STATUS REG.
24541 127010 000401 BR 4#
(1) 127012 1#:
(2) 127012 104000 EMT
24542 127014 010337 000004 4#: MOV R3,#04 ;RESTORE TIMEOUT VECTOR
24543
24544
24545
24546 ;*****
(2) ;TEST 632 TEST ABILITY TO REFERENCE TBUF2
(3) ;*****
(2) 127020 TS632:

```

```

24547 127020 013703 000004
24548 127024 012737 127040 000004
24549 127032 005777 177670
24550 127036 000401
(1) 127040 1$
(2) 127040 104000
24551 127042 010337 000004 4$: EMT
MOV R3,0#4 ;RESTORE TIMEOUT VECTOR
24552
24553
24554 ;*****
(2) ;TEST 633 TEST THAT TCSR2 BIT7(DONE) CLEARS WHEN XBUF IS LOADED
(3) ;*****
(2) 127046 TS633:
24555 127046 032737 000001 001020 BIT #1,0#ENV ;ARE WE RUNNING UNDER APT
(1) 127054 001403 BEQ 70$ ;IF NO THEN DO TEST
(1) 127056 005737 001006 TST 0#PASS ;IS THIS FIRST PASS
(2) 127062 001022 BNE TS634 ;IF NO THEN SHIP TO NEXT TEST
(1) 127064 70$:
24556 127064 005077 177636 CLR @TBUF2 ;LOAD XBUF
24557 127070 105777 177630 TSTB @TCSR2 ;CHECK DONE
24558 127074 100006 BPL 3$ ;BR IF CLEAR
24559 ;FILL SECOND BUFFER BECUASE REFRESH COULD CAUSE
24560 ;FIRST TEST TO FAIL
24561 127076 005077 177624 CLR @TBUF2 ;FILL DOUBLE BUFFER
24562 127102 105777 177616 TSTB @TCSR2 ;CHECK DONE
24563 127106 100001 BPL 3$
(2) 127110 104000 EMT ;
24564 127112 005000 3$: CLR R0 ;CLEAR TIMER
24565 127114 105777 177604 4$: TSTB @TCSR2 ;CHECK FOR XMIT DONE
24566 127120 100403 BMI 5$ ;IF DONE SETS, BR TO END OF TEST
24567 127122 005200 INC R0 ;INCREMENT TIMER
24568 127124 001373 BNE 4$
(2) 127126 104000 EMT ;
24569 127130 5$:
24570
24571
24572 ;*****
(2) ;TEST 634 TEST THAT TCSR2 "DONE" SETS WITH RESET
(3) ;*****
(2) 127130 TS634:
24573 127130 032737 000001 001020 BIT #1,0#ENV ;ARE WE RUNNING UNDER APT
(1) 127136 001403 BEQ 70$ ;IF NO THEN DO TEST
(1) 127140 005737 001006 TST 0#PASS ;IS THIS FIRST PASS
(2) 127144 001015 BNE TS635 ;IF NO THEN SHIP TO NEXT TEST
(1) 127146 70$:
24574 127146 005077 177554 CLR @TBUF2 ;LOAD TRANSMIT BUFFER
24575 127152 105777 177546 1$: TSTB @TCSR2 ;WAIT FOR DONE
24576 127156 100375 BPL 1$
24577 127160 005077 177542 CLR @TBUF2 ;LOAD SECOND BUFFER
24578 127164 000240 NOP
24579 127166 000005 RESET ;SET DONE WITH RESET
24580 127170 105777 177530 TSTB @TCSR2 ;CHECK FOR DONE SET
24581 127174 100401 BMI TS635
(3) 127176 104000 EMT ;
24582
24583

```

```

24584
24585
(2)
(3)
(2) 127200
24586 127200 013703 000004
24587 127204 012737 127220 000004
24588 127212 005777 177502
24589 127216 000401
(1) 127220
(2) 127220 104000
24590 127222 010337 000004
24591
24592
24593
(2)
(3)
(2) 127226
24594 127226 013703 000004
24595 127232 012737 127246 000004
24596 127240 005777 177456
24597 127244 000401
(1) 127246
(2) 127246 104000
24598 127250 010337 000004
24599
24600
24601
24602
24603
24604
24605
(2)
(3)
(2) 127254
24606 127254 032777 000001 177442
24607 127262 001401
(2) 127264 104000
24608 127266 052777 000001 177430
24609 127274 032777 000001 177422
24610 127302 001001
(2) 127304 104000
24611 127306 042777 000001 177410
24612 127314 032777 000001 177402
24613 127322 001401
(2) 127324 104000
24614 127326
(1) 127326 032737 000001 001020
(1) 127334 001403
(1) 127336 005737 001006
(2) 127342 001011
(1) 127344
24615 127344 052777 000001 177352
24616 127352 000005
24617 127354 032777 000001 177342
24618 127362 001401

```

```

;.....
;TEST 635 TEST ABILITY TO ACCESS RCSR2
;.....

```

```

TS635:
MOV R04,R3 ;SAVE TIMEOUT VECTOR
MOV R01,R04 ;SET UP TIMEOUT VECTOR
TST RCSR2 ;ACCESS RCSR
BR 2#
1#:
EHT
2#: MOV R3,R04 ;RESTORE TIMEOUT VECTOR

```

```

;.....
;TEST 636 TEST ABILITY TO ACCESS RBUF2
;.....

```

```

TS636:
MOV R04,R3 ;SAVE TIMEOUT VECTOR
MOV R01,R04 ;SET UP TIMEOUT VECTOR
TST RBUF2 ;ACCESS RBUF
BR 2#
1#:
EHT
2#: MOV R3,R04 ;RESTORE TIMEOUT VECTOR

```

```

;.....
;TEST 637 TEST THAT BITO(BREAK BIT) CAN BE SET & CLEARED & RESET
;.....

```

```

TS637:
BIT #BITO,#TCSR2 ;CHECK BITO OF TCSR CLEAR
BEQ 3#
EHT
3#: BIS #BITO,#TCSR2 ;SET BITO IN TCSR
BIT #BITO,#TCSR2 ;TEST BITO OF TCSR
BNE 4#
EHT
4#: BIC #BITO,#TCSR2 ;CLEAR BITO OF TCSR
BIT #BITO,#TCSR2 ;TEST BITO OF TCSR
BEQ 7#
EHT
7#:
BIT #1,#ENV ;ARE WE RUNNING UNDER APT
BEQ 70# ;IF NO THEN DO TEST
TST #PASS ;IS THIS FIRST PASS
BNE TS640 ;IF NO THEN SHIP TO NEXT TEST
70#:
BIS #BITO,#TCSR2 ;SET BITO IN TCSR
RESET ;CLEAR BITO WITH RESET
BIT #BITO,#TCSR2 ;TEST BITO CLEAR
BEQ TS640

```

```

(3) 127364 104000
24619
24620
24621
(2)
(3)
(2) 127366
24622 127366 017703 177342
24623 127372 012777 127414 177334
24624 127400 106427 000340
24625 127404 032777 000100 177312
24626 127412 001401
(1) 127414
(2) 127414 104000
24627 127416 052777 000100 177300
24628 127424 032777 000100 177272
24629 127432 001001
(2) 127434 104000
24630 127436 042777 000100 177260
24631 127444 032777 000100 177252
24632 127452 001401
(2) 127454 104000
24633 127456 032737 000001 001020
24634 127464 001403
24635 127466 005737 001006
24636 127472 001011
24637 127474
24638 127474 052777 000100 177222
24639 127502 000005
24640 127504 032777 000100 177212
24641 127512 001401
(2) 127514 104000
24642 127516 010377 177212
24643
24644
24645
(2)
(3)
(2) 127522
24646 127522 017703 177202
24647 127526 012777 127550 177174
24648 127534 106427 000340
24649 127540 032777 000100 177152
24650 127546 001401
(1) 127550
(2) 127550 104000
24651 127552 052777 000100 177140
24652 127560 032777 000100 177132
24653 127566 001001
(2) 127570 104000
24654 127572 042777 000100 177120
24655 127600 032777 000100 177112
24656 127606 001401
(2) 127610 104000
24657 127612 032737 000001 001020
24658 127620 001403

```

EMT

1

```

;*****
;TEST 640 TEST THAT BIT6(XMIT INT EN) CAN BE SET & RESET
;*****

```

```

TS640:
MOV    @TVECT2,R3      ;SAVE XMIT VECTOR
MOV    #1,@TVECT2     ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
MTPS   #340           ;SET PSW TO PRIORITY 7
BIT    @BIT6,@TCSR2   ;TEST BIT6 OF TCSR
BEQ    21

11:
EMT

21:
BIS    @BIT6,@TCSR2   ;SET BIT6 OF TCSR
BIT    @BIT6,@TCSR2   ;TEST BIT6 OF TCSR
BNE    31

31:
BIC    @BIT6,@TCSR2   ;CLEAR BIT6 OF TCSR
BIT    @BIT6,@TCSR2   ;TEST BIT6 OF TCSR
BEQ    41

41:
EMT
BIT    #1,@ENV        ;ARE WE RUNNING UNDER APT
BEQ    701
TST    @PASS          ;IF THIS FIRST PASS
BNE    51             ;IF NO THEN SKIP TO END OF TEST

701:
BIS    @BIT6,@TCSR2   ;SET BIT6 OF TCSR
RESET  @BIT6,@TCSR2   ;CLEAR BIT6 WITH RESET
BIT    @BIT6,@TCSR2   ;TEST BIT6 OF TCSR
BEQ    51

51:
EMT
MOV    R3,@TVECT2     ;RESTORE XMIT VECTOR

```

```

;*****
;TEST 641 TEST THAT BIT6 OF RCSR2 CAN BE SET & RESET
;*****

```

```

TS641:
MOV    @RVECT2,R3     ;SAVE RECEIVE VECTOR
MOV    #1,@RVECT2    ;SET UP INTERRUPT VECTOR FOR ERROR REPORT
MTPS   #340           ;SET PSW TO PRIORITY 7
BIT    @BIT6,@RCSR2  ;TEST BIT6 OF RCSR
BEQ    21

11:
EMT

21:
BIS    @BIT6,@RCSR2  ;SET BIT6 OF RCSR
BIT    @BIT6,@RCSR2  ;TEST BIT6 OF RCSR
BNE    31

31:
BIC    @BIT6,@RCSR2  ;CLEAR BIT6 OF RCSR
BIT    @BIT6,@RCSR2  ;TEST BIT6 OF RCSR
BEQ    41

41:
EMT
BIT    #1,@ENV        ;ARE WE RUNNING UNDER APT
BEQ    701

```

```

24659 127622 005737 001006          TST      #01PASS      ;IS THIS FIRST PASS
24660 127626 001011                   BNE      51           ;IF NO THEN SKIP TO END OF TEST
24661 127630                               701:
24662 127630 052777 000100 177062    BIS      #BIT6,RCSR2  ;SET BIT6 OF RCSR
24663 127636 000005                   RESET     ;CLEAR BIT6 OF RCSR2 WITH RESFT
24664 127640 032777 000100 177052    BIT      #BIT6,RCSR2  ;TEST BIT6 OF RCSR
24665 127646 001401                   BEQ      51
(2) 127650 104000                   EMT
24666 127652 010377 177052          51:     MOV      R3,RVECT2    ;RESTORE RECEIVE VECTOR
24667
24668
24669
24670

```

```

;*****
;TEST 642      TEST THAT XMIT INTERRUPTS ONLY WHEN ENABLED
;*****

```

```

(2) 127656
24671 127656 042777 000100 177040    TS642:  BIC      #BIT6,RTCSR2 ;CLEAR TRANSMIT INTERRUPT ENABLE
24672 127664 017703 177044           MOV      RTVECT2,R3   ;SAVE XMIT VECTOR
24673 127670 012777 127712 177036    MOV      #21,RTVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
24674 127676 105777 177022          11:     TSTB     RTCSR2      ;WAIT FOR DONE
24675 127702 100375                   BPL      11
24676 127704 106427 000140           MTPS     #140         ;SET PSW TO PRIORITY 3
24677 127710 000401                   BR       31
(1) 127712          21:
(2) 127712 104000                   EMT
24678 127714 012777 127734 177012    31:     MOV      #41,RTVECT2 ;SET XMIT VECTOR TO END OF TEST
24679 127722 052777 000100 176774    BIS      #BIT6,RTCSR2 ;ENABLE INTERRUPTS
24680 127730 000240                   NOP
24681
24682 127732 104000                   EMT      ;XMIT DID NOT INTERRUPT
24683
24684 127734 042777 000100 176762    41:     BIC      #BIT6,RTCSR2 ;DISABLE INTERRUPTS
24685 127742 022626                   CMP      (SP)+,(SP)+ ;RESTORE SP AFTER INTERRUPT
24686 127744 010377 176764           MOV      R3,RTVECT2  ;RESTORE XMIT VECTOR
24687
24688

```

```

;*****
;TEST 643      TEST THAT XMIT INTERRUPTS DO NOT OCCUR WHEN DISABLED
;*****

```

```

(2) 127750
24690 127750 042777 000100 176746    TS643:  BIC      #BIT6,RTCSR2 ;DISABLE INTERRUPTS
24691 127756 106427 000340           MTPS     #340         ;SET PSW TO PRIORITY 7
24692 127762 017703 176746           MOV      RTVECT2,R3   ;SAVE XMIT VECTOR
24693 127766 012777 130014 176740    MOV      #21,RTVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
24694 127774 105777 176724          11:     TSTB     RTCSR2      ;WAIT FOR DONE
24695 130000 100375                   BPL      11
24696 130002 052777 000100 176714    BIS      #BIT6,RTCSR2 ;ENABLE INTERRUPT
24697 130010 000240                   NOP
24698 130012 000401                   BR       31
(1) 130014          21:
(2) 130014 104000                   EMT
24699 130016 042777 000100 176700    31:     BIC      #BIT6,RTCSR2 ;CLEAR INTERRUPT ENABLE
24700 130024 012777 130042 176702    MOV      #41,RTVECT2 ;POINT XMIT VECTOR TO ERROR REPORT
24701 130032 106427 000140           MTPS     #140         ;SET PSW TO PRIORITY 3
24702 130036 000240                   NOP
24703 130040 000401                   BR       51

```

```

(1) 130042
(2) 130042 104000
24704 130044 010377 176664
24705
24706
24707
(2)
(3)
(2) 130050
24708 130050 042777 000100 176646
24709 130056 017703 176652
24710 130062 017704 176650
24711 130066 012777 130126 176640
24712 130074 012777 000340 176634
24713 130102 106427 000140
24714 130106 105777 176612
24715 130112 100375
24716 130114 052777 000100 176602
24717 130122 000240
24718
24719 130124 104000
24720
24721 130126 022626
24722 130130 012777 130154 176576
24723 130136 106427 000140
24724 130142 000240
24725 130144 042777 000100 176552
24726 130152 000401
(1) 130154
(2) 130154 104000
24727 130156 010377 176552
24728 130162 010477 176550
24729
24730
(2)
(3)
(2) 130166
24731 130166 032737 000001 001020
(1) 130174 001403
(1) 130176 005737 001006
(2) 130202 001043
(1) 130204
24732 130204 042777 000100 176512
24733 130212 106427 000340
24734 130216 017703 176512
24735 130222 012777 130272 176504
24736 130230 052777 000100 176466
24737 130236 005077 176464
24738 130242 105777 176456
24739 130246 100375
24740 130250 005077 176452
24741 130254 106427 000140
24742 130260 000240
24743 130262 042777 000100 176434
24744 130270 000401
(1) 130272

```

```

4: EMT
5: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR

;*****
;TEST 644 TEST TRANSMITTER FOR DOUBLE INTERRUPTS
;*****
TS644:
BIC #BIT6,@TCSR2 ;CLEAR INTERRUPT ENABLE
MOV @TVECT2,R3 ;SAVE XMIT VECTOR
MOV @TPSW2,R4 ;SAVE XMIT PSW VECTOR
MOV #2,@TVECT2 ;SET UP XMIT VECTOR
MOV #340,@TPSW2 ;SET PIO 7 AFTER INTERRUPT
MTPS #140 ;SET PSW TO PRIORITY 3
1: TSTB @TCSR2 ;WAIT FOR DONE
BPL 1:
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPTS
NOP

EMT
;XMIT INTERRUPT DID NOT OCCUR
2: CMP (SP),(SP) ;RESTORE SP AFTER INTERRUPT
MOV #4,@TVECT2 ;POINT XMIT VECTOR TO ERROR
MTPS #140 ;SET PSW TO PRIORITY 3
NOP ;GIVE TIME FOR ANY INTERRUPTS
BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS
BR 5:

4: EMT
5: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR
MOV R4,@TPSW2 ;RESTORE XMIT PSW VECTOR

;*****
;TEST 645 TEST THAT XMIT INTERRUPT CLEARS WITH LOADING TBUF2
;*****
TS645:
BIT #1,@#ENV ;ARE WE RUNNING UNDER APT
BEQ 70: ;IF NO THEN DO TEST
TST #PASS ;IS THIS FIRST PASS
BNE TS646 ;IF NO THEN SHIP TO NEXT TEST

70: BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS
MTPS #340 ;SET PSW TO PRIORITY 3
MOV @TVECT2,R3 ;SAVE XMIT VECTOR
MOV #2,@TVECT2 ;POINT XMIT VECTOR TO ERROR
BIS #BIT6,@TCSR2 ;ENABLE INTERRUPTS
CLR @TBUF2 ;LOAD TBUF
1: TSTB @TCSR2 ;WAIT FOR DONE (INTERRUPT)
BPL 1:
CLR @TBUF2 ;FILL SECOND BUFFER TO RESET INT.
MTPS #140 ;SET PSW TO PRIORITY 3
NOP ;GIVE TIME FOR ANY INTERRUPTS
BIC #BIT6,@TCSR2 ;DISABLE INTERRUPTS
BR 3:

