

PDP11-70

11/70-74 CPU #2  
CEKBBE0

AH-7968E-MC  
FICHE 1 OF 2

MAY 1980  
COPYRIGHT © 75, 79  
MADE IN USA



The main body of the document is a large grid of approximately 20 columns and 20 rows. Each cell in the grid contains technical data, likely representing memory addresses and their corresponding values. The data is organized into columns, with some columns containing binary or hexadecimal strings and others containing more complex alphanumeric sequences. The overall appearance is that of a memory dump or a detailed hardware configuration table.

PDP11-70

11/70-74 CPU #2  
CEKBBE0

AH-7968E-MC  
FICHE 2 OF 2

MAY 1980  
COPYRIGHT © 75, 79  
MADE IN USA



.REM .

IDENTIFICATION

PRODUCT CODE:            AC-7966E-MC  
PRODUCT NAME:            CEKBEO 11/70-74 CPU #2  
DATE :                    DEC, 1979  
MAINTAINER:              DIAGNOSTIC ENGINEERING  
AUTHORS:                  DON MONROE  
MODIFIED BY:              ERNEST PREISIG 11/3/78

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

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

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975,1979 BY DIGITAL EQUIPMENT CORPORATION

CONTENTS

1. ABSTRACT
2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 STORAGE
  - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
  - 3.1 METHOD
4. STARTING PROCEDURE
  - 4.1 CONTROL SWITCH SETTINGS
  - 4.2 STARTING ADDRESS
  - 4.3 PROGRAM AND OPERATOR ACTION
5. OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 SUBROUTINE ABSTRACTS
  - 5.3 OPERATOR ACTION
6. ERRORS
  - 6.1 ERROR HALTS AND DESCRIPTION
  - 6.2 ERROR RECOVERY
7. RESTRICTIONS
  - 7.1 STARTING RESTRICTIONS
  - 7.2 OPERATING RESTRICTIONS
8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 PASS COUNT
  - 8.4 ITERATIONS
  - 8.5 SPECIAL REGISTERS
  - 8.6 T BIT TRAPPING
  - 8.7 OSCILLOSCOPE SYNC POINTS
  - 8.8 CACHE CONTROL
9. PROGRAM DESCRIPTION AND HISTORY
  - 9.1 CEKBB
10. LISTINGS
  - 10.1 CEKBB

1. ABSTRACT

-----  
CEKBA/B ARE PROGRAMS DESIGNED TO DETECT AND REPORT LOGIC FAULTS IN THE PDP 11/70-74MP CENTRAL PROCESSING UNIT. THEY CONSISTS OF 210(8) INDIVIDUAL TESTS CAREFULLY DESIGNED AND SEQUENCED TO DETECT AND ATTEMPT TO IDENTIFY LOGIC FAULTS AT A MINIMUM HARDWARE/SOFTWARE LEVEL. THESE TESTS ARE PARTITIONED INTO TWO STAND-ALONE PROGRAMS AS DESCRIBED BELOW.

A. BASIC INSTRUCTION TESTS

CEKBA CONSISTS OF A LOGICALLY SEQUENCED SET OF INSTRUCTION TESTS DESIGNED TO VERIFY THE INTEGRITY OF THOSE INSTRUCTIONS AND LOGIC OPERATIONS USED BY THE UTILITY ROUTINES THAT PROVIDE ERROR REPORTING AND SCOPE LOOPING FACILITIES FOR CEKBB.

ANY FAULT DETECTED IN THIS PROGRAM CAUSES THE PROGRAM TO "HALT" WITH THE CONSOLE ADDRESS LIGHTS INDICATING THE ERROR PROGRAM COUNTER AND THE CONSOLE DATA LIGHTS SHOWING THE TEST NUMBER (FOR TESTS 24 AND ABOVE). ADDITIONAL FAULT IDENTIFICATION INFORMATION IS AVAILABLE IN THE PROGRAM ANNOTATION FOR THE FAILING TEST.

IF THE PROGRAM HALTS AT LOCATION 6 OR 12 (ADDRESS LIGHTS OF 10 OR 14) THE PROGRAM ANNOTATION FOR THE INDICATED TEST NUMBER, SHOULD GIVE A CLUE TO THE PROBLEM. TO LOOP ON THE ERROR THE HALT MUST BE REPLACED BY THE OCTAL CODE SHOWN IN THE COMMENT FIELD OF THE HALT AND THE PROGRAM RESTARTED AT 200, OR THE START ADDRESS OF THAT PARTICULAR TEST.

DURING THE FIRST PASS THE PROGRAM WILL TYPE "AA" AND THE PROGRAM TITLE.