2:

```

(2) 130272 104000
24745 130274 010377 176434
24746 130300 005000
24747 130302 005200
24748 130304 001376
24749 130306 005777 176410
24750
24751
24752

EMT
3\$: MOV R3,@TVECT2 ;RESTORE XMIT VECTOR
CLR R0 ;INIT LOOP COUNTER
4\$: INC R0 ;INCREMENT COUNTER
BNE 4\$;UNTIL COUNTER = 0
TST @RBUF2 ;CLEAR RECEIVER BUFFER

;*****
;TEST 646 TEST THAT RCVR DONE (7) SET & CLEAR PROPERLY
;*****

TS646:

(2) 130312
24753 130312 005000
24754 130314 005077 176406
24755 130320 105777 176374
24756 130324 100403
24757 130326 005200
24758 130330 001373
(2) 130332 104000

CLR R0 ;CLEAR A TIMER
CLR @TBUF2 ;LOAD TRANSMIT BUFFER
WDONE2: TSTB @RCSR2 ;CHECK FOR RECEIVER DONE
BMI 6\$;BR, IF DONE
INC R0 ;INCREMENT TIMER, IF NOT DONE
BNE WDONE2
EMT ;RECEIVER DONE NEVER SET

24759
24760 130334 032737 000001 001020
24761 130342 001403
24762 130344 005737 001006
24763 130350 001005

6\$: BIT #1,@ENV ;ARE WE RUNNING UNDER APT
BEQ 70\$;IF NO THEN DO TEST
TST @PASS ;IS THIS FIRST PASS
BNE 2\$;IF NO THEN SKIP TO END OF TEST

24764 130352
24765 130352 000005
24766 130354 105777 176340
24767 130360 001401

70\$: RESET ;CLEAR DONE WITH RESET
TSTB @RCSR2 ;CHECK FOR DONE CLEAR
BEQ 2\$
EMT ;RESET DID NOT CLEAR RCVR DONE

(2) 130362 104000
24768 130364 005000
24769 130366 005200
24770 130370 001376
24771 130372 005777 176324
24772
24773
24774

2\$: CLR R0 ;INIT LOOP COUNTER
3\$: INC R0 ;INCREMENT COUNTER
BNE 3\$;UNTIL COUNTER = 0
TST @RBUF2 ;CLEAR RECEIVER BUFFER

;*****
;TEST 647 TEST THAT READING RBUF2 CLEARS RECEIVER DONE
;*****

TS647:

(2) 130376
24775 130376 005077 176324
24776 130402 105777 176312
24777 130406 100375
24778 130410 017700 176306
24779 130414 105777 176300
24780 130420 001401
(3) 130422 104000

1\$: CLR @TBUF2 ;LOAD TRANSMITTER
TSTB @RCSR2 ;WAIT FOR RECEIVER DONE
BPL 1\$
MOV @RBUF2,R0 ;READ RECEIVE BUFFER
TSTB @RCSR2 ;CHECK FOR RECEIVE DONE CLEAR
BEQ TS650
EMT ;READING RBUF2 DID NOT CLEAR RCVR DONE

24781
24782
24783
24784
(2) 130424

;*****
;TEST 650 TEST THAT RCVR INTERRUPTS ONLY WHEN ENABLED
;*****

TS650:

(2) 130424
24785 130424 042777 000100 176272
24786 130432 042777 000100 176260
24787 130440 017703 176264

BIC #BIT6,@TCSR2 ;DISABLE TRANSMIT INTERRUPTS
BIC #BIT6,@RCSR2 ;DISABLE RECEIVER INTERRUPTS
MOV @RVECT2,R3 ;SAVE RECEIVE VECTOR

24788	130444	012777	130472	176256		MOV	#2#,BRVECT2	POINT RCV VECTOR TO ERROR REPORT
24789	130452	106427	000140			MTPS	#140	SET PSW TO PRIORITY 3
24790	130456	005077	176244			CLR	@TBUF2	SEND A CHARACTER
24791	130462	105777	176232		1#:	TSTB	@RCSR2	WAIT FOR RECEIVER DONE
24792	130466	100375				BPL	1#	
24793	130470	000401				BR	3#	
(1)	130472				2#:			
(2)	130472	104000				EMT		
24794	130474	012777	130514	176226	3#:	MOV	#4#,BRVECT2	POINT RCV VECTOR TO END OF TEST
24795	130502	052777	000100	176210		BIS	@BIT6,@RCSR2	ENABLE RCV INTERRUPTS
24796	130510	000240				NOP		GIVE ANY INTERRUPTS TIME
24797	130512	104000				EMT		
24798	130514	042777	000100	176176	4#:	BIC	@BIT6,@RCSR2	DISABLE INTERRUPTS
24799	130522	022626				CMR	(SP)*,(SP)*	RESTORE SP AFTER INTERRUPT
24800	130524	005777	176172			TST	@RBUF2	CLEAR CHARACTER FROM RECEIVER BUFFER
24801	130530	010377	176174			MOV	R3,@RVECT2	RESTORE RECEIVE VECTOR

;TEST 651 TEST THAT RCVR INTERRUPTS DO NOT OCCUR WHEN DISABLED

(2)	130534					TS651:		
24805	130534	106427	000340			MTPS	#340	SET PSW TO PRIORITY 7
24806	130540	017703	176164			MOV	@RVECT2,R3	SAVE RECEIVE VECTOR
24807	130544	012777	130576	176156		MOV	#2#,BRVECT2	POINT RCVR VECTOR TO ERROR REPORT
24808	130552	005077	176150			CLR	@TBUF2	SEND A CHARACTER
24809	130556	105777	176136		1#:	TSTB	@RCSR2	WAIT FOR RECEIVER DONE
24810	130562	100375				BPL	1#	
24811	130564	052777	000100	176126		BIS	@BIT6,@RCSR2	ENABLE INTERRUPTS
24812	130572	000240				NOP		GIVE TIME FOR INTERRUPT
24813	130574	000401				BR	3#	
(1)	130576				2#:			
(2)	130576	104000				EMT		RCVR INTERRUPTS AT PRIORITY 7
24814	130600	042777	000100	176112	3#:	BIC	@BIT6,@RCSR2	CLEAR INTERRUPT ENABLE
24815	130606	012777	130624	176114		MOV	#4#,BRVECT2	POINT RCVR VECTOR TO ERROR REPORT
24816	130614	106427	000140			MTPS	#140	SET PSW TO PRIORITY 3
24817	130620	000240				NOP		GIVE TIME FOR ANY INTERRUPT
24818	130622	000401				BR	5#	
(1)	130624				4#:			
(2)	130624	104000				EMT		RCVR INTERRUPT REQUEST PASSED WITH BIT6 CLEAR
24819	130626	005777	176070		5#:	TST	@RBUF2	CLEAR CHARACTER FROM RECEIVER BUFFER
24820	130632	010377	176072			MOV	R3,@RVECT2	RESTORE RECEIVE VECTOR

;TEST 652 TEST RECEIVER FOR DOUBLE INTERRUPTS

(2)	130636					TS652:		
24824	130636	017703	176066			MOV	@RVECT2,R3	SAVE RECEIVE VECTOR
24825	130642	017704	176064			MOV	@RPSW2,R4	SAVE RECEIVE PSW VECTOR
24826	130646	012777	130712	176054		MOV	#2#,BRVECT2	POINT RCVR VECTOR TO CONTINUE TEST
24827	130654	012777	000340	176050		MOV	#340,@RPSW2	SET PRIORITY TO 7 AFTER INTERRUPT
24828	130662	106427	000140			MTPS	#140	SET PSW TO PRIORITY 3
24829	130666	005077	176034			CLR	@TBUF2	SEND A CHARACTER
24830	130672	105777	176022		1#:	TSTB	@RCSR2	WAIT FOR RCVR DONE
24831	130676	100375				BPL	1#	

24832	130700	052777	000100	176012	BIS	#BIT6, BRCSR2	;ENABLE RCV INTERRUPTS	
24833	130706	000240			NOP		;GIVE SOME TIME	
24834	130710	104000			EMT			
24835	130712	022626			24:	CMP	(SP), (SP)	;RESTORE SP AFTER INTERRUPT
24836	130714	012777	130750	176006	MOV	#31, BRVECT2	;POINT RCV VECTOR TO ERROR REPORT	
24837	130722	106427	000140		MTPS	#140	;SET PSW TO PRIORITY 3	
24838	130726	000240			NOP		;GIVE SOME TIME	
24839	130730	042777	000100	175762	BIC	#BIT6, BRCSR2	;CLEAR INTERRUPT ENABLE	
24840	130736	010377	175764		MOV	R3, BRVECT2	;RESTORE RECEIVE VECTOR	
24841	130742	010477	175764		MOV	R4, BRPSW2	;RESTORE RECEIVE PSW VECTOR	
24842	130746	000401			BR	44		
(1)	130750				34:			
(2)	130750	104000			EMT			
24843	130752	005777	175744		44:	TST	BRBUF2	;CLEAR CHARACTER FROM RECEIVER BUFFER
24844	130756	010377	175746		MOV	R3, BRVECT2	;RESTORE RECEIVE VECTOR	

;TEST 653 TEST THAT RCVR INTERRUPT CLEARS BY READING RBUF2

(2)	130762				TS653:			
24848	130762	106427	000340		MTPS	#340	;SET PSW TO PRIORITY 7	
24849	130766	017703	175736		MOV	BRVECT2, R3	;SAVE RECEIVE VECTOR	
24850	130772	012777	131042	175730	MOV	#21, BRVECT2	;POINT RCV VECTOR TO ERROR REPORT	
24851	131000	052777	000100	175712	BIS	#BIT6, BRCSR2	;SET RCVR INTERRUPT ENABLE	
24852	131006	005077	175714		CLR	BTBUF2	;SEND A CHARACTER	
24853	131012	105777	175702		14:	TSTB	BRCSR2	;WAIT FOR DONE (INTERRUPT)
24854	131016	100375			BPL	14		
24855	131020	005777	175676		TST	BRBUF2	;READ RBUF TO CLEAR PENDING INTERRUPT	
24856	131024	106427	000140		MTPS	#140	;SET PSW TO PRIORITY 3	
24857	131030	000240			NOP		;ALLOW TIME FOR ANY ERRONEOUS INTERRUPT	
24858	131032	042777	000100	175660	BIC	#BIT6, BRCSR2	;NO INTERRUPT-CLEAR INT. ENABLE	
24859	131040	000401			BR	34		
(1)	131042				24:			
(2)	131042	104000			EMT			
24860	131044	010377	175660		34:	MOV	R3, BRVECT2	;RESTORE RECEIVE VECTOR

;TEST 654 TEST THAT RESET CLEARS RECEIVE INTERRUPT

(2)	131050				TS654:			
24865	131050	032737	000001	001020	BIT	#1, B#ENV	;ARE WE RUNNING UNDER APT	
(1)	131056	001403			BEQ	704	;IF NO THEN DO TEST	
(1)	131060	005737	001006		TST	B#PASS	;IS THIS FIRST PASS	
(2)	131064	001036			BNE	TS655	;IF NO THEN SHIP TO NEXT TEST	
(1)	131066				704:			
24866	131066	106427	000340		MTPS	#340	;SET PSW TO PRIORITY 7	
24867	131072	017703	175632		MOV	BRVECT2, R3	;SAVE RECEIVE VECTOR	
24868	131076	012777	131154	175624	MOV	#21, BRVECT2	;POINT RCV VECTOR TO ERROR REPORT	
24869	131104	052777	000100	175606	BIS	#BIT6, BRCSR2	;SET RCVR INTERRUPT ENABLE	
24870	131112	012777	000377	175606	MOV	#377, BTBUF2	;SEND AN ALL 1'S CHARACTER	
24871	131120	105777	175574		14:	TSTB	BRCSR2	;WAIT FOR RCV DONE
24872	131124	100375			BPL	14		
24873	131126	000005			RESET		;CLEAR RCV INTERRUPT & RBUF2	

CJKL580 LCP 5 CPL CLSTR DIAG
CJKL58.F11 07-JAN 85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 31-29
T654 TEST THAT RESET CLEARS RECEIVE INTERRUPT

SEQ 0446

24874	131130	052777	000100	175562	BIS	#BIT6, @RCSR2	;SET RECEIVER INTERRUPT
24875	131136	106427	000140		MTPS	#140	;SET PSW TO PRIORITY 3
24876	131142	000240			NOP		;ALLOW TIME FOR AN ERRONEOUS INTERRUPT
24877	131144	042777	000100	175546	BIC	#BIT6, @RCSR2	;NO INTERRUPT-CLEAR INT. ENABLE
24878	131152	000401			BR	3#	
(1)	131154			2#:			
(2)	131154	104000			EMT		
24879	131156	010377	175546	3#:	MOV	R3, @RVECT2	;RESTORE RECEIVE VECTOR

24880
24881
24882
(2)
(3)
(2)

```

;*****
;TEST 655 TEST THAT THE "OR" ERROR (BIT14) & "ERROR" (BIT15) CAN BE SET
;*****
TS655:

```

24883	131162	012700	000003		MOV	#3, R0	;SET CHARACTER COUNT TO SEND 3 CHAR.	
24884	131166	005077	175534	1#:	CLR	@TBUF2	;LOAD TRANSMIT BUFFER	
24885	131172	105777	175526	2#:	TSTB	@TCSR2	;WAIT FOR TRANSMIT DONE	
24886	131176	100375			BPL	2#		
24887	131200	005300			DEC	R0	;DECREMENT CHARACTER COUNT	
24888	131202	001371			BNE	1#	;BR IF ALL CHARACTERS NOT TRANSMITTED	
24889	131204	032777	040000	175510	BIT	#BIT14, @RBUF2	;TEST FOR "OR" ERROR FLAG	
24890	131212	001001			BNE	3#		
(2)	131214	104000			EMT			
24891	131216	032777	100000	175476	3#:	BIT	#BIT15, @RBUF2	;TEST "ERROR" FLAG
24892	131224	001001			BNE	4#		
(2)	131226	104000			EMT			
24893	131230	005000			4#:	CLR	R0	;CLEAR LOOP COUNTER
24894	131232	005200			5#:	INC	R0	;INCREMENT LOOP COUNTER
24895	131234	001376			BNE	5#	; UNTIL COUNTER = 0	
24896	131236	005777	175460		TST	@RBUF2	;CLEAR CHARACTER FROM RECEIVER BUFFER	

24897
24898
24899
(2)
(3)
(2)

```

;*****
;TEST 656 TEST THAT BREAK TRANSMITS ALL ZEROES
;*****
TS656:

```

24900	131242	032737	000001	001020	BIT	#1, @ENV	;ARE WE RUNNING UNDER APT
(1)	131250	001403			BEQ	70#	;IF NO THEN DO TEST
(1)	131252	005737	001006		TST	@PASS	;IS THIS FIRST PASS
(2)	131256	001027			BNE	TS657	;IF NO THEN SHIP TO NEXT TEST
(1)	131260			70#:			
24901	131260	012777	177777	175440	MOV	#-1, @TBUF2	;TRANSMIT ALL ONES TO RCVR
24902	131266	105777	175426	1#:	TSTB	@RCSR2	;WAIT FOR RCVR DONE
24903	131272	100375			BPL	1#	
24904	131274	005777	175422		TST	@RBUF2	;CLEAR DONE (LEAVING ALL ONES IN RBUF)
24905	131300	052777	000001	175416	BIS	#BIT0, @TCSR2	;TRANSMIT BREAK
24906	131306	005000			CLR	R0	;CLEAR A TIMER
24907	131310	105777	175404	2#:	TSTB	@RCSR2	;WAIT FOR RCVR DONE
24908	131314	100403			BMI	CONT42	;BR IF DONE
24909	131316	005200			INC	R0	;IF NOT, INCREMENT TIMER
24910	131320	001373			BNE	2#	
(2)	131322	104000			EMT		;BREAK DID NOT TRANSMIT ANYTHING
24911							
24912	131324	105777	175372	CONT42:	TSTB	@RBUF2	;CHECK RECEIVE BUFFER FOR ZERO
24913	131330	001401			BEQ	3#	
(2)	131332	104000			EMT		;BREAK DID NOT TRANSMIT ALL ZEROES

```

24914
24915 131334 000005      3$:  RESET                      ;CLEAR ERRORS
24916
24917
24918
(2)
(3)
(2) 131336
24919 131336 052777 000001 175360      BIS      @BIT0,@TCSR2      ;SEND BREAK
24920 131344 005077 175356      CLR      @TBUF2           ;TRANSMIT A CHARACTER TO TIME BREAK
24921 131350 105777 175344      1$:  TSTB     @RCSR2       ;WAIT FOR RCVR DONE
24922 131354 100375
24923 131356 042777 000001 175340      BIC      @BIT0,@TCSR2      ;CLEAR BREAK BITS
24924 131364 032777 020000 175330      BIT      @BIT13,@RBUF2     ;CHECK FOR FRAMING ERROR FLAG
24925 131372 001001
(2) 131374 104000
24926 131376 032777 100000 175316      2$:  BIT      @BIT15,@RBUF2 ;TEST "ERROR" FLAG
24927 131404 001001
(2) 131406 104000
24928 131410 005777 175306      3$:  EMT
TST      @RBUF2           ;"ERROR" FLAG DID NOT SET WITH "OR" FLAG
;CLEAR RECEIVER BUFFER
24929
24930
(2)
(3)
(2) 131414
24931 131414 005001
24932 131416 105201
24933 131420 010177 175302
24934 131424 005000
24935 131426 105777 175266      2$:  CLR      R1           ;CLEAR REGISTER FOR TEST DATA
TSTB     @RCSR2       ;INCREMENT THE TEST DATA
24936 131432 100403
24937 131434 005200
24938 131436 001373
(2) 131440 104000
24939 131442 017702 175254      3$:  MOV      @RBUF2,R2     ;XMIT A CHARACTER
CMP      R1,R2         ;CLEAR A TIMER
24940 131446 020102
24941 131450 001401
(2) 131452 104000
24942 131454 105701      4$:  TSTB     R1           ;WAIT FOR RECEIVER DONE
BNE     1$            ;BR IF DONE
24943 131456 001357
24944 131460 000167 000120      JMP      UNIQUE         ;INCREMENT TIMER IF NOT
24945
24946 131464 000000      $BDADR: 0
24947 131466 000000      $BDDAT: 0
24948 131470 000000      $GDADR: 0
24949 131472 000000      $GDDAT: 0
24950
24951
24952 131474 004767 001730      ERROR7: JSR     PC,ABORT   ;ARE WE UNDER UFD ?
24953 131500 012737 000006 001002      MOV      @6,@FATAL      ;SET UP FATAL ERROR NUMBER
24954 131506 012767 000001 047264      MOV      @1,@MSGTY      ;SET FATAL ERROR FLAG
24955 131514 032737 000001 001020      BIT      @1,@ENV        ;UNDER APT ?
24956 131522 001004
24957 131524 012700 131536      BNE     SL2HLT          ;YES
24958 131530 004767 001630      MOV      @SL2MSG,R0
24959 131534 000777      JSR     PC,TYPE
SL2HLT: BR      .

```

```

24960
24961 131536 040506 046111 051125 SL2MSG: .ASCIZ /FAILURE DURING SLU 2 TEST/<12><15>
      131544 020105 052504 044522
      131552 043516 051440 052514
      131560 031040 052040 051505
      131566 005124 000015
24962 .EVEN
24963
24964
24965 131572 177560 DADTBL: .WORD 177560
24966 131574 177564 .WORD 177564
24967 131576 176500 .WORD 176500
24968 131600 176504 .WORD 176504
24969 131602 177564 TBLEND: .WORD 177564
24970
24971
24972 131604 032737 000040 000052 UNIQUE: BIT #40,#52 ;ARE WE RUNNING UNDER UFD ?
24973 131612 001002 BNE 10# ;NO, THEN CONTINUE
24974 131614 000167 001340 JMP ENDPAS ;YES, THEN SKIP TESTS
24975 131620 032777 000034 067750 10#: BIT #34,#5WR
24976 131626 001402 BEQ 1#
24977 131630 000167 001324 JMP ENDPAS
24978 131634 012737 000007 001004 1#: MOV #7,#TESTN ;UPDATE TEST NUMBER FOR APT
24979 131642 012737 132460 000030 MOV #ERROR8,#30 ;SET UP FOR CORRECT ERROR CALL
24980
24981 ;*****
      (2) ;TEST 661 UNIQUE INTERNAL ADDRESS TEST
      (3) ;*****
      (2) TS661:
24982 131650 032737 000001 001020 BIT #1,#ENV ;ARE WE RUNNING UNDER APT
      (1) 131656 001403 BEQ 70# ;IF NO THEN DO TEST
      (1) 131660 005737 001006 TST #PASS ;IS THIS FIRST PASS
      (2) 131664 001044 BNE TS662 ;IF NO THEN SHIP TO NEXT TEST
      (1) 131666
24983 131666 012767 000340 046102 70#: MOV #340,PS ;WE WILL BE PLAYING WITH BIT6
24984 ;SO LOCK OUT EXTRANEIOUS INTERRUPTS
24985 131674 012700 131572 MOV #DADTBL,R0 ;GET LOCATION OF FIRST REGISTER ADDRESS
24986 131700 012703 131572 1#: MOV #DADTBL,R3 ;MAKE R3 POINT TO LOCATION OF FIRST
24987 ;REGISTER ADDRESS
24988 131704 012701 000005 MOV #5,R1 ;SET LOOP COUNTER TO CLEAR ALL REG.
24989 131710 005033 2#: CLR #(R3)+ ;CLEAR A REGISTER
24990 131712 077102 SOB R1,2# ;LOOP UNTIL ALL REGISTERS CLEARED
24991 131714 012770 000100 000000 MOV #BIT6,#(R0) ;SET TEST BIT IN DEVICE REGISTERS
24992 131722 012701 131572 MOV #DADTBL,R1 ;GET LOCATION OF FIRST REGISTER ADDRESS
24993 131726 012702 000005 MOV #5,R2 ;SET UP TEST LOOP COUNTER
24994 131732 032731 000100 3#: BIT #BIT6,#(R1)+ ;IS TEST BIT SET IN THIS REGISTER
24995 131736 001006 BNE 5# ;IF YES GO SEE IF THERE IS AN ERROR
24996 131740 077204 4#: SOB R2,3# ;LOOP UNTIL ALL REGISTER CHECKED
24997 131742 005030 CLR #(R0)+ ;CLEAR REGISTER JUST TESTED AND POINT
24998 ;TO NEXT ONE
24999 131744 020027 131602 CMP R0,#TBLEND ;ARE WE DONE TESTING
25000 131750 001407 BEQ 7# ;IF YES GO TO NEXT TEST
25001 131752 000752 BR 1# ;CONTINUE TESTING
25002 131754 021041 5#: CMP (R0),-(R1) ;DID WE COMPARE THE REGISTER TO ITSELF?
25003 131756 001401 BEQ 6#
      (2) 131760 104000 EMT ;WRITE TO 1 INTERNAL ADDRESS MODIFIED

```

L3

```
25004 ;ANOTHER SO ADDRESS NOT UNIQUE
25005 131762 062701 000002 64: ADD #2,R1 ;RESTORE POINTER
25006 131766 000764 BR 44 ;GET BACK IN TEST LOOP
25007 131770 005000 74: CLR R0 ;INITIALIZE LOOP COUNTER
25008 131772 005200 84: INC R0 ;INCREMENT COUNTER
25009 131774 001376 BNE 84 ; UNTIL LOOP COUNTER = 0
25010
25011
25012
25013 ;*****
(2) ;TEST 662 TEST ALL INTERNAL OPTIONS SIMULTANEOUSLY
(3) ;*****
(2) TS662:
25014 131776 032737 000001 001020 104: BIT #1, @#ENV ;ARE WE RUNNING UNDER APT
25015 132004 001405 BEQ 704 ;IF NO DO TEST
25016 132006 005737 001006 TST @#PASS ;IS THIS FIRST PASS
25017 132012 001402 BEQ 704 ;IF YES DO TEST
25018 132014 000167 001140 JMP ENDPAS ;IF NO THEN SKIP THIS TEST
25019 132020 000005 704: RESET ;CLEAR EVERY BODY
25020 132022 012767 000340 045746 MOV #340,PS ;SET PROCESSOR PRIORITY TO 7
25021 132030 017767 174700 105534 MOV @TVECT2, #TMP0
25022 132036 017767 174666 105530 MOV @RVECT2, #TMP1
25023 132044 017767 172554 105524 MOV @TVECT, #TMP2
25024 132052 017767 172542 105520 MOV @RVECT, #TMP3
25025 132060 017767 172546 105514 MOV @RTCVT, #TMP4
25026 132066 005067 000360 CLR XMTCT2
25027 132072 005067 000356 CLR RECCT2
25028 132076 005067 000354 CLR TICKS
25029 132102 012777 132220 174624 MOV @XMIT2, @TVECT2 ;SET UP SLU2 TRANSMIT VECTOR
25030 132110 012777 000340 174620 MOV #340, @TPSW2 ;AND PSW
25031 132116 012777 132254 174604 MOV @REC2, @RVECT2 ;SET UP SLU2 RECEIVER VECTOR
25032 132124 012777 000340 174600 MOV #340, @RPSW2 ;AND PSW
25033 132132 012777 132210 172472 MOV @TICKER, @RTCVT ;SET UP RTC VECTOR
25034 132140 012777 000340 172466 MOV #340, @RTCPSW ;AND PSW
25035 132146 052777 000100 174550 BIS #BIT6, @TCSR2 ;ENABLE SLU2 XMIT INTERRUPT
25036 132154 052777 000100 174536 BIS #BIT6, @RCSR2 ;ENABLE SLU2 RECEIVER INTERRUPT
25037 132162 012703 132560 MOV @BUF2, R3 ;SET UP RECEIVER BUFFER
25038 132166 052777 000100 172434 BIS #BIT6, @LKS ;ENABLE RTC INTERRUPTS
25039 132174 012701 177777 34: MOV #-1, R1 ;INITIALIZE DATA FOR SLU2
25040 132200 005067 045572 CLR PS ;DROP PROCESSOR PRIORITY TO 0
25041 132204 000001 WAITIO: WAIT ;WAIT FOR INTERRUPT
25042 132206 000776 BR WAITIO
25043
25044 132210 005267 000242 TICKER: INC TICKS ;UPDATE COUNT
25045 132214 000167 000056 JMP IOHAND ;GO TO INTERRUPT HANDLER
25046
25047 132220 005267 000226 XMIT2: INC XMTCT2 ;UPDATE XMIT INTERRUPT COUNT
25048 132224 005201 INC R1 ;UPDATE XMIT DATA
25049 132226 010177 174474 MOV R1, @TBUF2 ;SEND NEXT CHARACTER
25050 132232 026727 000214 000400 CMP XMTCT2, #400 ;IF 256 CHARACTERS HAVE NOT
25051 132240 002403 BLT 14 ;BEEN TRANSFERRED CONTINUE
25052 132242 042777 000100 174454 BIC #BIT6, @TCSR2 ;ELSE NO MORE XMIT INTERRUPTS
25053 132250 000167 000022 14: JMP IOHAND ;GO TO INTERRUPT HANDLER
25054
25055 132254 005267 000174 REC2: INC RECCT2 ;UPDATE RECEIVER INTERRUPT COUNT
25056 132260 005777 174436 TST @RBUF2 ;BIT 15 SETS IF ANY ERRORS OCCURRED
```

```

25057 132264 100002          BPL          3#          ;IF BIT IS CLEAR NO ERRORS
25058 132266 000005          RESET          ;CLEAR THE WORLD STOP ALL
25059                                     ;INTERRUPTS
25060 132270 104000          EMT          ;RECEIVER STATUS ERROR
25061 132272 117723 174424 3#:  MOVB  @RBUF2, (R3)+ ;GET DATA AND STORE IT
25062                                     ;HAS 1 SEC ELAPSED
25063 132276 026727 000154 000074 IOHAND: CMP  TICKS, #74
25064 132304 001401          BEQ          1#          ;IF YES STOP TEST
25065 132306 000002          RTI          ;RETURN FROM INTERRUPT TO AWAIT NEXT
25066 132310 042777 000100 172276 1#:  BIC  @BIT6, @TCSR ;IF YES STOP TRANSMISSIONS
25067 132316 042777 000100 174400          BIC          ;
25068 132324 042777 000100 172276          BIC  @BIT6, @LKS ;TURN OFF LINE CLOCK
25069                                     ;LOWER PRIORITY TO ALLOW TIME FOR RECEIVER TO FINISH
25070 132332 106427 000000          WAITER: MTPS  #0
25071 132336 012705 140000          MOV  @-40000,R5 ;SET UP LOOP COUNTER
25072 132342 062705 000001          1#:  ADD  @1, R5 ;DO LOOP UNTIL R5 = 0
25073 132346 001375          BNE          1#
25074 132350 000005          RESET          ;STOP EVERYONE SHOULD BE DONE
25075
25076
25077 132352 026767 000074 000074 CHECK2: CMP  XMTCT2, RECCT2 ;#OF XMIT INTERRUPTS = REC INTERRUPTS
25078 132360 001401          BEQ          1#
25079 (2) 132362 104000          EMT          ;INTERRUPT COMPARISON ERROR
25080 132364 012703 132560          1#:  MOV  @BUF2, R3 ;INITIALIZE TO FIRST RECEIVED DATA
25081 132370 005001          CLR  R1 ;INITIALIZE TO FIRST XMIT DATA
25082 132372 016704 000054          MOV  XMTCT2, R4 ;GET # OF BYTES TRANSFERRED
25083 132376 122301          2#:  CMPB (R3)+, R1 ;IS RECEIVED DATA = EXPECTED DATA
25084 (2) 132402 104000          BEQ          3#
25085 132404 005201          EMT          ;SLU2 DATA COMPARISON ERROR
25086 132406 077405          INC  R1 ;UPDATE TO NEXT GOOD DATA
25087 132410 01C777 105156 174316 FINIE: MOV  @TMP0, @TVECT2 ;LOOP UNTIL ALL DATA CHECKED
25088 132416 016777 105152 17430# MOV  @TMP1, @RVECT2 ;RESTORE VECTORS
25089 132424 016777 105146 172172          MOV  @TMP2, @TVECT
25090 132432 016777 105142 172160          MOV  @TMP3, @RVECT
25091 132440 016777 105136 172164          MOV  @TMP4, @RTCVT
25092 132446 000167 000506          JMP  ENDPAS ;FINISHED TESTING GO TO END OF PASS
25093 132452 000000          XMTCT2: .WORD 0
25094 132454 000000          RECCT2: .WORD 0
25095 132456 000000          TICKS:  .WORD 0
25096
25097 132460 004767 000744          ERROR8: JSR  PC,ABORT ;ARE WE UNDER UFD ?
25098 132464 012737 000007 001002          MOV  @7,@#FATAL ;SET UP FATAL ERROR NUMBER
25099 132472 012767 000001 046300          MOV  @1,@MSGTY ;SET FATAL ERROR FLAG
25100 132500 032737 000091 001020          BIT  @1,@#ENV ;UNDER APT ?
25101 132506 001004          BNE  COMHLT ;YES
25102 132510 012700 132522          MOV  @COMMSG,R0
25103 132514 004767 000644          JSR  PC,TYPE
25104 132520 000777          COMHLT: BR  .
25105
25106 132522 040506 046111 051125          COMMSG: .ASCIZ /FAILURE DURING COMMON TEST/<12><15>
132530 020105 052504 044522
132536 043516 041440 046517
132544 047515 020116 042524
132552 052123 006412 000

```