B. ADVANCED INSTRUCTION AND MISCELLANEOUS LOGIC TESTS

CEKBB CONSISTS OF A LOGICALLY SEQUENCED SET OF INSTRUCTION TESTS FOLLOWED BY A SET OF MISCELLANEOUS LOGIC TESTS. THE INSTRUCTION TESTS COMPLETE THE TEST OF THE PDP 11/70-74MP INSTRUCTION REPERTOIRE. THE LOGIC TESTS VERIFY SUCH THINGS AS: 1) THE INTERNAL REGISTERS; 2) REGISTERS SET 1; 3) INTERNAL INTERRUPTS; 4) BUS REQUEST LEVELS 4, 5, AND 6; 5) INTERNAL TRAPS, AND ABORTS; 6) OUTER MODE SELECTION; AND 7) EXTERNAL TRAPS AND ABORTS. EACH TEST IN THIS PROGRAM CALLS A "SCOPE LOOP" UTILITY THAT FACILITATES USER CONTROL OF TEST SELECTION AND EXECUTION VIA THE CONSOLE SWITCH REGISTER.

UPON DETECTION OF A LOGIC FAULT EACH TEST IN THIS SECTION CALLS AN "ERROR SERVICE" THAT REPORTS IT AS HARD COPY ON THE CONSOLE TERMINAL DEVICE. THE ERROR SERVICE ROUTINE ALSO FACILITATES USER CONTROL OF THE PROGRAM SEQUENCE VIA

CONSOLE SWITCH REGISTER OPTIONS. AFTER REPORTING THE ERROR THE PROGRAM CONTINUES ON ITS NORMAL SEQUENCE UNLESS MODIFIED BY THE USER ACTIVATING THE 'HALT ON ERROR' SWITCH OPTION.

C. IMPORTANT NOTE

THE PROGRAM ANNOTATION IN CEKBA AND THE TYPED ERROR REPORTS IN CEKBB ARE BASED UPON THE KNOWLEDGE THAT ALL PREVIOUS TESTS WERE FAULTLESS AND THAT THERE IS ONLY ONE SINGLE POINT FAILURE IN THE PROCESSOR. THIS MEANS THAT IF EITHER PROGRAM, OR THE PROGRAMS THEMSELVES, ARE NOT RUN IN SEQUENCE, THE ERROR MESSAGE MAY NOT BE VALID.

ALTHOUGH EACH ERROR ANNOTATION AND TYPED MESSAGE CONCLUSION HAS BEEN PROVEN BY PHYSICAL FAULT INSERTION (ONE SIGNAL STUCK LOW), IT IS HUMANLY IMPOSSIBLE TO GUARANTEE THAT THE ERROR REPORT IS 100% CORRECT. THE SOLE FUNCTION OF THE ERROR REPORT IS TO DIRECT THE USER TO THE MOST PROBABLE AREA OF FAILURE.

2. REQUIREMENTS  
-----

2.1 EQUIPMENT  
-----

PDP 11/70-74MP CPU WITH OPERATORS CONSOLE  
LA30 OR EQUIVALENT TERMINAL

2.2 STORAGE  
-----

CEKBA REQUIRES 16K TO LOAD AND RUN  
CEKBB REQUIRES 16K TO LOAD AND 32K TO RUN

2.3 PRELIMINARY PROGRAMS  
-----

CEKBA REQUIRES THAT TWO INSTRUCTIONS WORK:  
'BR' AND 'HALT'

CEKBB REQUIRES THAT CEKBA RUN

3. LOADING PROCEDURE  
-----

3.1 METHOD  
-----

BOTH CEKBA AND CEKBB ARE LOADED FROM THE XXDP MEDIA.  
REFER TO THE XXDP MANUAL FOR FURTHER INFORMATION.

4. STARTING PROCEDURE  
-----

4.1 CONTROL SWITCH SETTINGS

SEE 5.1

4.2 STARTING ADDRESS

200

4.3 PROGRAM AND OPERATOR ACTION

A. CEKBA

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3)
2. LOAD ADDRESS 200
3. PRESS START
4. THE PROGRAM WILL PRINT 'AA' AND THE TITLE THE FIRST TIME THROUGH.
5. THE PROGRAM WILL LOOP AND END OF PASS WILL BE TYPED AFTER THE REQUIRED NUMBER OF PASSES.

B. CEKBB

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3)
2. ENSURE RH CONTROLLER IS ENABLED, IF SW<5>=0
3. IF AN RK05 IS AVAILABLE (AND THERE WAS NO RH) ENSURE AT LEAST ONE DRIVE IS ENABLED; IF SW<5>=0
4. LOAD ADDRESS 200
5. SET SWITCHES (SEE SECTION 5.1)
6. PRESS START
7. THE PROGRAM WILL LOOP AND AN END OF PASS MESSAGE WILL BE TYPED EVERY PASS.