```

25107          132560          .EVEN
25108
25109 132560 000200          BUF2: .BLKW 200
25110
25111
25112
25113
25114 133160 005327          ENDPAS: DEC      (PC)+          ;DECREMENT TEST LOOP COUNTER
25115 133162 000001          $EOPCT: .WORD    1
25116 133164 003051          BGT      $DOAGN          ;IF COUNTER NOT 0 DO TEST AGAIN
25117 133166 005267 045614          INC      $PASS          ;INCREMENT PASS COUNTER
25118 133172 042767 100000 045606          BIC      #100000,$PASS          ;DON'T LET IT BE NEGATIVE
25119 133200 016767 045620 177754          MOV      $USMR,$EOPCT          ;RESET TEST LOOP COUNTER
25120 133206 012700 133607          MOV      #ENDMSG,R0          ;LET R0 POINT TO ENDPASS MESSAGE
25121 133212 004767 000146          JSR      PC,TYPE          ;GO TYPE END PASS MESSAGE
25122 133216 016700 044620          MOV      42,R0          ;GET MONITOR ADDRESS
25123 133222 001413          BEQ      DOAGIN          ;IF = 0 NO MONITOR SO DON'T STOP
25124 133224 000005          RESET          ;IF MONITOR CLEAR THE WORLD
25125 133226 013737 001270 000030          $ENDAD: MOV      @SAV30,@#30          ;GET BACK THE VECTOR
25126 133234 013737 001272 000032          MOV      @SAV32,@#32
25127 133242 004710          JSR      PC,(R0)          ;GO TO MONITOR
25128 133244 000240          NOP
25129 133246 000240          NOP
25130 133250 000240          NOP
25131 133252 013737 000004 037572          DOAGIN: MOV      @#4,@#TMP0
25132 133260 012737 133276 000004          MOV      @1,@#4
25133 133266 012737 000001 164000          MOV      @1,@#164000
25134 133274 000402          BR      2$
25135 133276 062706 000004          1$: ADD      #4,SP
25136 133302 013737 037572 000004          2$: MOV      @#TMP0,@#4
25137 133310 000137          $DOAGN: JMP      @PC)+          ;RETURN TO TEST AT LOCATION RESTR
25138 133312 001404          .WORD    RESTR
25139
25140
25141
25142
25143          ;*****
25144          ;*COMMON SUBROUTINES THAT ARE NEEDED BY THE PROGRAM
25145          ;*
25146          ;*****
25147 133314 012737 133330 000024          PWRDN: MOV      @PWRUP,@#24          ;SET UP POWER FAIL VECTOR FOR POWER UP
25148 133322 004567 000102          JSR      R5,ABORT          ;GO SEE IF WE ARE UNDER UFD
25149 133326 000000          HALT
25150
25151 133330 012737 133314 000024          PWRUP: MOV      @PWRDN,@#24          ;SET UP POWER FAIL VECTOR FOR POWER DOWN
25152 133336 012706 001000          MOV      @STBOT,SP          ;SET UP STACK
25153 133342 005737 000172          TST      @#MTFLAG          ;ARE WE ON MULTI-OPTION TESTER
25154 133346 001004          BNE      1$          ;IF YES SKIP TYPE OUT
25155 133350 012700 133570          MOV      @PWRMSG,R0          ;POINT R0 TO POWER FAIL MESSAGE
25156 133354 004767 000004          JSR      PC,TYPE          ;GO TYPE IT
25157 133360 000167 046020          1$: JMP      RESTR          ;GO RESTART TEST
25158
25159
25160 133364 132767 000040 045427          TYPE: BITB      #40,$ENVM          ;TYPE OUTS DISABLED
25161 133372 001015          BNE      3$          ;IF YES GET TO EXIT
25162 133374 132737 000100 000052          BITB      #100,@#52          ;UNDER UFD ?

```


CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 31 35
T662 TEST ALL INTERNAL OPTIONS SIMULTANEOUSLY

25163 133402 001011
25164 133404 105737 177564
25165 133410 100375
25166 133412 112037 177566
25167 133416 001372
25168 133420 105737 177564
25169 133424 100375
25170 133426 000207
25171
25172

18: BNE 38 ;IF YES GET TO EXIT
TSTB @TTCSR ;TEST FOR PRINTER READY BIT
BPL 18 ;IF NOT READY WAIT FOR IT
MOVB (RO),@TPB ;WHEN READY PRINT A CHARACTER
BNE 18 ;IF LAST CHARACTER NOT NULL CONTINUE TYPING
24: TSTB @TTCSR ;WAIT FOR PRINTER TO FINISH
BPL 24
34: RTS PC
;UFD ABORT MACRO

.SBTTL ABORT ROUTINE FOR LCP/ORION UFD MODE

(1)
(1)
(1)
(1) 133430 005767 045640
(1) 133434 001454
(1) 133436 020027 000032
(1) 133442 0C1443
(1) 133444 020027 000003
(1) 133450 001404
(1) 133452 005767 045620
(1) 133456 ^01443
(1)
(1) 133460 000422
(1)
(1)
(1)
(1) 133462 016767 045602 044340
(1) 133470 016767 045576 044334
(1) 133476 104043
(1) 133500 105720
(1) 133502 001376
(1) 133504 112760 000057 177777
(1) 133512 112720 000136
(1) 133516 112720 000103
(1) 133522 105010
(1) 133524 000412
(1) 133526 016767 045536 044274
(1) 133534 016767 045532 044270
(1) 133542 104042
(1) 133544 012760 177777 000042
(1) 133552 013700 000042
(1) 133556 005037 000042
(1) 133562 000167 177440
(1) 133566 000207
(1)
(1)

ABORT: TST UFDPLG ;TEST FOR USER FRIENDLY MODE
BEQ NOABRT ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
CMP RO,#32 ;IS IT A ^Z ?
BEQ ABORTZ ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
CMP RO,#3 ;IS IS A ^C ?
BEQ ABORTC ;BR TO LOAD ^C ON XXDP STACK (NO ERROR)
TST UQUIET ;TEST FOR USER-QUIET MODE
BEQ NOABRT ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
BR ABORTE ;SET DRSEERR THEN LEAVE

ABORTC: MOV SAV30,30 ;RESTORE EMT LOCATION (30)
MOV SAV32,32 ;RESTORE EMT PRIORITY LOCATION (32)
EMT +43 ;GET XXDP STACK LOC. INTO RO FROM MONITOR
18: TSTB (RO), ;FIND END OF STACK
BNE 18
MOVB #' /,-1(RO) ;LOAD SLASH OVER ZERO
MOVB #' ^,(RO) ;LOAD UPARROW
MOVB #' C,(RO) ;LOAD C
CLRB (RO) ;MAKE NEW END TO STACK
BR ABORTZ ;NOW LEAVE
ABORTE: MOV SAV30,30 ;RESTORE EMT LOCATION (30)
MOV SAV32,32 ;RESTORE EMT PRIORITY LOCATION (32)
EMT +42 ;GET DCA LOCATION INTO RO FROM MONITOR
ABORTZ: MOV #-1,42(RO) ;SET A -1 INTO LOCATION DRSEERR IN MONITOR
;AND PUT THE MONITOR RETURN ADDRESS IN RO
CLR @42 ;CLEAR MONITOR RETURN FLAG
JMP @ENDAD ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
NOABRT: RTS PC ;IF NOTUFD RETURN TO MAINLINE

25173
25174
25175
25176
25177
25178 133570 047520 042527 020122
133576 040506 046111 042105
133604 006412 000
25179 133607 105 042116 047440
133614 020106 040520 051523

; *MESSAGES
; *

PURMSG: .ASCIZ /POWER FAILED/<12><15>
ENDMSG: .ASCIZ /END OF PASS CJKL580/<12><15>

CJKLS80 LCP 5 CPU CLSTR DIAG
CJKLS8.P11 07 JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 31-36
ABORT ROUTINE FOR LCP/ORION UFD MODE

SEQ 0455

	133622	041440	045512	032514	
	133630	030102	006412	000	
25180	133635	012	051415	040524	STRMSG: .ASCIZ <12><15>/START TESTING/<12><15>
	133642	052122	052040	051505	
	133650	044524	043516	006412	
	133656	000			
25181					
25182					
25183	000001				.END

ACD22	=	000000	6424				
ACPUOP	=	000000	6424				
ACTSUM	=	054130	14904#	14917#	14921#	14946	14956
ADC1	=	014604	9385	9386	9387#		
ADC2	=	014606	9387	9388#			
ADC3	=	014626	9392	9393	9394#		
ADC4	=	014630	9394	9395#			
ADC5	=	014646	9398	9399	9400	9401#	
ADDW0	=	000000	6424				
ADDW1	=	000000	6424				
ADDW10	=	000000	6424				
ADDW11	=	000000	6424				
ADDW12	=	000000	6424				
ADDW13	=	000000	6424				
ADDW14	=	000000	6424				
ADDW15	=	000000	6424				
ADDW2	=	000000	6424				
ADDW3	=	000000	6424				
ADDW4	=	000000	6424				
ADDW5	=	000000	6424				
ADDW6	=	000000	6424				
ADDW7	=	000000	6424				
ADDW8	=	000000	6424				
ADDW9	=	000000	6424				
ADD1	=	014466	9353	9354	9355#		
ADD2	=	014470	9355	9356#			
ADD3	=	014504	9359	9360	9361#		
ADD4	=	014506	9361	9362#			
ADD5	=	014524	9365	9366	9367#		
ADD6	=	014526	9367	9368#			
ADD7	=	014540	9369	9370	9371#		
ADD8	=	014542	9371	9372#			
ADD9	=	014562	9375	9376	9377	9378#	
ADEVCT	=	000000	6424				
ADEVN	=	000000	6424				
ADONE	=	054742	15131#				
AENV	=	000000	6424				
AENVN	=	000000	6424				
AERR1	=	054732	15110	15111	15112	15127#	
AFATAL	=	000000	6424				
AMADR1	=	000000	6424				
AMADR2	=	000000	6424				
AMADR3	=	000000	6424				
AMADR4	=	000000	6424				
AMAMS1	=	000000	6424				
AMAMS2	=	000000	6424				
AMAMS3	=	000000	6424				
AMAMS4	=	000000	6424				
AMSGAD	=	000000	6424				
AMSGLG	=	000000	6424				
AMSGTY	=	000000	6424				
AMTYP1	=	000000	6424				
AMTYP2	=	000000	6424				
AMTYP3	=	000000	6424				
AMTYP4	=	000000	6424				
ANSR	=	054122	14901#				

APASS = 000000	6424				
APRIOR = 000000	6424				
APTENV = 000001	6404#	10956			
AROUN 001740	6564	6567#			
AROUND 025256	11071	11082#			
ASLB1 016164	9735	9736#			
ASLB2 016152	9731	9732	9733#		
ASLB3 016154	9733	9734#			
ASLB4 016176	9737	9738	9739	9740#	
ASL1 015556	9624	9625	9626	9627#	
ASL2 015560	9627	9628#			
ASL3 015576	9631	9632	9633	9634#	
ASL4 015600	9634	9635#			
ASL5 015614	9638	9639	9640#		
ASL6 015616	9640	9641#			
ASL7 015642	9644	9645	9646	9647	9649#
ASRB1 016222	9749	9750#			
ASRB2 016234	9751	9752	9753	9754#	
ASRB3 016236	9754	9755#			
ASRB4 016244	9756	9757#			
ASRB5 016246	9757	9758#			
ASRB6 016266	9761	9762	9763	9764#	
ASRB7 016270	9764	9765#			
ASR1 015666	9655	9656	9657	9658#	
ASR2 015670	9658	9659#			
ASR3 015712	9663	9664	9665	9666#	
ASR4 015714	9666	9667#			
ASR5 015730	9670	9671	9672	9673#	
ASR6 015732	9673	9674#			
ASR7 015762	9678	9679	9680	9681	9683#
ASTART 025746	11351#	11543	11547		
ASMREG = 000000	6424				
ATESTN = 000000	6424				
ATRAP 023652	10825	10841#			
AUNIT = 000000	6424				
AUSMR = 000000	6424				
AUTO1 023744	10864#				
AVECT1 = 000000	6424				
AVECT2 = 000000	6424				
A1 054702	15116#	15130			
A11 054702	15117#				
A12 054716	15122#				
A2 054734	15127	15129#			
BA 051760	14513#	14550#	14551#	14676#	
BBBDON 074462	18486	18493#			
BBBER1 074256	18423	18435#			
BBBP1 074442	18424	18426	18446	18467	18488#
BBBP2 074452	18444	18463	18489#		
BBB0 074212	18419#				
BBB1 074240	18428#				
BBB2 074272	18438	18441#			
BBB3 074312	18447#				
BBB4 074344	18457	18460#			
BBB5 074372	18469#	18472			
BBB6 074400	18466	18472#			
BBB7 074260	18435	18436#			

BBCDON	115122	22540	22551*						
BBCTP1	115114	22528	22536	22545*					
BBC1G	115120	22530	22537	22539	22550*				
BBC2	115070	22532*							
BBDATO	067250	17137	17155	17174	17192	17209	17226	17235*	
BBDOON	124420	24114	24123*						
BBDOOE	067400	17234	17279*						
BB01	124200	24077*							
BBPATO	067260	17165	17176	17239*					
BBPAT1	067270	17134	17152	17170	17188	17206	17223	17243*	
BBPAT2	067300	17132	17139	17247*					
BBPAT3	067310	17184	17251*						
BBPAT4	067320	17150	17255*						
BBPAT5	067330	17204	17259*						
BBPAT6	067340	17221	17228	17263*					
BBP10	067360	17157	17271*						
BBP11	067370	17211	17275*						
BBP7	067350	17194	17267*						
BB10	066726	17159*	17161						
BB13	066734	17160	17161*						
BB14	066744	17163	17165*						
BB15	066764	17171*							
BB16	067010	17178*	17180						
BB17	067016	17179	17180*						
BB2	066624	17135*							
BB20	067026	17182	17184*						
BB21	067046	17189*							
BB22	067072	17196*	17198						
BB25	067100	17197	17198*						
BB26	067110	17200	17202*						
BB27	067130	17207*							
BB3	066630	17137*							
BB30	067152	17213*	17215						
BB31	067160	17214	17215*						
BB32	067170	17217	17219*						
BB33	067210	17224*							
BB34	067232	17230*	17232						
BB35	067240	17231	17232*						
BB4	066646	17141*	17143						
BB5	066654	17142	17143*						
BB6	066664	17146	17148*						
BB7	066704	17153*							
BCF	054116	14899*	14916	14919	14940*	14950*			
BDONE	054776	15153*							
BDVHLT	054442	6527*	14987	14990*	14999				
BDVMSG	054444	14988	14992*						
BDVTST	053514	14535	14704	14773	14804*				
BDVTS2	054176	14878	14935*						
BELL	000240	6387*							
BIC1	014046	9199	9200	9201*					
BIC2	014050	9201	9202*						
BIC3	014066	9205	9206	9207*					
BIS1	014112	9213	9214	9215	9216*				
BIS2	014114	9216	9217*						
BIS3	014134	9220	9221	9222	9223*				
BITCHK	020314	10196*							

BUF2	132560	25037	25079	25109#					
B1	054752	15142#	15152						
B2	054754	15145#							
B3	054774	15151	15152#						
CARRY1	003456	6733	6734#						
CC	177776	6399#	11106#						
CCBDON	110264	21382	21385#						
CCB10	110262	21379	21381	21384#					
CCB2	110236	21374#							
CCCDON	075214	18612	18668#						
CCC1	074466	18507#							
CCC10	074752	18580#							
CCC11	074776	18588#							
CCC12	075022	18597#							
CCC13	075046	18605#							
CCC2	074512	18515#							
CCC3	074536	18523#							
CCC4	074562	18530#							
CCC5	074606	18538#							
CCC6	074632	18546#							
CCC7	074656	18555#							
CCC8	074702	18564#							
CCC9	074726	18572#							
CCDATO	066440	16971	16988	17006	17024	17041	17058	17067#	
CCERR	020612	10282#							
CCP0	066450	16966	16983	16998	17036	17053	17071#		
CCP1	066460	16985	17075#						
CCP10	066550	17026	17103#						
CCP11	066560	17043	17107#						
CCP12	066570	17111#							
CCP2	066470	16968	16973	17079#					
CCP3	066500	17016	17038	17083#					
CCP4	066510	17055	17060	17087#					
CCP5	066520	17020	17091#						
CCP6	066530	17002	17008	17095#					
CCP7	066540	16990	17099#						
CCXDON	066600	17066	17116#						
CCX12	066124	16993	16994#						
CCX13	066134	16996	16998#						
CCX14	066154	17003#							
CCX15	066200	17010#	17012						
CCX18	066206	17011	17012#						
CCX19	066216	17014	17016#						
CCX2	066014	16969#							
CCX20	066236	17021#							
CCX21	066262	17028#	17030						
CCX24	066270	17029	17030#						
CCX25	066300	17032	17034#						
CCX26	066320	17039#							
CCX27	066342	17045#	17047						
CCX3	066036	16975#	16977						
CCX30	066350	17046	17047#						
CCX31	066360	17049	17051#						
CCX32	066400	17056#							
CCX33	066422	17062#	17064						
CCX36	066430	17063	17064#						

C35	055054	15187#								
C4	055072	15192	15193#							
C45	055076	15195#								
C5	055114	15200	15201#							
C55	055120	15204#								
C6	055134	15209	15210#							
C65	055142	15212#								
C7	055160	15217	15218#							
C75	055166	15220#								
C8	055204	15225	15226#							
C85	055210	15228#								
DADTBL	131572	24965#	24985	24986	24992					
DAERR	020366	10197	10199	10201	10203	10205	10207	10209#		
DALTB1	044340	13235	13288#							
DALTB2	044374	13249	13306#							
DALTB3	044624	13334	13387#							
DALTB4	044660	13348	13405#							
DATA	051764	14515#	14554*	14555*	14672*					
DBE	025052	11032#	11035							
DBE1	025060	11027	11035#							
DBE2	025056	11029	11034#							
DBE3	025072	11036	11037#							
DBE4	025104	11038	11039#							
DBE5	025114	11040	11041#							
DOBBFO	110450	21438	21460#							
DOBDON	110460	21447	21465#							
DOBTP1	110430	21424	21451#							
DOBTP2	110440	21428	21455#							
DOB2	110364	21433#								
DOB5	110426	21443	21446	21448#						
DOCDON	115222	22567	22573#							
DOCTP1	115206	22557	22563	22570#						
DCC10	115220	22560	22564	22566	22572#					
DCC2	115154	22559	22561#							
DDOATO	070150	17296	17314	17332	17350	17367	17384	17402	17412#	
DDODON	075706	18741	18801#							
DDONE	070300	17411	17456#							
DDO1	075220	18681#								
DDO2	075260	18689#								
DDO3	075320	18697#								
DDO4	075360	18705#								
DDO5	075420	18713#								
DDO6	075460	18723#								
DDO7	075520	18732#								
DDONE	055374	15279	15300#							
DDPO	070160	17316	17334	17416#						
DDP1	070170	17291	17293	17311	17327	17420#				
DDP2	070200	17309	17329	17424#						
DDP3	070210	17345	17364	17428#						
DDP4	070220	17379	17399	17432#						
DDP5	070230	17381	17397	17436#						
DDP6	070240	17347	17362	17440#						
DDP7	070250	17352	17369	17444#						
DDP8	070260	17386	17404	17448#						
DDP9	070270	17298	17452#							
DD10	067532	17318#	17320							

DNMB3C	010132	8225	8226#						
DNMB3D	010150	8229	8230	8231#					
DNMB3E	010152	8231	8232#						
DNMB4A	010272	8259	8260	8261	8262#				
DNMB4B	010274	8262	8263#						
DNMB4C	010304	8264	8265#						
DNMB4D	010314	8266	8267#						
DNMB4E	010316	8267	8268#						
DNMB4F	010324	8269	8270#						
DNM03A	007262	7956	7957	7958	7959#				
DNM03B	007264	7959	7960#						
DNM03C	007274	7961	7963#						
DNM1	007174	7931	7932#						
DNM1A	007714	8166	8167	8168	8169#				
DNM1B	007716	8169	8170#						
DNM2	007202	7933	7934#						
DNM2A	007746	8178	8179	8180#					
DNM2B	007750	8180	8181#						
DNM2C	007756	8183#							
DNM2D	007760	8183	8184#						
DNM3	007212	7936	7937#						
DNM4	007226	7941	7942#						
DNM4A	010216	8243	8244	8245	8246#				
DNM4B	010220	8246	8247#						
DNM4C	010226	8248	8249#						
DNM5A	010366	8280	8281	8282	8283#				
DNM5B	010370	8283	8284#						
DNM5C	010400	8285	8286#						
DNM6A	010442	8295	8296	8297	8298#				
DNM6B	010444	8298	8299#						
DNM6C	010454	8300	8301#						
DNM7A	010520	8310	8311	8312	8313#				
DNM7B	010522	8313	8314#						
DNM7C	010532	8315	8316#						
DOAGIN	133252	25123	25131#						
DOPB2A	007502	8086	8087#						
DOPB2B	007534	8104	8105#						
DOP0A	007030	7877	7878#						
DOP0B	007046	7883	7884#						
DOP0C	007060	7887	7888#						
DOP0D	007102	7895	7896#						
DOP03A	007142	7913	7914	7915	7916#				
DOP03B	007144	7916	7917#						
DOP1	007402	8031	8032#						
DOP2	007452	8066	8067#						
DOP4	011514	8550#							
DOP5	011562	8574#							
DPAT3	062122	16030#							
DRLP	054376	14935*	14936	14974#					
DTRAP1	023710	10853#	10863						
DUMMY	000000	11173#	11869	11870	11871	11872	11873	11874	11875
D1	055252	15261#	15275	15281					
D2	055264	15264*	15265#						
D3	055266	15266#	15283						
D4	055272	15268#							
D5	055300	15272#	15296						

D6	055314	15273	15277#			
D7	055324	15278	15280#			
D8	055360	15290#				
D9	055372	15295	15298#			
EDONE	055442	15323	15324#			
EEBFO	110374	21498#				
EEBFF1	110604	21470	21476	21481	21487	21502#
EEBDON	110616	21491	21507#			
EEBTP1	110564	21469	21494#			
EEB10	110614	21477	21484	21488	21490	21506#
EEB2	110524	21478#				
EECDON	115330	22585	22595#			
EECTP1	115304	22576	22577#	22590#	22592	
EECTP2	115314	22575	22581	22592#		
EEC10	115326	22578	22582	22584	22594#	
EEC2	115252	22579#				
EEDATO	070454	17473	17491	17501#		
EEDONE	070536	17500	17526#			
EEEDON	076440	18918	18974#			
EEE1	075712	18815#				
EEE10	076176	18887#				
EEE11	076222	18895#				
EEE12	076246	18903#				
EEE13	076272	18911#				
EEE2	075736	18823#				
EEE3	075762	18831#				
EEE4	076006	18839#				
EEE5	076032	18847#				
EFE6	076056	18855#				
EEE7	076102	18863#				
EEE8	076126	18871#				
EEE9	076152	18879#				
EEO0	070464	17475	17493	17505#		
EEO1	070476	17468	17470	17510#		
EEO2	070506	17514#				
EEO3	070516	17486	17488	17518#		
EEO4	070526	17522#				
EERRO	055440	15310	15319	15323#		
EE12	070440	17496	17497#			
EE2	070324	17471#				
EE3	070346	17477#	17479			
EE6	070354	17478	17479#			
EE7	070370	17482	17484#			
EE8	070410	17489#				
EE9	070432	17495#	17497			
EISEND	037552	12071	12072#			
EMTA	104377	6402#	6646			
EMTSV	001202	6484#				
ENDMSG	133607	25120	25179#			
ENDPAS	133160	24974	24977	25018	25091	25114#
ENT176	031434	11856	11864#			
ENT51	027360	11602	11606#			
ER	001762	6563	6565	6577#		
ERRORA	051550	14374	14485#			
ERRORB	053402	14537	14778#			
ERRJRC	054402	14815	14983#			

CJKLS80 LCP 5 CPU CLSTR DIAG
CJKLS8.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 32-16
CROSS REFERENCE TABLE - USER SYMBOLS

SEQ 0472

G34	057736	156560						
G35	057770	156560						
G36	057772	156560						
G37	060036	156570						
G4	056736	156480						
G40	060070	156570						
G41	060072	156570						
G42	060136	156590						
G43	060170	156580						
G44	060172	156580						
G5	05677J	156480						
G6	056772	156480						
G7	057036	156490						
HADR	061112	157830	15790	157990				
HA1R	061166	15775	15785	15793	158080			
HA1W	061116	15741	15750	15760	15784	158020		
HA2R	061176	15776	158090					
HA2W	061126	15751	15761	158030				
HA3R	061206	15777	158100					
HA3W	061136	15752	15762	158040				
HA4R	061216	15778	158110					
HA4W	061146	15753	15763	158050				
HASR	061226	15779	158120					
HASW	061156	15754	15764	158060				
HCLR	061074	15747	15774	157930				
HCLR1	061104	157950	15796					
HCMP	061040	15758	15760	15761	15762	15763	15764	157830
HCMP1	061060	157870	15789					
HCMP2	061066	15788	157890					
HDAT1	061236	15742	158140					
HDAT2	061246	158150						
HDAT3	061256	158160						
HDAT4	061266	158170						
HDAT5	061276	158180						
HDOE	061306	15768	158200					
HERE	000000	105060	105250	105300	10569			
HFLAG	061114	157400	15766	157700	158000			
HBBF0	111226	21593	21605	21620	216210			
HBBF1	111236	21599	21610	216250				
HBDON	111250	21614	216300					
HHTP1	111206	21592	216170					
HMB10	111246	21600	21608	21611	21613	216290		
HMB2	111146	216020						
HMC DON	115634	22654	226620					
HMC TP1	115612	22642	226430	226590				
HMC TP2	115622	22641	22650	226600				
HMC10	115632	22645	22649	22651	22653	226610		
HMC2	115560	226460						
HMDAT0	071366	17633	17655	17676	17694	17712	177220	
HDOE	071536	17721	177740					
HBDON	102772	19977	200550					
HH1	101562	198490						
HH10	102332	199370						
HH11	102402	199480						
HH12	102452	199590						
HH13	102522	199700						

HH2	101632	19858#							
HH3	101702	19868#							
HH4	101752	19878#							
HH5	102022	19888#							
HH6	102072	19899#							
HH7	102142	19908#							
HH8	102212	19917#							
HH9	102262	19926#							
HHP0	071376	17628	17726#						
HHP1	071406	17630	17730#						
HHP10	071516	17696	17766#						
HHP11	071526	17714	17770#						
HHP2	071416	17635	17734#						
HHP3	071426	17738#							
HHP4	071436	17657	17742#						
HHP5	071446	17650	17707	17709		17746#			
HHP6	071456	17652	17750#						
HHP7	071466	17678	17754#						
HHP8	071476	17671	17689	17691		17758#			
HHP9	071506	17673	17762#						
HH10	071116	17660	17661#						
HH11	071132	17664	17668#						
HH12	071152	17674#							
HH13	071174	17680#	17682						
HH16	071202	17681	17682#						
HH17	071216	17685	17687#						
HH18	071236	17692#							
HH19	071260	17698#	17700						
HH2	071006	17631#							
HH20	071266	17699	17700#						
HH21	071302	17703	17705#						
HH22	071322	17710#							
HH23	071344	17716#	17718						
HH24	071352	17717	17718#						
HH3	071030	17637#	17639						
HH6	071036	17638	17639#						
HH7	071046	17641	17647#						
HH8	071066	17653#							
HH9	071110	17659#	17661						
HICORE	023604	10829#	10844						
HLT	000000	6375#							
HSTD	060762	15756	15760	15761	15762	15763	15764	15774#	
HXDAT0	073272	18005	18030	18060	18076	18099	18119	18131#	
HXDONE	073402	18129	18171#						
HXER9	072776	18045	18052#						
HXP1	073302	17997	18019	18090	18094	18102	18116	18121	18136#
HXP2	073312	17999	18001	18022	18025	18057	18141#		
HXP3	073322	18145#							
HXP4	073332	18073	18149#						
HXP5	073342	18079	18153#						
HXP6	073352	18055	18071	18114		18157#			
HXP7	073362	18007	18062	18162#					
HXP8	073372	18033	18166#						
HX10	072674	18026	18028#						
HX11	072716	18035#	18037						
HX14	072724	19036	18037#						

HX15	072740	18041	18043#						
HX16	072756	18047#							
HX165	072762	18048#							
HX17	073000	19052	18053#						
HX18	073020	18058#							
HX19	073040	18064#	18066						
HX2	072564	18000#							
HX20	073046	18065	18066#						
HX22	073070	18074#							
HX23	073110	18081#	18083						
HX26	073116	18082	18083#						
HX28	073142	18095#							
HX29	073164	18104#	18106						
HX3	072576	18002	18003#						
HX30	073172	18105	18106#						
HX31	073206	18110	18112#						
HX32	073226	18117#							
HX33	073250	18123#	18125						
HX34	073256	18124	18125#						
HX4	072616	18009#	18011						
HX7	072624	18010	18011#						
HX8	072640	18014	18016#						
HX9	072662	18024#							
H1	060424	15740#							
H10	060666	15763#							
H11	060720	15764#							
H12	060752	15767	15770#						
H2	060444	15744#	15745						
H3	060454	15749#	15771						
H4	060526	15756#							
H5	060550	15760#							
H6	060602	15761#							
H7	060634	15762#							
IDATIO	056274	15443	15449	15464	15473	15478	15503	15525#	
IDATI1	056276	15526#							
IDATI2	056300	15500#	15527#						
IDATI3	056302	15501#	15528#						
IDAT00	056264	15460	15463	15497	15502	15520#			
IDAT01	056266	15521#							
IDAT02	056270	15522#							
IDAT03	056272	15523#							
IDONE	056304	15508	15530#						
IIBBF0	111374	21634	21646	21661	21662#				
IIBBF1	111404	21640	21651	21666#					
IIBDON	111416	21655	21672#						
IIBTP1	111354	21633	21658#						
IIB10	111414	21641	21649	21652	21654	21671#			
IIB2	111314	21643#							
IICDON	115700	22686	22688#						
IIC2	115656	22679#							
IIC20	115676	22674	22686#						
IIIDON	077274	19131	19208#						
III1	076776	19095#							
III2	077026	19105#							
III3	077056	19114#							
III4	077106	19123#							

IJMP	013264	9025	9028#				
IJMP4	013102	8992	8995#	9002			
IJMP5	013234	9018	9021#				
ILLA	- 004700	6392#	10691				
ILLB	- 000100	6393#	10694				
INC1	014156	9241	9242	9243#			
INC2	014160	9243	9244#				
INC3	014202	9248	9249	9250	9251#		
INC4	014204	9251	9253#				
INC5	014220	9256	9257	9258#			
INST	025612	11102*	11103*	11107	11115	11120	11165#
IOMAND	132276	25045	25053	25063#			
IPAT10	056244	15444	15472	15510#			
IPAT11	056246	15511#					
IPAT12	056250	15512#					
IPAT13	056252	15513#					
IPAT20	056254	15452	15481	15515#			
IPAT21	056256	15516#					
IPAT22	056260	15517#					
IPAT23	056262	15518#					
ITRAP5-	000004	6377#					
I1	056006	15441#					
I105	056106	15467	15468#				
I12	056110	15472#					
I13	056124	15475#	15476				
I14	056152	15487#					
I15	056154	15488#					
I16	056156	15489#					
I17	056172	15493	15494#				
I2	056024	15446#	15447				
I20	056232	15505#	15507				
I23	056240	15506	15507#				
I3	056050	15456#					
I4	056052	15457#					
I5	056054	15458#					
I6	056100	15466#	15468				
JBUF0	061720	15941	15954#				
JBUF1	061722	15955#					
JBUF2	061724	15956#					
JBUF3	061726	15957#					
JDAT10	061730	15931	15943	15950	15959#		
JDAT11	061732	15960#					
JDAT12	061734	15961#					
JDAT13	061736	15962#					
JDAT00	061740	15938	15944	15964#			
JDAT0	061750	15928	15969#				
JDAT01	061742	15965#					
JDAT02	061744	15966#					
JDAT03	061746	15967#					
JDAT1	061752	15970#					
JDAT2	061754	15971#					
JDAT3	061756	15972#					
JDONE	061760	15952	15975#				
JJBBF0	111536	21676	21682	21688	21693	21704#	
JJBDON	111550	21697	21709#				
JJBTP1	111524	21675	21700#				

LLBDON	112036	21771	21784#		
LLBTP1	112014	21752	21774#		
LLB10	112034	21759	21767	21770	21783#
LLB2	111760	21761#			
LLCDON	117552	23051	23110#		
LLC1	117002	22971#			
LLC10	117324	23027#			
LLC11	117362	23036#			
LLC12	117420	23044#			
LLC2	117040	22979#			
LLC3	117076	22987#			
LLC4	117134	22995#			
LLC5	117172	23003#			
LLC6	117230	23011#			
LLC7	117266	23019#			
LLL00N	100622	19536	19617#		
LLL1	100214	19492#			
LLL2	100260	19503#			
LLL3	100324	19515#			
LLL4	100370	19526#			
LPAT10	062256	16045	16081#		
LPAT11	062260	16082#			
LPAT12	062262	16083#			
LPAT13	062264	16084#			
LPAT20	062266	16049	16086#		
LPAT21	062270	16087#			
LPAT22	062272	16088#			
LPAT23	062274	16089#			
LSREG -	177524	6396#	6514*	14805*	14811
LTCHLT	125664	6523*	24511	24514#	
LTCMSG	126666	24512	24515#		
L1	062130	16042#			
L2	062170	16059#			
L3	062172	16060#			
L4	062174	16062#			
L5	062244	16076#	16078		
L6	062252	16077	16078#		
MBDM2A	010654	8367	8368	8369#	
MBDM2B	010656	8369	8370#		
MBDM2C	010666	8371	8372#		
MBDM2D	010700	8373	8374	8375#	
MBDM2E	010702	8375	8376#		
MBDM2F	010712	8377	8378#		
MBDM4A	011122	8439	8440#		
MBDM4B	011132	8441	8442#		
MBDM4C	011144	8443	8444	8445#	
MBDM4D	011146	8445	8446#		
MBDM4E	011154	8447	8448#		
MDAT00	062450	16130	16133	16152#	
MDAT01	062452	16153#			
MDAT02	062454	16154#			
MDAT03	062456	16155#			
MDM1A	010564	8331	8332	8333#	
MDM1B	010566	8333	8334#		
MDM2A	010614	8349	8350	8351#	
MDM2B	010616	8351	8352#		

MDM2C	010624	8354*						
MDM2D	010626	8354	8355*					
MDM3A	010750	8391	8392	8393*				
MDM3B	010752	8393	8394*					
MDM3C	010762	8395	8396*					
MDM3D	010774	8397	8398*					
MDM3E	011014	8400	8401*					
MDM4A	011056	8417	8418	8419*				
MDM4B	011060	8419	8420*					
MDM4C	011066	8421	8422*					
MDM5A	011206	8465	8466	8467*				
MDM5B	011210	8467	8468*					
MDM5C	011220	8469	8470*					
MDM5D	011230	8471	8472*					
MDM5E	011250	8475	8476*					
MDM6A	011302	8492	8493	8494*				
MDM6B	011304	8494	8495*					
MDM6C	011314	8496	8497*					
MDM6D	011326	8498	8499*					
MDM6E	011350	8502	8503*					
MDM7A	011406	8518	8519	8520*				
MDM7B	011410	8520	8521*					
MDM7C	011420	8522	8523*					
MDM7D	011432	8524	8525*					
MDM7E	011450	8527	8528*					
MOONE	062460	16140	16157*					
MEM	050750	14362*	14462*	14464	14465*	14477*	14541*	14562
MERR3	062366	16120	16129*					
MFPIO	020140	10148	10151*					
MFPI0A	020160	10153	10155*					
MFPS1	017230	9984	9986*					
MFPS2A	017302	9998	9999	10000	10001*			
MFPS2B	017304	10001	10002*					
MFPS2C	017316	10003	10004*					
MFPS3A	017356	10012	10013	10014	10015*			
MFPS3B	017360	10015	10016*					
MFPS3C	017372	10017	10018*					
MFPS4A	017432	10026	10027	10028	10029*			
MFPS4B	017434	10029	10030*					
MFPS4C	017446	10031	10032*					
MFPS5A	017506	10040	10041	10042	10043*			
MFPS5B	017510	10043	10044*					
MFPS5C	017522	10045	10046*					
MFPS6A	017564	10054	10055	10056	10057*			
MFPS6B	017566	10057	10058*					
MFPS6C	017600	10059	10060*					
MFPS7A	017642	10068	10069	10070	10071*			
MFPS7B	017644	10071	10072*					
MFPS7C	017656	10073	10074*					
MFPT	000007	10410*	10412					
MFDF0	112152	21788	21799	21812	21813*			
MFDF1	112162	21796*	21817*					
MFDDON	112174	21806	21823*					
MFSTP1	112132	21787	21809*					
MFBI0	112172	21794	21802	21805	21822*			
MFBI2	112076	21796*						

NDONE	062634	16194	162170					
NEGAT	026760	11382	11413	11444	11475	11508	11540	115440
NEG00	005040	7198	7199	7200	72010			
NEG01	005042	7201	72030					
NEG02	005050	7204	72060					
NEG03	005064	7208	7209	7210	72110			
NEG04	005066	7211	72120					
NEG1	014672	9418	9419	9420	94210			
NEG10	005114	7219	7220	7221	72220			
NEG11	005116	7222	72240					
NEG12	005126	7225	72260					
NEG13	005142	7228	7229	7230	72310			
NEG14	005144	7231	72320					
NEG2	014674	9421	94220					
NEG20	005174	7239	7240	7241	72420			
NEG21	005176	7242	72430					
NEG22	005216	7249	72500					
NEG3	014716	9426	9427	9428	94290			
NEG30	005426	7354	7355	7356	73570			
NEG31	005430	7357	73580					
NEG32	005436	7359	73600					
NEG33	005454	7363	73640					
NEG34	005462	7365	73660					
NEGA	014720	9429	94300					
NEGA0	005716	7472	7473	7474	74750			
NEGA1	005720	7475	74760					
NEGA2	005726	7477	74780					
NEG5	014740	9434	9435	9436	94370			
NEG50	005764	7489	7490	7491	74920			
NEG51	005766	7492	74930					
NEG52	005774	7494	74950					
NEG60	006034	7506	7507	7508	75090			
NEG61	006036	7509	75100					
NEG70	006070	7518	7519	7520	75210			
NEG71	006072	7521	75220					
NERRO	062554	16176	161920					
NEXT	051174	14378	144130					
NNBFF0	112306	21841	21843	218620				
NNBDON	112320	21851	218680					
NNBTP1	112266	21834	218540					
NNBTP2	112276	21844	218580					
NNB10	112316	21847	21850	218670				
NNB2	112214	218360						
NNCDON	121170	23391	233980					
NNCTB0	121152	23375	233940					
NNCTB1	121156	23382	23385	23387	23389	233950		
NNC10	121166	23380	23386	23388	23390	233960		
NNC2	121120	233840						
NNNDON	104136	20263	203490					
NNN1	103436	202200						
NNN2	103512	202310						
NNN3	103566	202420						
NNN4	103642	202530						
NOABRT	133566	251720						
NODL	024000	10871	108740					
NODL2	024426	109360						

NOP	000240	6360	6388#			
NOR	023600	10827#	10830			
NOSUB	023674	10846	10849#			
NPAT10	062614	16171	16207#			
NPAT11	062616	16208#				
NPAT12	062620	16209#				
NPAT13	062622	16210#				
NPAT20	062602	16174	16186	16202#		
NPAT21	062604	16203#				
NPAT22	062606	16204#				
NPAT23	062610	16205#				
NXTST	024704	10955	10987#			
N1	062464	16168#				
N12	062534	16187	16188#			
N13	062550	16191#	16193			
N14	062556	16192	16193#			
N2	062510	16179#				
N3	062512	16180#				
N4	062514	16181#				
ODATIO	063000	16250	16263	16274#		
ODATI1	063002	16275#				
ODATI2	063004	16276#				
ODATI3	063006	16277#				
ODAT00	062736	16244	16249	16257#		
UDAT01	062740	16258#				
ODAT02	062742	16259#				
ODAT03	062744	16260#				
ODONE	063010	16255	16279#			
OERR0	062706	16236	16248#			
O0BDON	112450	21893	21906#			
O0BTP1	112426	21871	21876	21883	21889	21896#
O0BTP2	112436	21872	21884	21900#		
O0B10	112446	21887	21890	21892	21905#	
O0B2	112362	21881#				
O0CDON	121302	23420	23427#			
O0CTB0	121264	23404	23423#			
O0CTB1	121270	23411	23414	23416	23418	23424#
O0C10	121300	23409	23415	23417	23419	23425#
O0C2	121232	23413#				
O0ODON	105130	20522	20524#			
O00T	105064	20498	20509#			
O002	105062	20505#	20510			
O003	105076	20513#				
O004	105122	20520	20521#			
OPAT10	062770	16231	16269#			
OPAT11	062772	16270#				
OPAT12	062774	16271#				
OPAT13	062776	16272#				
OPAT20	062754	16247	16263#			
OPAT21	062756	16234	16264#			
OPAT22	062760	16265#				
OPAT23	062762	16266#				
OPAT24	062764	16267#				
OVDNTT	077646	19325	19327	19339#		
OVDTT	100612	19600	19602	19615#		
OVFNTT	077264	19192	19194	19206#		

SHR	003564	6758#	6761			
SHRE	003600	6759	6762#			
SIMG0A	051772	14518#	14560#	14561#	14679#	
SKTST2	024514	10950#				
SLU1ST	124636	10875	10886	15093	24127	24188#
SLU2ST	126740	24383	24503	24529#		
SL1MLT	125740	6524#	24371	24374#		
SL1MSG	125742	24372	24376#			
SL2MLT	131534	6525#	24956	24959#		
SL2MSG	131536	24957	24961#			
SNP0A	006206	7579	7580	7581#		
SNP1A	006256	7616	7617	7618#		
SNP1B	006260	7618	7619#			
SNP1C	006302	7624	7625	7626	7627#	
SNP2A	006362	7665	7666	7667#		
SNP2B	006364	7667	7668#			
SNP2C	006372	7669	7670#			
SNP2D	006412	7674	7675	7676	7677#	
SNP2E	006414	7677	7678#			
SNP3A	006514	7724	7725	7726#		
SNP3B	006516	7726	7727#			
SNP3C	006534	7730	7731	7732	7733#	
SNP3D	006536	7733	7734#			
SNP10A	006164	7560	7561	7562	7563#	
SNP11A	006232	7597	7598	7599	7600#	
SNP12A	006326	7642	7643	7644	7645#	
SNP12B	006330	7645	7646#			
SNP13A	006452	7697	7698	7699	7700#	
SNP13B	006454	7700	7701#			
SNP14A	006570	7752	7753	7754#		
SNP14B	006572	7754	7755#			
SNP15A	006626	7774	7775	7776#		
SNP15B	006630	7776	7777#			
SNP16A	006666	7796	7797	7798#		
SNP16B	006670	7798	7799#			
SNP17A	006724	7818	7819	7820#		
SNP17B	006726	7820	7821#			
S0B1	016460	9841#	9846			
S0B2	016466	9841	9842	9843#		
S0B3	016470	9843	9844#			
S0B4	016510	9847	9848	9849	9851#	
S0PA	006120	7539	7540#			
S0PB	006140	7542	7544#			
S0PB0A	004514	7039	7040#			
S0PB0B	004524	7041	7043#			
S0PB1A	004566	7075	7076#			
S0PB1B	004604	7077	7080	7082#		
S0PB1C	004624	7100	7101#			
S0PB1D	004644	7106	7108#			
S0PB2A	004730	7154	7155#			
S0PB2B	004750	7159	7162#			
S0PB2C	004774	7177	7178#			
S0PB2D	005020	7184	7187#			
S0PB3A	005304	7301	7302#			
S0PB3B	005330	7305	7307	7311#		
S0PB3C	005352	7334	7335#			

SOP83D	005374	7342	7344#																
SOPX	006112	7538#	7540#	7541*	7545														
SOPXAD	006142	7543*	7545#																
SOPZA	004666	7127	7128#																
SOP0A	004430	6998	6999#																
SOP0B	004442	7002	7003#																
SOP0C	004460	7019	7020#																
SOP0D	004504	7022	7025	7029#															
SOP1A	004536	7056	7057#																
SOP1B	004550	7058	7061#																
SOP2B	004706	7131	7135#																
SOP3A	005244	7273	7274#																
SOP3B	005260	7277	7279#																
SOP4A	005514	7384	7385#																
SOP4B	005534	7388	7392#																
SOP5A	005566	7419	7420#																
SOP5B	005602	7423	7425#																
SOP6A	005622	7440	7441#																
SOP6B	005636	7442	7444#																
SOP7A	005660	7461	7462#																
SOP7B	005674	7463	7465#																
SPAT10	061574	15832	15852	15861	15876	15906#													
SPAT11	061576	15907#																	
SPAT12	061600	15908#																	
SPAT13	061602	15909#																	
SRO	177572	12081#	12281*	12405	12449	12498*	12500	12505*	12847*	12848*	12849*	12858*	12860*	12879*					
		12881*	12909*	12910*	12911*	12912*	12958*	13000*	13035	13036*	13108	13121*	13168	13181*					
		13260	13274*	13277*	13359	13373*	13376*	13455*	13503	13508	13523*	13524	13537*	13636					
		13650*	13686	13688*	14188*	14385*	14393*	14401*	14413*	14423*	14426*	14540*	14750*	14761*					
		14771*																	
SR1	177574	12081#	12517																
SR2	177576	12081#	12457	12475	13109	13169	13261	13360	13449	13485	13492	13504	13509	13528					
		13538	13637	13687															
SR3	172516	12081#	12522*	12523	12529*	12530	12847*	12848*	12849*	12857*	12878*	12909*	12910*	12911*					
		12912*	12957*	14386*	14394*	14402*	14424*	14746*	14762*	14772*									
SSB00N	113244	22053	22067#																
SSBTP1	113220	22029	22035	22036*	22043	22056#	22064												
SSBTP2	113230	22030	22044	22060#															
SSBTP3	113240	22034	22049	22064#															
SSB10	113242	22047	22050	22052	22066#														
SSB2	113154	22040#																	
SSCDON	122006	23541	23547#																
SSCTB0	121770	23522	23544#																
SSCTB1	121774	23530	23535	23537	23539	23545#													
SSC10	122004	23527	23534	23536	23538	23540	23546#												
SSC2	121726	23532#																	
SSSA1	106114	20732	20745	20757#															
SSSBF0	106104	20723	20736	20738	20753#														
SSSDON	106136	20750	20769#																
SSSTP1	106124	20729	20739	20763#															
SSS10	106134	20731	20737	20742	20748	20768#													
SSS2	106024	20734#																	
START	001202	6475	6484#																
STATUS	177776	6389#	6626*	6627*	6645*	6646*	6655*	6662*	6663*	10835*	10836	10860*	10861	10879*					
		10885*	10895*	10902*	10911*	10947*	10973*	10976											
STBOT	001000	6400	6401	6467#	6476	6502	6509	6626	6627	6645	6646	6655	6662	6663					

S6	025656	111960	12008	12011										
S7	025660	111970												
S8	025662	111980												
S9	025664	111990	11200	12048	12051	12053								
TABLE	025470	11091	111240											
TBIT	000020	120850	14202	14207	14220									
TBITPS	037570	122720	122800	142050	14220	14222	142230							
TBLENC	131602	249690	24999											
TBL1	011526	8544	85450	85460	85470	85480	8549	85560	85690	85700	85710	85720	8573	8575
		8576	8577	8578	8579	8591	85920	85930	85940	85950	8596	86100	86110	86120
		86130	8614											
TBL2	011574	8568	85790	8609										
TBUF	124616	241770	24208	242190	242240	242360	242390	243520	243550					
TBUF2	126726	245230	24549	245560	245610	245740	245770	247370	247400	247540	247750	247900	248080	248290
		248520	248700	248840	249010	249200	249330	250490						
TCCBF0	115756	22701	22709	227170										
TCCBF1	115766	22705	227180											
TCCDON	115776	22713	227200											
TCC2	115716	227020												
TCC3	115746	22710	227120											
TCSR	124614	241760	24199	24220	24225	24228	24237	24242	242870	24290	242950	243000	243060	24310
		243120	243150	243240	24330	243320	243410	243470	243510	24353	243580	250660		
TCSR2	126724	245220	24540	24557	24562	24565	24575	24580	24606	246080	24609	246110	24612	246150
		24617	24625	246270	24628	246300	24631	246380	24640	246710	24674	246790	246840	246900
		24694	246960	246990	247080	24714	247160	247250	247320	247360	24738	247430	247850	24885
		249050	249190	249230	250350	250520	250670							
TDAT0	056600	15541	15549	15564	15573	15578	15603	156260						
TDATI1	056602	156270												
TDATI2	056604	156000	156280											
TDATI3	056606	156010	156290											
TDAT00	056570	15560	15563	15597	15602	156210								
TDAT01	056572	156220												
TDAT02	056574	156230												
TDAT03	056576	156240												
TDEC1	022414	10639	106420											
TDEC2	022430	10646	106480											
TDEC3	022460	10654	106560											
TDEC4	022512	10660	106630											
TDEC6	022526	106660												
TDEC7	022530	10664	106670											
TDEC77	024074	10881	108890											
TDEC8	024072	10883	108880											
TDONE	056610	15608	156310											
TEMP1	025630	108820	10892	108960	10907	109140	10925	109630	10980	111810	11182	113270	11351	11384
		11415	11446	11477	11511	115440	115500	115650	115720	115900	116030	116400	11648	11687
		11726	117670	117750	117890	117970	118040	118090	118160	118500	118570			
TEMP2	025632	111830	11184	113280	11354	11358	113780	11387	11390	114090	11418	11421	114400	11449
		11452	114710	11481	11484	115040	11514	11517	115370	115450	115510	115570	115610	115660
		115730	115780	115840	115910	115970	116410	11649	11688	11727	117680	118170	118270	118380
		118430	118510	118580										
TEMP3	025634	111850	11186	113290	11369	113790	11401	114100	11432	114410	11463	114720	11495	115050
		11528	115380	115460	115520	115670	115740	115790	115850	115920	115980	116420	11653	11656
		116790	11692	11695	117170	11731	11734	117570	117690	117810	117850	117900	118100	118190
		118320	118390	118440	118590									
TEMP4	025636	111870	11188	113300	11366	11398	11429	11460	11492	11525	115530	115680	115800	115860
		115930	115990	116040	116430	11667	116820	11706	117200	11745	117600	117700	117760	117910

TR3	024316	10913	10920#		
TR4	024322	10915	10922#		
TR5	024320	10917	10921#		
TSTSPC	037520	12065#			
TST160	030712	11686	11724	11764	11773#
TST161	030740	11774	11779#		
TST162	030756	11780	11783#		
TST163	030774	11784	11787#		
TST164	031036	11788	11795#		
TST165	031074	11796	11802#		
TST166	031120	11803	11807#		
TST167	031150	11808	11813#		
TST170	031206	11814	11823#		
TST171	031232	11825	11830#		
TST172	031260	11831	11836#		
TST173	031302	11837	11841#		
TST174	031340	11842	11848#		
TST175	031374	11849	11855#		
TST176	031446	11869#			
TST177	031514	11870#			
TST200	031562	11871#			
TST201	031634	11872#			
TST202	031702	11873#			
TST203	031750	11874#			
TST204	032022	11875#			
TST205	032070	11876#			
TST206	032162	11882#			
TST207	032240	11883#			
TST210	032316	11884#			
TST211	032374	11885#			
TST212	032450	11886#			
TST213	032524	11887#			
TST214	032600	11888#			
TST215	032656	11889#			
TST216	032734	11890#			
TST217	033010	11891#			
TST220	033064	11988#			
TST221	033134	11989#			
TST222	033204	11990#			
TST223	033254	11991#			
TST224	033330	11992#			
TST225	033400	11993#			
TST226	033450	11994#			
TST227	033524	11995#			
TST230	033574	11996#			
TST231	033644	11997#			
TST232	033714	11998#			
TST233	033770	11999#			
TST234	034040	12000#			
TST235	034110	12001#			
TST236	034160	12002#			
TST237	034234	12003#			
TST240	034304	12004#			
TST241	034354	12005#			
TST242	034444	12010#			
TST243	034514	12011#			

TST244	034564	12012#					
TST245	034634	12013#					
TST246	034702	12014#					
TST247	034750	12015#					
TST250	035016	12016#					
TST251	035066	12017#					
TST252	035136	12018#					
TST253	035204	12019#					
TST254	035252	12030#					
TST255	035326	12031#					
TST256	035402	12032#					
TST257	035462	12033#					
TST260	035536	12034#					
TST261	035612	12035#					
TST262	035672	12036#					
TST263	035746	12037#					
TST264	036022	12038#					
TST265	036076	12039#					
TST266	036156	12040#					
TST267	036232	12041#					
TST270	036306	12042#					
TST271	036366	12043#					
TST272	036442	12044#					
TST273	036516	12045#					
TST274	036564	12046#					
TST275	036642	12051#					
TST276	036716	12052#					
TST277	036772	12053#					
TST300	037046	12054#					
TST301	037120	12055#					
TST302	037172	12056#					
TST303	037244	12057#					
TST304	037320	12058#					
TST305	037374	12059#					
TST306	037446	12060#					
TST37	027006	11383	11414	11445	11476	11509	11542 11548#
TST40	027042	11549	11555#				
TST41	027056	11556	11559#				
TST42	027074	11560	11563#				
TST43	027130	11564	11570#				
TST44	027162	11571	11576#				
TST45	027214	11577	11582#				
TST46	027246	11583	11588#				
TST47	027304	11589	11595#				
TST50	027334	11596	11601#				
TST51	027406	11613#					
TST52	027446	11614#					
TST53	027506	11615#					
TST54	027546	11616#					
TST55	027604	11617#					
TST56	027642	11618#					
TST57	027700	11619#					
TST60	027740	11620#					
TST61	030000	11621#					
TST62	030036	11622#					
TS1	001616	6517	6551#				

TS10	003116	6663#	7573#
TS100	006166	7563	7591#
TS101	006210	7581	7609#
TS102	006234	7600	7636#
TS103	006304	7627	7658#
TS104	006340	7648	7689#
TS105	006424	7680	7714#
TS106	006464	7703	7745#
TS107	006546	7736	
TS11	003312	6685#	
TS110	006600	7756	7765#
TS111	006640	7779	7788#
TS112	006676	7800	7810#
TS113	006734	7822	7830#
TS114	006752	7836	7842#
TS115	006770	7848	7854#
TS116	007020	7864	7874#
TS117	007114	7899	7906#
TS12	003330	6689	6691#
TS120	007160	7919	7926#
TS121	007236	7944	7950#
TS122	007276	7963	7971#
TS123	007316	7978	7986#
TS124	007336	7993	8002#
TS125	007356	8009	8022#
TS126	007412	8034	8043#
TS127	007434	8051	8060#
TS13	003350	6695	6697#
TS130	007462	8069	8080#
TS131	007512	8088	8096#
TS132	007544	8107	8116#
TS133	007570	8121	8129#
TS134	007614	8134	8142#
TS135	007640	8147	8149#
TS136	007672	8159	8161#
TS137	007724	8171	8173#
TS14	003370	6701	6703#
TS140	007770	8185	8187#
TS141	010064	8208	8211#
TS142	010164	8234	8236#
TS143	010240	8250	8252#
TS144	010334	8271	8273#
TS145	010412	8287	8289#
TS146	010466	8302	8304#
TS147	010544	8317	8325#
TS15	003410	6706	6722#
TS150	010574	8335	8343#
TS151	010636	8356	8363#
TS152	010724	8379	8386#
TS153	011034	8403	8412#
TS154	011076	8423	8433#
TS155	011164	8449	8460#
TS156	011260	8477	8487#
TS157	011362	8504	8513#
TS16	003436	6727	6729#
TS160	011462	8529	8543#

TS161	011530	8550	8567#
TS162	011576	8574	8590#
TS163	011644	8597	8608#
TS164	011712	8615	8624#
TS165	011762	8638	8649#
TS166	012072	8673	8682#
TS167	012216	8712	8721#
TS17	003470	6735	6737#
TS170	012326	8739	8749#
TS171	012364	8758	8770#
TS172	012432	8780	8790#
TS173	012462	8796	8806#
TS174	012522	8813	8824#
TS175	012544	8829	8838#
TS176	012570	8843	8853#
TS177	012624	8860	8870#
TS2	002040	6626#	
TS20	003522	6745	6747#
TS200	012650	8874	8884#
TS201	012704	8891	8903#
TS202	012746	8910	8924#
TS203	013000	8929	8943#
TS204	013034	8948	8984#
TS205	013302	9031	9050#
TS206	013670	9131	9141#
TS207	013720	9150	9165#
TS21	003550	6752	6754#
TS210	013760	9178	9179#
TS211	014024	9193	9194#
TS212	014070	9207	9208#
TS213	014136	9223	9236#
TS214	014222	9258	9260#
TS215	014324	9287	9299#
TS216	014344	9306	9308#
TS217	014404	9322	9323#
TS22	003602	6762	6784#
TS220	014446	9337	9349#
TS221	014564	9378	9380#
TS222	014650	9401	9413#
TS223	014742	9437	9439#
TS224	015062	9467	9469#
TS225	015104	9476	9489#
TS226	015216	9517	9519#
TS227	015324	9548	9560#
TS23	003614	6788	6790#
TS230	015432	9589	9590#
TS231	015534	9618	9619#
TS232	015644	9649	9650#
TS233	015764	9683	9686#
TS234	016132	9723	9726#
TS235	016200	9740	9743#
TS236	016300	9766	9776#
TS237	016362	9798	9808#
TS24	003630	6793	6795#
TS240	016450	9829	9837#
TS241	016512	9851	9859#

TS242	016634	9885	9896#
TS243	016700	9909	9911#
TS244	016736	9919	9921#
TS245	017002	9930	9932#
TS246	017042	9940	9942#
TS247	017100	9950	9952#
TS25	003644	6798	6800#
TS250	017144	9960	9962#
TS251	017210	9970	9980#
TS252	017256	9991	9993#
TS253	017326	10005	10007#
TS254	017402	10019	10021#
TS255	017456	10033	10035#
TS256	017532	10047	10049#
TS257	017610	10061	10063#
TS26	003660	6803	6805#
TS260	017666	10075	10085#
TS261	017726	10086	10090 10098#
TS262	017766	10117#	
TS263	020100	10142#	
TS264	020160	10157#	
TS265	020242	10170	10181#
TS266	020374	10218#	
TS267	020432	10234#	
TS27	003706	6813	6815#
TS270	020454	10241	10256#
TS271	020646	10299#	
TS272	020702	10310	10312#
TS273	020736	10323	10325#
TS274	021026	10343	10345#
TS275	021116	10364	10374#
TS276	021134	10378	10380#
TS277	021156	10384	10386#
TS3	002162	6627#	
TS30	003740	6824	6825#
TS300	021206	10391	10393#
TS301	021232	10397	10399#
TS302	021254	10403	10409#
TS303	021272	10414	10420#
TS304	021322	10426	10428#
TS305	021364	10436	10438#
TS306	021616	10523#	
TS307	022016	10571	10572#
TS31	003766	6833	6835#
TS310	022256	10611	10612#
TS311	022376	10634	10637#
TS312	022414	10644#	
TS313	022440	10649	10651#
TS314	022530	10678#	
TS315	022564	10679#	
TS316	022620	10681#	
TS317	022654	10689#	
TS32	004020	6844	6846#
TS320	022710	10690#	
TS321	022744	10691#	
TS322	023010	10693	10694#

TS323	023044	10696#		
TS324	023126	10712#		
TS325	023156	10721#		
TS326	023216	10730	10731#	
TS327	023254	10739	10742#	
TS33	004046	6854	6856#	
TS330	023310	10751	10754#	
TS331	023364	10771#		
TS332	023424	10782	10784#	
TS333	023510	10800#		
TS334	023560	10822#		
TS335	024020	10878#		
TS336	024120	10893#		
TS337	024230	10909#		
TS34	004100	6865	6867#	
TS340	024362	10929#		
TS341	024426	10930	10935	10937#
TS342	024514	10938	10952#	
TS343	024704	10982#		
TS344	024736	10992	10997#	
TS345	024766	11003	11010#	
TS346	025016	11016	11026#	
TS347	025134	11052#		
TS35	004126	6875	6877#	
TS350	025176	11063#		
TS351	037670	12298#		
TS352	037724	12317#		
TS353	037766	12338#		
TS354	040054	12356	12369#	
TS355	040112	12376	12404#	
TS356	040136	12416#		
TS357	040156	12422#		
TS36	004160	6886	6889#	
TS360	040176	12428#		
TS361	040216	12434#		
TS362	040236	12447#		
TS363	040354	12495#		
TS364	040410	12515#		
TS365	040472	12531	12544#	
TS366	040562	12569	12581#	
TS367	040660	12608	12621#	
TS37	004206	6897	6899#	
TS370	040764	12649	12660#	
TS371	041070	12687	12702#	
TS372	041234	12743#		
TS373	041420	12746	12832#	
TS374	042074	12886	12905#	
TS375	042424	12937#		
TS376	042772	12999	13010#	
TS377	043146	13016#		
TS4	002402	6645#		
TS40	004240	6908	6910#	
TS400	043334	13030#		
TS401	043456	13079#		
TS402	043734	13098	13145#	
TS403	044124	13232#		

TS404	044424	13255	13285	13333#
TS405	044710	13354	13384	13435#
TS406	045064	13481#		
TS407	045440	13562#		
TS41	004266	6918	6920#	
TS410	045576	13596#		
TS411	045642	13607	13624#	
TS412	045774	13670#		
TS413	046300	13744#		
TS414	046774	13881#		
TS415	047526	13939	14021#	
TS416	050126	14135#		
TS417	050210	14168#		
TS42	004320	6929	6945#	
TS420	050756	14364#		
TS421	051174	14405#		
TS422	051322	14432#		
TS423	051774	14522#		
TS424	052272	14574#		
TS425	052704	14664#		
TS426	053012	14691#		
TS427	053156	14729#		
TS43	004342	6949	6951#	
TS430	053514	14800#		
TS431	054114	14881#		
TS432	054654	15108#		
TS433	054746	15139#		
TS434	055002	15165#		
TS435	055232	15256#		
TS436	055400	15309#		
TS437	055446	15339#		
TS44	004362	6954	6956#	
TS440	056006	15440#		
TS441	056310	15538#		
TS442	056614	15645#		
TS443	060424	15739#		
TS444	061312	15830#		
TS445	061620	15924#		
TS446	061764	15984#		
TS447	062130	16041#		
TS45	004402	6959	6961#	
TS450	062326	16111#		
TS451	062464	16166#		
TS452	062640	16226#		
TS453	063014	16287#		
TS454	063136	16335#		
TS455	063270	16398#		
TS456	064056	16540#		
TS457	064432	16634#		
TS46	004422	6964	6996#	
TS460	065026	16735#		
TS461	065256	16800#		
TS462	065470	16863#		
TS463	065774	16962#		
TS464	066604	17128#		
TS465	067404	17287#		

TS466	070304	174640	
TS467	070542	175360	
TS47	004450	7004	70160
TS470	070766	176220	
TS471	071542	177900	
TS472	072544	179930	
TS473	073406	181830	
TS474	074212	184170	
TS475	074466	185040	
TS476	075220	186780	
TS477	075712	188120	
TS5	002524	66460	
TS50	004506	7029	70370
TS500	076444	189850	
TS501	076776	190920	
TS502	077300	192210	
TS503	077662	193570	
TS504	100214	194890	
TS505	100626	196300	
TS506	101562	198460	
TS507	102776	200670	
TS51	004526	7043	70530
TS510	103436	202170	
TS511	104142	203570	
TS512	105040	204930	
TS513	105134	20511	205340
TS514	105316	205880	
TS515	105604	206630	
TS516	105756	207200	
TS517	106142	207760	
TS52	004552	7061	70700
TS520	106306	208250	
TS521	106452	208710	
TS522	106610	209130	
TS523	106764	209600	
TS524	107416	211350	
TS525	110050	213050	
TS526	110110	213280	
TS527	110230	213710	
TS53	004606	7082	70940
TS530	110270	213950	
TS531	110330	214200	
TS532	110464	214670	
TS533	110622	215090	
TS534	110750	215470	
TS535	111106	215900	
TS536	111254	216310	
TS537	111422	216730	
TS54	004646	7108	71200
TS540	111554	217100	
TS541	111724	217510	
TS542	112042	217850	
TS543	112200	218300	
TS544	112324	218690	
TS545	112454	219070	
TS546	112604	219460	

TS547	112746	21986#	
TS55	004710	7135	7147#
TS550	113106	22028#	
TS551	113250	22068#	
TS552	113400	22106#	
TS553	113534	22142#	
TS554	113676	22185#	
TS555	114050	22234#	
TS556	114770	22498#	
TS557	115052	22525#	
TS56	004752	7162	7169#
TS560	115126	22554#	
TS561	115226	22574#	
TS562	115334	22596#	
TS563	115434	22618#	
TS564	115530	22640#	
TS565	115640	22672#	
TS566	115704	22697#	
TS567	116002	22730#	
TS57	005022	7187	7194#
TS570	117002	22969#	
TS571	117556	23121#	
TS572	121062	23372#	
TS573	121174	23401#	
TS574	121306	23431#	
TS575	121420	23459#	
TS576	121542	23489#	
TS577	121666	23519#	
TS6	002652	6655#	
TS60	005074	7213	7214#
TS600	122012	23551#	
TS601	122150	23588#	
TS602	122226	23612#	
TS603	122304	23640#	
TS604	123404	23880#	
TS605	123444	23901#	
TS606	124002	24024#	
TS607	124110	24055#	
TS61	005154	7233	7234#
TS610	124200	24076#	
TS611	124670	24196#	
TS612	124716	24205#	
TS613	124744	24213#	
TS614	125032	24235#	
TS615	125064	24243	24247#
TS616	125112	24255#	
TS617	125140	24265#	
TS62	005226	7251	7267#
TS620	125256	24286#	
TS621	125350	24305#	
TS622	125450	24323#	
TS623	125566	24346#	
TS624	126020	24388#	
TS625	126046	24395#	
TS626	126202	24419#	
TS627	126372	24459#	

XXCDON	123440	238920				
XXXDON	107412	21056	211280			
XXX1	106764	209630				
XXX2	107030	209820				
XXX3	107074	210010				
XXX4	107140	210200				
XXX5	107204	210390				
X10	064572	16666	166670			
X11	064606	16670	166710			
X12	064626	166760				
X13	064652	166830	16685			
X14	064660	16684	166850			
X15	064674	16688	166890			
X16	064714	166940				
X17	064740	167010	16703			
X2	064452	166400				
X20	064746	16702	167030			
X3	064476	166470	16649			
X4	064504	16648	166490			
X5	064520	16652	166530			
X6	064540	166580				
X7	064564	166650	16667			
Y	124414	24096	241200			
YBR	001760	65630	65660	65760		
YDAT00	065212	16748	167710			
YDAT01	065214	167720				
YDAT02	065216	167730				
YDAT03	065220	167740				
YDONE	065252	16764	167910			
YFLAG	065202	167360	16757	167590	167660	
YNTAB	021470	6553	104660			
YPAT00	065222	16737	16761	167760		
YPAT01	065224	167770				
YPAT02	065226	167780				
YPAT03	065230	167790				
YPAT10	065232	16738	16760	167810		
YPAT11	065234	167820				
YPAT12	065236	167830				
YPAT13	065240	167840				
YPAT20	065242	16742	167860			
YPAT21	065244	167870				
YPAT22	065246	167880				
YPAT23	065250	167890				
YTMP1	065204	167370	16744	167600	167670	
YTMP2	065206	167380	16751	167610	167680	
YTMP3	065210	167390	16755	167620	167690	
YYBDON	114044	22214	222280			
YYBTP1	114020	22187	22193	221940	22204	222170 22225
YYBTP2	114030	22188	22205	222210		
YYBTP3	114040	22192	22210	222250		
YYB10	114042	22203	22208	22211	22213	222260
YYB2	113746	221990				
YYCDON	123776	23954	240120			
YYC1	123444	239040				
YYC2	123472	239120				
YYC3	123520	239200				

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 32-56
CROSS REFERENCE TABLE USER SYMBOLS

SEQ 0512

YYC4	123546	23929#				
YYC5	123574	23938#				
YYC6	123622	23947#				
YYYDON	110044	21231	21299#			
YYY1	107416	21138#				
YYY2	107462	21157#				
YYY3	107526	21176#				
YYY4	107572	21195#				
YYY5	107636	21214#				
Y1	065054	16740#	16763			
Y2	065074	16745#				
Y3	065120	16752#	16754			
Y4	065140	16756	16757#			
Y5	065200	16758	16764#			
Z	124416	24097#	24115#	24116	24121#	
ZDAT00	065424	16812	16833#			
ZDAT01	065426	16834#				
ZDAT02	065430	16835#				
ZDAT03	065432	16836#				
ZDONE	065464	16827	16853#			
ZFLAG	065416	16801#	16821	16823#	16829#	
ZPAT00	065434	16802	16838#			
ZPAT01	065436	16839#				
ZPAT02	065440	16840#				
ZPAT03	065442	16841#				
ZPAT10	065444	16824	16843#			
ZPAT11	065446	16844#				
ZPAT12	065450	16845#				
ZPAT13	065452	16846#				
ZPAT20	065454	16806	16848#			
ZPAT21	065456	16849#				
ZPAT22	065460	16850#				
ZPAT23	065462	16851#				
ZTMP1	065420	16802#	16808	16815	16824#	16830#
ZTMP2	065422	16803#	16819	16825#	16831#	
ZZCBF	124072	24031	24037	24049#		
ZZCDON	124104	24044	24052#			
ZZC10	124066	24039	24041	24043	24045#	
ZZC2	124010	24029#	24040			
ZZC3	124030	24034#				
ZZZDON	110104	21317	21321#			
ZZZ10	110102	21314	21316	21319#		
ZZZ2	110056	21309#				
Z1	065276	16804#	16826			
Z2	065316	16809#				
Z3	065342	16816#	16818			
Z4	065350	16817	16818#			
Z5	065362	16820	16821#			
Z6	065414	16822	16827#			
#APTHD	001030	6425#				
#BDAOR	131464	24946#				
#BDDAT	131466	24947#				
#CPUOP	001026	6424#				
#DEVCT	001010	6424#				
#DOAGN	133310	25116	25137#			
#ENDAD	133226	6422	25125#	25172		

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07-JAN-85 09:28 PAGE 32-57
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0513

\$ENV	001020	6424#	6494	6516	6636	10086	10520	10930	10938	10956	12746	14327	14343	14488
		14781	14986	14998	24133	24214	24370	24407	24460	24487	24510	24555	24573	24614
		24633	24657	24731	24760	24865	24900	24955	24982	25014	25100			
\$ENVM	001021	6424#	25160											
\$EOPCT	133162	25115#	25119#											
\$ETABL	001020	6424#												
\$ETEND	001030	6424#	6425											
\$FATAL	001002	6424#	6511#	6634#	14341#	14486#	14779#	14984#	14996#	24131#	24368#	24508#	24953#	25098#
\$GDADR	131470	24948#												
\$GDDAT	131472	24949#												
\$HIBTS	001030	6425#												
\$MAIL	001000	6424#	6425											
\$MBADR	001032	6425#												
\$MSGAD	001014	6424#												
\$MSGLG	001016	6424#												
\$MSGTY	001000	6424#	6512#	6635#	14342#	14487#	14780#	14985#	14997#	24132#	24369#	24509#	24954#	25099#
\$PASS	001006	6424#	6500#	10086	10930	10938	10958	11352	11385	11416	11447	11479	11512	11651
		11690	11729	12746	14329	24216	24409	24460	24487	24555	24573	24614	24635	24659
		24731	24762	24865	24900	24982	25016	25117#	25118#					
\$PASTM	001036	6425#												
\$SETUP	000020	6484#												
\$STUP	177777	6484#												
\$SVPC	000400	6422#												
\$SMR	000000	6351#												
\$SMREG	001022	6424#	10522											
\$TESTN	001004	6424#	6477	6510#	10210	11323	12287#	14373#	14536#	14806#	15095#	24192#	24384#	24533#
		24978#												
\$TMP0	037572	12272#	12847#	12848#	12849#	12909#	12910#	12911#	12912#	12963#	13096#	13122	13156#	13182
		13503#	13510	13694#	13700#	13704#	13707#	13711#	13712	24082#	24124	25021#	25086	25131#
		25136												
\$TMP1	037574	12272#	13500#	13512#	13515#	13516	13521#	13527#	13531#	13532	24083#	24125	25022#	25087
\$TMP2	037576	12272#	13504#	13513	13832#	13833	13855#	13856	13962	13963#	13993#	13995	14093#	14094
		14116#	14117	20499#	20643#	22504#	22559#	25023#	25088					
\$TMP3	037600	12272#	25024#	25089										
\$TMP4	037602	12272#	25025#	25090										
\$TN	000663	6352#	6551#	6626#	6627#	6645#	6646#	6655#	6662#	6663#	6685#	6689	6691#	6695
		6697#	6701	6703#	6706	6722#	6727	6729#	6735	6737#	6745	6747#	6752	6754#
		6762	6784#	6788	6790#	6793	6795#	6798	6800#	6803	6805#	6813	6815#	6824
		6825#	6833	6835#	6844	6846#	6854	6856#	6865	6867#	6875	6877#	6886	6889#
		6897	6899#	6908	6910#	6918	6920#	6929	6945#	6949	6951#	6954	6956#	6959
		6961#	6964	6996#	7004	7016#	7029	7037#	7043	7053#	7061	7070#	7082	7094#
		7108	7120#	7135	7147#	7162	7169#	7187	7194#	7213	7214#	7233	7234#	7251
		7267#	7279	7293#	7311	7327#	7344	7345#	7369	7379#	7392	7409#	7425	7435#
		7444	7455#	7465	7467#	7479	7480#	7497	7498#	7511	7512#	7526	7536#	7544
		7555#	7563	7573#	7581	7591#	7600	7609#	7627	7636#	7648	7658#	7680	7689#
		7703	7714#	7736	7745#	7756	7765#	7779	7788#	7800	7810#	7822	7830#	7836
		7842#	7848	7854#	7864	7874#	7899	7906#	7919	7926#	7944	7950#	7963	7971#
		7978	7986#	7993	8002#	8009	8022#	8034	8043#	8051	8060#	8069	8080#	8088
		8096#	8107	8116#	8121	8129#	8134	8142#	8147	8149#	8159	8161#	8171	8173#
		8185	8187#	8208	8211#	8234	8236#	8250	8252#	8271	8273#	8287	8289#	8302
		8304#	8317	8325#	8335	8343#	8356	8363#	8379	8386#	8403	8412#	8423	8433#
		8449	8460#	8477	8487#	8504	8513#	8529	8543#	8550	8567#	8574	8590#	8597
		8608#	8615	8624#	8638	8649#	8673	8682#	8712	8721#	8739	8749#	8758	8770#
		8780	8790#	8796	8806#	8813	8824#	8829	8838#	8843	8853#	8860	8870#	8874
		8884#	8891	8903#	8910	8924#	8929	8943#	8948	8984#	9031	9050#	9131	9141#
		9150	9165#	9178	9179#	9193	9194#	9207	9208#	9223	9236#	9258	9260#	9287

9299#	9306	9308#	9322	9323#	9337	9349#	9378	9380#	9401	9413#	9437	9439#
9467	9469#	9476	9489#	9517	9519#	9548	9560#	9589	9590#	9618	9619#	9649
9650#	9683	9686#	9723	9726#	9740	9743#	9766	9776#	9798	9808#	9829	9837#
9851	9859#	9885	9896#	9909	9911#	9919	9921#	9930	9932#	9940	9942#	9950
9952#	9960	9962#	9970	9980#	9991	9993#	10005	10007#	10019	10021#	10033	10035#
10047	10049#	10061	10063#	10075	10085#	10086	10090	10098#	10117#	10142#	10157#	10170
10181#	10218#	10234#	10241	10256#	10299#	10310	10312#	10323	10325#	10343	10345#	10364
10374#	10378	10380#	10384	10386#	10391	10393#	10397	10399#	10403	10409#	10414	10420#
10426	10428#	10436	10438#	10523#	10571	10572#	10611	10612#	10634	10637#	10644#	10649
10651#	10678#	10679#	10681#	10689#	10690#	10691#	10693	10694#	10696#	10712#	10721#	10730
10731#	10739	10742#	10751	10754#	10771#	10782	10784#	10800#	10822#	10878#	10893#	10909#
10929#	10930	10935	10937#	10938	10952#	10982#	10992	10997#	11003	11010#	11016	11026#
11052#	11063#	12298#	12317#	12338#	12356	12369#	12376	12404#	12416#	12422#	12428#	12434#
12447#	12495#	12515#	12531	12544#	12569	12581#	12608	12621#	12649	12660#	12687	12702#
12743#	12746	12832#	12886	12905#	12937#	12999	13010#	13016#	13030#	13079#	13098	13145#
13232#	13255	13285	13333#	13354	13384	13435#	13481#	13562#	13596#	13607	13624#	13670#
13744#	13881#	13939	14021#	14135#	14168#	14364#	14405#	14432#	14522#	14574#	14664#	14691#
14729#	14800#	14881#	15108#	15139#	15165#	15256#	15309#	15339#	15440#	15538#	15645#	15739#
15830#	15924#	15984#	16041#	16111#	16166#	16226#	16287#	16335#	16398#	16540#	16634#	16735#
16800#	16863#	16962#	17128#	17287#	17464#	17536#	17622#	17790#	17993#	18183#	18417#	18504#
18678#	18812#	18985#	19092#	19221#	19357#	19489#	19630#	19846#	20067#	20217#	20357#	20493#
20511	20534#	20588#	20663#	20720#	20776#	20825#	20871#	20913#	20960#	21135#	21305#	21328#
21371#	21395#	21420#	21467#	21509#	21547#	21590#	21631#	21673#	21710#	21751#	21785#	21830#
21869#	21907#	21946#	21986#	22028#	22068#	22106#	22142#	22185#	22234#	22498#	22525#	22554#
22574#	22596#	22618#	22640#	22672#	22697#	22730#	22969#	23121#	23372#	23401#	23431#	23459#
23489#	23519#	23551#	23588#	23612#	23640#	23880#	23901#	24024#	24055#	24076#	24196#	24205#
24213#	24235#	24243	24247#	24255#	24265#	24286#	24305#	24323#	24346#	24388#	24395#	24419#
24459#	24460	24486#	24487	24537#	24546#	24554#	24555	24572#	24573	24581	24585#	24593#
24605#	24614	24618	24621#	24645#	24670#	24689#	24707#	24730#	24731	24752#	24774#	24780
24784#	24804#	24823#	24847#	24864#	24865	24882#	24899#	24900	24918#	24930#	24981#	24982
25013#												
11202#												
11203#												
6425#												
6424#												
6425#												
6424#	25119											
6551#	6626#	6627#	6645#	6646#	6655#	6662#	6663#	6685#	6691#	6697#	6703#	6722#
6729#	6737#	6747#	6754#	6784#	6790#	6795#	6800#	6805#	6815#	6825#	6835#	6846#
6856#	6867#	6877#	6889#	6899#	6910#	6920#	6945#	6951#	6956#	6961#	6996#	7016#
7037#	7053#	7070#	7094#	7120#	7147#	7169#	7194#	7214#	7234#	7267#	7293#	7327#
7345#	7379#	7409#	7435#	7455#	7467#	7480#	7498#	7512#	7536#	7555#	7573#	7591#
7609#	7636#	7658#	7689#	7714#	7745#	7765#	7788#	7810#	7830#	7842#	7854#	7874#
7906#	7926#	7950#	7971#	7986#	8002#	8022#	8043#	8060#	8080#	8096#	8116#	8129#
8142#	8149#	8161#	8173#	8187#	8211#	8236#	8252#	8273#	8289#	8304#	8325#	8343#
8363#	8386#	8412#	8433#	8460#	8487#	8513#	8543#	8567#	8590#	8608#	8624#	8649#
8682#	8721#	8749#	8770#	8790#	8806#	8824#	8838#	8853#	8870#	8884#	8903#	8924#
8943#	8984#	9050#	9141#	9165#	9179#	9194#	9208#	9236#	9260#	9299#	9308#	9323#
9349#	9380#	9413#	9439#	9469#	9489#	9519#	9560#	9590#	9619#	9650#	9686#	9726#
9743#	9776#	9808#	9837#	9859#	9896#	9911#	9921#	9932#	9942#	9952#	9962#	9980#
9993#	10007#	10021#	10035#	10049#	10063#	10085#	10098#	10117#	10142#	10157#	10181#	10218#
10234#	10256#	10299#	10312#	10325#	10345#	10374#	10380#	10386#	10393#	10399#	10409#	10420#
10428#	10438#	10523#	10572#	10612#	10637#	10644#	10651#	10678#	10679#	10681#	10689#	10690#
10691#	10694#	10696#	10712#	10721#	10731#	10742#	10754#	10771#	10784#	10800#	10822#	10878#
10893#	10909#	10929#	10937#	10952#	10982#	10997#	11010#	11026#	11052#	11063#	12298#	12317#
12338#	12369#	12404#	12416#	12422#	12428#	12434#	12447#	12495#	12515#	12544#	12581#	12621#

\$TPB 025672
 \$TPS 025674
 \$TSTM 001034
 \$UNIT 001012
 \$UNITM 001040
 \$USWR 001024
 \$X 131776

188

12660#	12702#	12743#	12832#	12905#	12937#	13010#	13016#	13030#	13079#	13145#	13232#	13333#
13435#	13481#	13562#	13595#	13624#	13670#	13744#	13881#	14021#	14135#	14168#	14364#	14405#
14432#	14522#	14574#	14664#	14691#	14729#	14800#	14881#	15108#	15139#	15165#	15256#	15309#
15339#	15440#	15538#	15645#	15739#	15830#	15924#	15984#	16041#	16111#	16166#	16226#	16287#
16335#	16398#	16540#	16634#	16735#	16800#	16863#	16962#	17128#	17287#	17464#	17536#	17622#
17790#	17993#	18183#	18417#	18504#	18678#	18812#	18985#	19092#	19221#	19357#	19489#	19630#
19846#	20067#	20217#	20357#	20493#	20534#	20588#	20663#	20720#	20776#	20825#	20871#	20913#
20960#	21135#	21305#	21328#	21371#	21395#	21420#	21467#	21509#	21547#	21590#	21631#	21673#
21710#	21751#	21785#	21830#	21869#	21907#	21946#	21986#	22028#	22068#	22106#	22142#	22185#
22234#	22498#	22525#	22554#	22574#	22596#	22618#	22640#	22672#	22697#	22730#	22969#	23121#
23372#	23401#	23431#	23459#	23489#	23519#	23551#	23588#	23612#	23640#	23880#	23901#	24024#
24055#	24076#	24196#	24205#	24213#	24235#	24247#	24255#	24265#	24286#	24305#	24323#	24346#
24388#	24395#	24419#	24459#	24486#	24537#	24546#	24554#	24572#	24585#	24593#	24605#	24621#
24645#	24670#	24689#	24707#	24730#	24752#	24774#	24784#	24804#	24823#	24847#	24864#	24882#
24899#	24918#	24930#	24981#	25013#								
6406#	6410	6412#	6413	6415#	6419	6422#	6423#	6425#	6429#	6438#	6441#	6444#
6447#	6453#	6461#	6466#	6471#	6474#	6481#	6551	6561	6580	6582	6626	6627
6640	6645	6646	6655	6662	6663	6685	6691	6697	6703	6722	6729	6737
6747	6754	6784	6790	6795	6800	6805	6815	6825	6835	6846	6856	6867
6877	6889	6899	6910	6920	6945	6951	6956	6961	6996	7016	7037	7053
7070	7094	7120	7147	7169	7194	7214	7234	7267	7293	7327	7345	7379
7409	7435	7455	7467	7480	7498	7512	7536	7555	7573	7591	7609	7636
7658	7689	7714	7745	7765	7788	7810	7830	7842	7854	7874	7906	7926
7950	7971	7986	8002	8022	8043	8060	8080	8096	8116	8129	8142	8149
8161	8173	8187	8211	8236	8252	8273	8289	8304	8325	8343	8363	8386
8412	8433	8460	8487	8513	8543	8567	8590	8608	8624	8649	8682	8721
8749	8770	8790	8806	8824	8838	8853	8870	8884	8903	8924	8943	8984
8988	9050	9141	9165	9179	9194	9208	9236	9260	9299	9308	9323	9349
9380	9413	9439	9469	9489	9519	9560	9590	9619	9650	9686	9726	9743
9776	9808	9837	9859	9896	9911	9921	9932	9942	9952	9962	9980	9993
10007	10021	10035	10049	10063	10085	10098	10117	10142	10157	10181	10218	10234
10256	10299	10312	10325	10345	10374	10380	10386	10393	10399	10409	10420	10428
10438	10449	10450	10451	10452	10453	10454	10455	10456	10457	10458	10459	10460
10461	10462	10463	10523	10572	10612	10637	10644	10651	10678	10679	10681	10689
10690	10691	10694	10696	10712	10716	10721	10725	10731	10735	10738	10742	10754
10771	10784	10792	10800	10822	10878	10893	10909	10929	10937	10952	10982	10997
11010	11026	11052	11063	11164	11178#	11180#	11182#	11184#	11186#	11188#	12298	12317
12338	12369	12404	12416	12422	12428	12434	12447	12495	12515	12544	12581	12621
12660	12702	12743	12832	12905	12937	13010	13016	13030	13079	13145	13232	13333
13435	13481	13562	13596	13624	13670	13744	13881	14021	14135	14168	14347	14363#
14364	14405	14432	14496	14503#	14522	14574	14664	14691	14729	14785	14787#	14800
14881	14990	15004#	15108	15139	15165	15256	15309	15339	15440	15538	15645	15739
15830	15924	15984	16041	16111	16123#	16166	16226	16287	16335	16398	16540	16634
16735	16800	16863	16962	17128	17287	17464	17536	17622	17790	17993	18183	18417
18504	18678	18812	18985	19092	19221	19357	19489	19630	19846	20067	20217	20357
20493	20534	20588	20663	20720	20776	20825	20871	20913	20960	21135	21305	21328
21371	21395	21420	21467	21509	21547	21590	21631	21673	21710	21751	21785	21830
21869	21907	21946	21986	22028	22068	22106	22142	22185	22234	22498	22525	22554
22574	22596	22618	22640	22672	22697	22730	22969	23121	23372	23401	23431	23459
23489	23519	23551	23588	23612	23640	23880	23901	24024	24055	24076	24137	24196
24205	24213	24235	24247	24255	24265	24286	24305	24323	24346	24374	24377#	24388
24395	24419	24459	24486	24514	24537	24546	24554	24572	24585	24593	24605	24621
24645	24670	24689	24707	24730	24752	24774	24784	24804	24823	24847	24864	24882
24899	24918	24930	24959	24981	25013	25104	25107#	25109#				
15131	15153	15234	15300	15324	15432	15530	15631	15730	15820	15916	15975	16032
16101	16157	16217	16279	16326	16382	16625	16726	16791	16853	16951	17116	17279

- 133657

.RSET 124552

17456	17526	17601	17774	17985	18171	18405	18493	18668	18801	18974	19080	19208
19341	19477	19617	19834	20055	20205	20349	20483	20524	20578	20656	20713	20769
20818	20864	20906	20953	21128	21299	21321	21364	21385	21411	21465	21507	21546
21589	21630	21672	21709	21750	21784	21823	21868	21906	21945	21984	22026	22067
22105	22141	22183	22228	22488	22522	22551	22573	22595	22617	22639	22662	22688
22720	22961	23110	23362	23398	23427	23455	23485	23516	23547	23580	23604	23629
23870	23892	24012	24052	24072	24126	24147						

.SX * 001030

64250

ACCMAC	15048#	15647	15648	15649	15650	15651	15652	15653	15654	15655	15656	15657	15658		
ADDTST	12168#	12847	12848	12849	12909	12910	12911	12912							
APTSKP	6275#	10086	10930	10938	12746	24460	24487	24555	24573	24614	24731	24865	24900	24982	
COMMEN	1577#														
ENDCOM	1589#														
ERROR	6312#	6577	6626	6627	6645	6646	6655	6662	6663	6689	6695	6701	6706	6727	6733
	6735	6745	6752	6762	6788	6793	6798	6803	6813	6824	6833	6844	6854	6865	6875
	6886	6897	6908	6918	6929	6949	6954	6959	6964	6998	7002	7004	7019	7029	7039
	7043	7056	7061	7075	7082	7100	7103	7127	7135	7154	7162	7177	7187	7201	7204
	7211	7213	7222	7225	7231	7233	7242	7249	7251	7273	7279	7301	7311	7334	7344
	7357	7359	7363	7365	7369	7384	7392	7419	7425	7440	7444	7461	7465	7475	7477
	7479	7492	7494	7497	7509	7511	7521	7526	7539	7544	7563	7581	7600	7618	7627
	7645	7648	7667	7669	7677	7680	7700	7703	7726	7733	7736	7754	7756	7776	7779
	7798	7800	7820	7822	7836	7848	7862	7864	7877	7883	7887	7895	7899	7916	7919
	7931	7933	7936	7941	7944	7959	7963	7978	7993	8009	8031	8034	8051	8066	8069
	8086	8088	8104	8107	8121	8134	8147	8157	8159	8169	8171	8180	8183	8185	8196
	8198	8203	8206	8208	8223	8225	8231	8234	8246	8248	8250	8262	8264	8267	8269
	8271	8283	8285	8287	8298	8300	8302	8313	8315	8317	8333	8335	8351	8354	8356
	8369	8371	8375	8377	8379	8393	8395	8397	8400	8403	8419	8421	8423	8439	8441
	8445	8447	8449	8467	8469	8471	8475	8477	8494	8496	8498	8502	8504	8520	8522
	8524	8527	8529	8550	8574	8597	8615	8631	8638	8657	8664	8673	8692	8701	8712
	8727	8733	8739	8758	8780	8796	8813	8827	8829	8843	8858	8860	8874	8889	8891
	8908	8910	8929	8948	8989	8991	8998	9003	9005	9011	9017	9024	9031	9062	9073
	9084	9094	9104	9112	9120	9131	9148	9150	9171	9178	9186	9193	9201	9207	9216
	9223	9243	9251	9258	9267	9273	9279	9287	9306	9315	9322	9330	9337	9355	9361
	9367	9371	9378	9387	9394	9401	9421	9429	9437	9446	9453	9461	9467	9476	9496
	9503	9510	9517	9527	9534	9541	9548	9568	9575	9581	9589	9598	9605	9612	9618
	9627	9634	9640	9649	9658	9666	9673	9683	9692	9697	9701	9703	9714	9716	9721
	9723	9733	9735	9740	9749	9754	9757	9764	9766	9786	9798	9819	9829	9843	9851
	9867	9874	9881	9885	9901	9909	9917	9919	9928	9930	9938	9940	9948	9950	9958
	9960	9968	9970	9984	9991	10001	10003	10005	10015	10017	10019	10029	10031	10033	10043
	10045	10047	10057	10059	10061	10071	10073	10075	10090	10106	10123	10126	10132	10150	10154
	10167	10170	10209	10225	10241	10267	10282	10305	10310	10318	10323	10333	10336	10340	10343
	10353	10357	10361	10364	10378	10384	10391	10397	10403	10414	10426	10436	10447	10490	10491
	10492	10493	10494	10495	10496	10497	10498	10499	10527	10532	10537	10543	10549	10555	10560
	10566	10571	10579	10587	10595	10603	10611	10614	10617	10620	10623	10625	10628	10631	10634
	10641	10649	10657	10662	10666	10693	10709	10719	10728	10730	10739	10751	10765	10769	10779
	10782	10795	10796	10808	10833	10837	10839	10856	10862	10864	10888	10904	10905	10921	10935
	10946	10975	10977	10979	10990	10992	11003	11014	11016	11033	11034	11036	11038	11040	11114
	11117	11119	11367	11370	11399	11402	11430	11433	11461	11464	11493	11496	11526	11529	11607
	11613	11614	11615	11616	11617	11618	11619	11620	11621	11622	11665	11668	11672	11675	11704
	11707	11711	11713	11743	11746	11750	11752	11865	11869	11870	11871	11872	11873	11874	11875
	11876	11882	11883	11884	11885	11886	11887	11888	11889	11890	11891	11988	11989	11990	11991
	11992	11993	11994	11995	11996	11997	11998	11999	12000	12001	12002	12003	12004	12005	12010
	12011	12012	12013	12014	12015	12016	12017	12018	12019	12030	12031	12032	12033	12034	12035
	12036	12037	12038	12039	12040	12041	12042	12043	12044	12045	12046	12051	12052	12053	12054
	12055	12056	12057	12058	12059	12060	12069	12071	12305	12324	12346	12356	12376	12453	12460
	12470	12478	12501	12518	12524	12531	12530	12559	12587	12598	12631	12642	12670	12680	12753
	12764	12774	12786	12847	12848	12849	12866	12886	12909	12910	12911	12912	12969	13001	13011
	13018	13039	13050	13098	13104	13111	13117	13158	13164	13171	13177	13255	13264	13270	13278
	13354	13363	13369	13377	13451	13488	13495	13517	13533	13541	13577	13607	13631	13640	13646
	13713	13784	13795	13805	13814	13824	13837	13848	13861	13895	13911	13926	13939	13940	13954
	13970	13985	14002	14046	14057	14067	14076	14086	14098	14109	14121	14144	14153	14177	14183
	14274	14287	14299	14312	15127	15151	15174	15183	15192	15200	15209	15217	15225	15233	15271
	15284	15288	15299	15323	15361	15363	15368	15373	15390	15392	15397	15402	15467	15493	15506
	15567	15593	15606	15694	15788	15855	15857	15879	15881	15885	15900	15942	15947	15951	16000

MSG30	130710															
MSG31	131370															
MSG32	132200															
MSG33	133210															
MSG34	134210															
MSG35	134670															
MSG36	135500															
MSG36A	135900															
MSG37	136120															
MSG4	123610															
MSG40	136550															
MSG41	137330															
MSG42	138690															
MSG43	140100															
MSG44	141280															
MSG45	141600															
MSG5	123940															
MSG6	124130															
MSG7	124190															
MULT	45700															
NEWTST	16350	62990	6551	6626	6627	6645	6646	6655	6662	6663	6685	6691	6697	6703	6722	
	6729	6737	6747	6754	6784	6790	6795	6800	6805	6815	6825	6835	6846	6856	6867	
	6877	6889	6899	6910	6920	6945	6951	6956	6961	6996	7016	7037	7053	7070	7094	
	7120	7147	7169	7194	7214	7234	7267	7293	7327	7345	7379	7409	7435	7455	7467	
	7480	7498	7512	7536	7555	7573	7591	7609	7636	7658	7689	7714	7745	7765	7788	
	7810	7830	7842	7854	7874	7906	7926	7950	7971	7986	8002	8022	8043	8060	8080	
	8096	8116	8129	8142	8149	8161	8173	8187	8211	8236	8252	8273	8289	8304	8325	
	8343	8363	8386	8412	8433	8460	8487	8513	8543	8567	8590	8608	8624	8649	8682	
	8721	8749	8770	8790	8806	8824	8838	8853	8870	8884	8903	8924	8943	8984	9050	
	9141	9165	9179	9194	9208	9236	9260	9299	9308	9323	9349	9380	9413	9439	9469	
	9489	9519	9560	9590	9619	9650	9686	9726	9743	9776	9808	9837	9859	9896	9911	
	9921	9932	9942	9952	9962	9980	9993	10007	10021	10035	10049	10063	10085	10098	10117	
	10142	10157	10181	10218	10234	10256	10299	10312	10325	10345	10374	10380	10386	10393	10399	
	10409	10420	10428	10438	10523	10572	10612	10637	10644	10651	10678	10679	10681	10689	10690	
	10691	10694	10696	10712	10721	10731	10742	10754	10771	10784	10800	10822	10878	10893	10909	
	10929	10937	10952	10982	10997	11010	11026	11052	11063	12298	12317	12338	12369	12404	12416	
	12422	12428	12434	12447	12495	12515	12544	12581	12621	12660	12702	12743	12832	12905	12937	
	13010	13016	13030	13079	13145	13232	13333	13435	13481	13562	13596	13624	13670	13744	13881	
	14021	14135	14168	14364	14405	14432	14522	14574	14664	14691	14729	14800	14881	15108	15139	
	15165	15256	15309	15339	15440	15538	15645	15739	15830	15924	15984	16041	16111	16166	16226	
	16287	16335	16398	16540	16634	16735	16800	16863	16962	17128	17287	17464	17536	17622	17790	
	17993	18183	18417	18504	18678	18812	18985	19092	19221	19357	19489	19530	19846	20067	20217	
	20357	20493	20534	20588	20663	20720	20776	20825	20871	20913	20960	21135	21305	21328	21371	
	21395	21420	21467	21509	21547	21590	21631	21673	21710	21751	21785	21830	21869	21907	21946	
	21986	22028	22068	22106	22142	22185	22234	22498	22525	22554	22574	22596	22618	22640	22672	
	22697	22730	22969	23121	23372	23401	23431	23459	23489	23519	23551	23588	23612	23640	23880	
	23901	24024	24055	24076	24196	24205	24213	24235	24247	24253	24265	24286	24305	24323	24346	
	24388	24395	24419	24459	24486	24537	24546	24554	24572	24585	24593	24605	24621	24645	24670	
	24689	24707	24730	24752	24774	24784	24804	24823	24847	24864	24882	24899	24918	24930	24981	
	25013															
NXTTST	63410	12569	12608	12649	12687	12999	13285	13384								
POP	21460															
PUSH	21380															
REPORT	55160															
RSET	150070	15131	15153	15234	15300	15324	15432	15530	15631	15730	15820	15916	15975	16032	16101	
	16157	16217	16179	16326	16382	16625	16726	16791	16853	16951	17116	17279	17456	17526	17601	

TYPDEC	20530															
TYPNAM	18250															
TYPNUM	20200															
TYPOCS	19730															
TYPOCT	19360															
TYPTXT	18910															
USER	121010	12272														
VTRP	106680	10678	10679	10689	10690	10691	10694									
WMSG	122030															
WTST	122270	13011	13018													
ABORT	28610	64210	25172													
ASHC	112370	11869	11870	11871	11872	11873	11874	11875	11876							
ASHC1	112800	11882	11883	11884	11885	11886	11887	11888	11889	11890	11891					
ASHI	112110	11613	11614	11615	11616	11617	11618	11619	11620	11621	11622					
DIV	119340	12030	12031	12032	12033	12034	12035	12036	12037	12038	12039	12040	12041	12042	12043	
	12044	12045	12046	12051	12052	12053	12054	12055	12056	12057	12058	12059	12060			
MUL	118990	11988	11989	11990	11991	11992	11993	11994	11995	11996	11997	11998	11999	12000	12001	
	12002	12003	12004	12005	12010	12011	12012	12013	12014	12015	12016	12017	12018	12019		
SKIP	62830	10086	10930	10938	12746	24460	24487	24555	24373	24614	24731	24865	24900	24982		
ERCD	63330	6577	6626	6627	6645	6646	6655	6662	6663	6689	6695	6701	6706	6727	6733	
	6735	6745	6752	6762	6788	6793	6798	6803	6813	6824	6833	6844	6854	6865	6875	
	6886	6897	6908	6918	6929	6949	6954	6959	6964	6998	7002	7004	7019	7029	7039	
	7043	7056	7061	7075	7082	7100	7108	7127	7135	7154	7162	7177	7187	7201	7204	
	7211	7213	7222	7225	7231	7233	7242	7249	7251	7273	7279	7301	7311	7334	7344	
	7357	7359	7363	7365	7369	7384	7392	7419	7425	7440	7444	7461	7465	7475	7477	
	7479	7492	7494	7497	7509	7511	7521	7526	7539	7544	7563	7581	7600	7618	7627	
	7645	7648	7667	7669	7677	7680	7700	7703	7726	7733	7736	7754	7756	7776	7779	
	7798	7800	7820	7822	7836	7848	7862	7864	7877	7883	7887	7895	7899	7916	7919	
	7931	7933	7936	7941	7944	7959	7963	7978	7993	8009	8031	8034	8051	8066	8069	
	8086	8088	8104	8107	8121	8134	8147	8157	8159	8169	8171	8180	8183	8185	8196	
	8198	8203	8206	8208	8223	8225	8231	8234	8246	8248	8250	8262	8264	8267	8269	
	8271	8283	8285	8287	8298	8300	8302	8313	8315	8317	8333	8335	8351	8354	8356	
	8369	8371	8375	8377	8379	8393	8395	8397	8400	8403	8419	8421	8423	8439	8441	
	8445	8447	8449	8467	8469	8471	8475	8477	8494	8496	8498	8502	8504	8520	8522	
	8524	8527	8529	8550	8574	8597	8615	8631	8638	8657	8664	8673	8692	8701	8712	
	8727	8733	8739	8758	8780	8796	8813	8827	8829	8843	8858	8860	8874	8889	8891	
	8908	8910	8929	8948	8989	8991	8998	9003	9005	9011	9017	9024	9031	9062	9073	
	9084	9094	9104	9112	9120	9131	9148	9150	9171	9178	9186	9193	9201	9207	9216	
	9223	9243	9251	9258	9267	9273	9279	9287	9306	9315	9322	9330	9337	9355	9361	
	9367	9371	9378	9387	9394	9401	9421	9429	9437	9446	9453	9461	9467	9476	9496	
	9503	9510	9517	9527	9534	9541	9548	9568	9575	9581	9589	9598	9605	9612	9618	
	9627	9634	9640	9649	9658	9666	9673	9683	9692	9697	9701	9703	9714	9716	9721	
	9723	9733	9735	9740	9749	9754	9757	9764	9766	9786	9798	9819	9829	9843	9851	
	9867	9874	9881	9885	9901	9909	9917	9919	9928	9930	9938	9940	9948	9950	9958	
	9960	9968	9970	9984	9991	10001	10003	10005	10015	10017	10019	10029	10031	10033	10043	
	10045	10047	10057	10059	10061	10071	10073	10075	10090	10106	10123	10126	10132	10150	10154	
	10167	10170	10209	10225	10241	10267	10282	10305	10310	10318	10323	10333	10336	10340	10343	
	10353	10357	10361	10364	10378	10384	10391	10397	10403	10414	10426	10436	10447	10490	10491	
	10492	10493	10494	10495	10496	10497	10498	10499	10527	10532	10537	10543	10549	10555	10560	
	10566	10571	10579	10587	10595	10603	10611	10614	10617	10620	10623	10625	10628	10631	10634	
	10641	10649	10657	10662	10666	10693	10709	10719	10728	10730	10739	10751	10765	10769	10779	
	10782	10795	10796	10808	10833	10837	10839	10856	10862	10864	10888	10904	10905	10921	10935	
	10946	10975	10977	10979	10990	10992	11003	11014	11016	11033	11034	11036	11038	11040	11114	
	11117	11119	11367	11370	11399	11402	11430	11433	11461	11464	11493	11496	11526	11529	11607	
	11613	11614	11615	11616	11617	11618	11619	11620	11621	11622	11665	11668	11672	11675	11704	
	11707	11711	11713	11743	11746	11750	11752	11865	11869	11870	11871	11872	11873	11874	11875	

	11876	11882	11883	11884	11885	11886	11887	11888	11889	11890	11891	11988	11989	11990	11991
	11992	11993	11994	11995	11996	11997	11998	11999	12000	12001	12002	12003	12004	12005	12010
	12011	12012	12013	12014	12015	12016	12017	12018	12019	12030	12031	12032	12033	12034	12035
	12036	12037	12038	12039	12040	12041	12042	12043	12044	12045	12046	12051	12052	12053	12054
	12055	12056	12057	12058	12059	12060	12069	12071	12305	12324	12346	12356	12376	12453	12460
	12470	12478	12501	12518	12524	12531	12550	12559	12587	12598	12631	12642	12670	12680	12753
	12764	12774	12786	12847	12848	12849	12866	12886	12909	12910	12911	12912	12969	13001	13011
	13018	13039	13050	13098	13104	13111	13117	13158	13164	13171	13177	13255	13264	13270	13278
	13354	13363	13369	13377	13451	13488	13495	13517	13533	13541	13577	13607	13631	13640	13646
	13713	13784	13795	13805	13814	13824	13837	13848	13851	13895	13911	13926	13939	13940	13954
	13970	13985	14002	14046	14057	14067	14076	14086	14098	14109	14121	14144	14153	14177	14183
	14274	14287	14299	14312	15127	15151	15174	15183	15192	15200	15209	15217	15225	15233	15271
	15284	15288	15299	15323	15361	15363	15368	15373	15390	15392	15397	15402	15467	15493	15506
	15567	15593	15606	15694	15788	15855	15857	15879	15881	15885	15900	15942	15947	15951	16000
	16005	16009	16069	16077	16129	16135	16138	16187	16192	16248	16253	16306	16309	16355	16359
	16422	16439	16453	16467	16489	16493	16498	16555	16558	16574	16577	16591	16594	16610	16613
	16648	16652	16666	16670	16684	16688	16702	16706	16753	16756	16817	16820	16877	16895	16916
	16976	16979	16993	16996	17011	17014	17029	17032	17046	17049	17063	17066	17142	17146	17160
	17163	17179	17182	17197	17200	17214	17217	17231	17234	17301	17305	17319	17323	17337	17341
	17355	17358	17372	17375	17389	17393	17407	17411	17478	17482	17496	17500	17551	17554	17569
	17572	17638	17641	17660	17664	17681	17685	17699	17703	17717	17721	17806	17810	17821	17824
	17832	17836	17841	17857	17861	17874	17882	17886	17891	17907	17911	17922	17925	17933	17937
	17942	18002	18010	18014	18026	18036	18041	18052	18065	18082	18105	18110	18124	18129	18390
	18396	18435	18438	18454	18457	18471	18476	18483	18486	18659	18661	18663	18793	18797	18965
	18967	18969	19072	19076	19199	19203	19332	19336	19450	19453	19466	19470	19472	19590	19594
	19607	19610	19612	19816	19818	19821	19823	19825	20038	20045	20048	20078	20100	20121	20182
	20184	20187	20189	20191	20229	20251	20325	20332	20335	20361	20367	20371	20374	20379	20382
	20389	20392	20396	20401	20404	20408	20414	20417	20421	20427	20433	20441	20448	20455	20463
	20470	20506	20511	20520	20522	20556	20577	20610	20629	20646	20655	20697	20768	20816	20863
	20905	20952	21126	21297	21319	21352	21384	21409	21448	21506	21545	21588	21629	21671	21708
	21749	21783	21822	21867	21905	21944	21983	22025	22066	22104	22140	22182	22226	22483	22520
	22550	22572	22594	22616	22638	22661	22686	22710	22713	22556	23108	23357	23396	23425	23454
	23484	23515	23546	23579	23598	23623	23865	24001	24003	24006	24045	24064	24066	24095	24111
	24200	24209	24226	24231	24243	24251	24259	24270	24273	24276	24281	24293	24298	24314	24319
	24335	24342	24359	24392	24400	24403	24406	24415	24474	24480	24501	24541	24550	24563	24568
	24581	24589	24597	24607	24610	24613	24618	24626	24629	24632	24641	24650	24653	24656	24665
	24677	24682	24698	24703	24719	24726	24744	24758	24767	24780	24793	24797	24813	24818	24834
	24842	24859	24878	24890	24892	24910	24913	24925	24927	24938	24941	25003	25060	25078	25083
\$\$\$ERRU	63280	6689	6695	6701	6706	6727	6735	6745	6752	6762	6788	6793	6798	6803	6813
	6824	6833	6844	6854	6865	6875	6886	6897	6908	6918	6929	6949	6954	6959	6964
	7004	7029	7043	7061	7082	7108	7135	7162	7187	7213	7233	7251	7279	7311	7344
	7369	7392	7425	7444	7465	7479	7497	7511	7526	7544	7563	7581	7600	7627	7648
	7680	7703	7736	7756	7779	7800	7822	7836	7848	7864	7899	7919	7944	7963	7978
	7993	8009	8034	8051	8069	8088	8107	8121	8134	8147	8159	8171	8185	8208	8234
	8250	8271	8287	8302	8317	8335	8356	8379	8403	8423	8449	8477	8504	8529	8550
	8574	8597	8615	8638	8673	8712	8739	8758	8780	8796	8813	8829	8843	8860	8874
	8891	8910	8929	8948	9031	9131	9150	9178	9193	9207	9223	9258	9287	9306	9322
	9337	9378	9401	9437	9467	9476	9517	9548	9589	9618	9649	9685	9723	9740	9766
	9798	9829	9851	9885	9909	9919	9930	9940	9950	9960	9970	9991	10005	10019	10033
	10047	10061	10075	10090	10170	10241	10310	10323	10343	10364	10378	10384	10391	10397	10403
	10414	10426	10436	10571	10611	10634	10649	10693	10730	10739	10751	10782	10935	10992	11003
	11016	12356	12376	12531	12886	13098	13255	13354	13607	13939	20511	24243	24581	24618	24780
\$\$\$ESCA	17160														
\$\$\$NEWT	16710	62860	6551	6626	6627	6645	6646	6655	6662	6663	6685	6691	6697	6703	6722
	6729	6737	6747	6754	6784	6790	6795	6800	6805	6815	6825	6835	6846	6856	6867
	6877	6889	6899	6910	6920	6945	6951	6956	6961	6996	7016	7037	7053	7070	7094

7120	7147	7169	7194	7214	7234	7267	7293	7327	7345	7379	7409	7435	7455	7467
7480	7498	7512	7536	7555	7573	7591	7609	7636	7658	7689	7714	7745	7765	7788
7810	7830	7842	7854	7874	7906	7926	7950	7971	7986	8002	8022	8043	8060	8080
8096	8116	8129	8142	8149	8161	8173	8187	8211	8236	8252	8273	8289	8304	8325
8343	8363	8386	8412	8433	8460	8487	8513	8543	8567	8590	8608	8624	8649	8682
8721	8749	8770	8790	8806	8824	8838	8853	8870	8884	8903	8924	8943	8984	9050
9141	9165	9179	9194	9208	9236	9260	9299	9308	9323	9349	9380	9413	9439	9469
9489	9519	9560	9590	9619	9650	9686	9726	9743	9776	9808	9837	9859	9896	9911
9921	9932	9942	9952	9962	9980	9993	10007	10021	10035	10049	10063	10085	10098	10117
10142	10157	10181	10218	10234	10256	10299	10312	10325	10345	10374	10380	10386	10393	10399
10409	10420	10428	10438	10523	10572	10612	10637	10644	10651	10678	10679	10681	10689	10690
10691	10694	10696	10712	10721	10731	10742	10754	10771	10784	10800	10822	10878	10893	10909
10929	10937	10952	10982	10997	11010	11026	11052	11063	12298	12317	12338	12369	12404	12416
12422	12428	12434	12447	12495	12515	12544	12581	12621	12660	12702	12743	12832	12905	12937
13010	13016	13030	13079	13145	13232	13333	13435	13481	13562	13596	13624	13670	13744	13881
14021	14135	14168	14364	14405	14432	14522	14574	14664	14691	14729	14800	14881	15108	15139
15165	15256	15309	15339	15440	15538	15645	15739	15830	15924	15984	16041	16111	16166	16226
16287	16335	16398	16540	16634	16735	16800	16863	16962	17128	17287	17464	17536	17622	17790
17993	18183	18417	18504	18678	18812	18985	19092	19221	19357	19489	19630	19846	20067	20217
20357	20493	20534	20588	20663	20720	20776	20825	20871	20913	20960	21135	21305	21328	21371
21395	21420	21467	21509	21547	21590	21631	21673	21710	21751	21785	21830	21869	21907	21946
21986	22028	22068	22106	22142	22185	22234	22498	22554	22574	22596	22618	22640	22672	22672
22697	22730	22969	23121	23372	23401	23431	23459	23489	23519	23551	23588	23612	23640	23880
23901	24024	24055	24076	24196	24205	24213	24235	24247	24255	24265	24286	24305	24323	24346
24388	24395	24419	24459	24486	24537	24546	24554	24572	24585	24593	24605	24621	24645	24670
24689	24707	24730	24752	24774	24784	24804	24823	24847	24864	24882	24899	24918	24930	24981
25013														

\$\$SKIP	17490		
.EQUAT	1980		
.HEADE	740		
.KT11	3400	120780	12081
.SETUP	12320	64210	6484
.SURMI	1150		
.\$ACT1	51290	64210	6422
.\$APT8	51720	64210	6424
.\$APTH	54270	64210	6425
.\$APTY	55990		
.\$ASTA	54720		
.\$CATC	9240		
.\$CHTA	10330		
.\$DB2D	47650		
.\$DB2D	48860		
.\$DIV	46690		
.\$EOP	22040		
.\$ERR0	26830		
.\$ERRT	29580		
.\$MULT	46070		
.\$POWE	42830		
.\$RAND	43570		
.\$RDDE	39570		
.\$RDOC	38670		
.\$READ	34660		
.\$R2AZ	50270		
.\$SAVE	40310		
.\$SB2D	48480		
.\$SB2D	49470		

CJKL580 LCP 5 CPU CLSTR DIAG
CJKL58.P11 07-JAN-85 09:05

MACY11 30(1046) 07 JAN-85 09:28 PAGE 33.7
CROSS REFERENCE TABLE MACRO NAMES

SEQ 0524

.\$SCOP 2438#
.\$SIZE 4409#
.\$SUPR 4984#
.\$TRAP 4132#
.\$TYPB 3360#
.\$TYPD 3284#
.\$TYPE 3044#
.\$TYPO 3189#
.\$4OCA 962#
.1170 518#

. ABS. 133657 000

ERRORS DETECTED: 0

CJKL58.CJKL58/CRF/NL:TOC=ORION.SML.CJKL58.P11
RUN-TIME: 72 86 12 SECONDS
RUN-TIME RATIO: 488/171=2.8
CORE USED: 51K (102 PAGES)