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

A. CEKBA

NONE

B. CEKBB

WITH SW<15:0> 0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE. 'END OF PASS' WILL BE TYPED AT THE COMPLETION OF EACH PASS.

THE SWITCH SETTINGS ARE:

SW<15>=1 ... HALT ON ERROR  
SW<14>=1 ... LOOP ON TEST  
SW<13>=1 ... INHIBIT ERROR TYPEOUTS  
SW<12>=1 ... INHIBIT T BIT TRAPPING  
SW<11>=1 ... INHIBIT ITERATIONS  
SW<10>=1 ... RING BELL ON ERROR  
SW<9>=1 ... LOOP ON ERROR  
SW<8>=1 ... LOOP ON TEST IN SW<7:0>  
SW<7>=1 ... NO ACTION  
SW<6>=1 ... SKIP BUS REQUEST 6 TESTING  
SW<5>=1 ... SKIP BUS REQUEST 5 TESTING  
SW<4>=1 ... SKIP BUS REQUEST 4 TESTING  
SW<0>=1 ... SKIP OPERATOR INTERVENTION TESTING

5.2 SUBROUTINE ABSTRACTS  
-----

A. CEKBA

SEE 5.2.4 AND 5.2.5

B. CEKBB

5.2.1 SPURIOUS ERROR HANDLER

THIS ROUTINE IS CALLED BY AN UNEXPECTED TRAP TO LOCATION 4 OR 114. IT PRINTS A SHORT MESSAGE FOLLOWED BY THE PROGRAM COUNTER AT THE TIME OF THE TRAP AND THE APPROPRIATE ERROR REGISTER (I.E. THE CPU ERROR REGISTER IN CASES OF A TRAP TO 4 AND THE MEMORY ERROR REGISTER IN CASES OF A TRAP TO 114).

5.2.2 SCOPE

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED BETWEEN EACH TEST. IT RECORDS THE STARTING ADDRESS OF EACH TEST IN LOCATION '\$LPADR' AND '\$LPERR' AS IT IS BEING ENTERED. IT ALSO CONTROLS TEST ITERATION, LOOPING, AND SEQUENTIAL FLOW CHECKS (SEE 5.2.8).

5.2.3 ERROR

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6)

5.2.4 TRAP CATCHER

A '+2' - 'HALT' SEQUENCE IS REPEATED FROM LOCATION 0 TO

LOCATION 776 TO CATCH ANY UNEXPECTED TRAPS (EXCEPT 4 AND 114). THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2.



### 5.2.5 TRAP

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION. FOLLOWING ARE THE CALLS USED AND THE LABEL OF THE STARTING ADDRESS OF THE SUBROUTINES.

#### 5.2.5.1 TYPE (\$TYPE)

ROUTINE TO TYPE AN ASCII STRING ON THE TTY.

THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

#### 5.2.5.2 TYPEOC (\$TYPOC)

ROUTINE TO CONVERT A 16-BIT BINARY NUMBER TO A 6-DIGIT OCTAL NUMBER AND TYPE IT.

#### 5.2.5.3 TYPEDS (\$TYPDS)

ROUTINE TO CONVERT A 16-BIT BINARY NUMBER TO A 5-DIGIT SIGNED DECIMAL NUMBER AND TYPE IT.

### 5.2.6 POWER DOWN AND UP

THIS SUBROUTINE CALL (VIA A POWER DOWN) SAVES THE RETURN PC UPON A POWER DOWN. WHEN POWER IS RESTORED A MESSAGE IS TYPED AND THE TEST WILL RESTART.

### 5.2.7 MONITOR RESTORE (QUIT)

THIS SUBROUTINE IS ENTERED BY TYPING A 'CONTROL C' OR WHEN THE END OF PASS IS REACHED AND THE PROGRAM IS RUNNING UNDER A MONITOR. SEE 7.2 FOR DETAILS.

### 5.2.8 CHECK TEST SEQUENCE (SEQENC)

THIS ROUTINE IS CALLED IN THE SCOPE ROUTINE. IT COMPARES THE ADDRESS OF THE SCOPE CALL WITH THE ADDRESS POINTED TO BY \$STNM IN THE 'TEST ADDRESS TABLE'. IF THEY DON'T COMPARE, A MESSAGE IS TYPED, INDICATING THAT A TEST WAS SKIPPED.

### 5.2.9 PERIPHERAL DETERMINATOR AND INTERRUPT ENABLE ROUTINES

#### 5.2.9.1 PERIPHERAL DETERMINATOR

THIS ROUTINE IS EXECUTED, IN LINE, BETWEEN TESTS 32 AND 33 OF CEKBB. IT CHECKS THE SYSTEM TO DETERMINE IF ONE OF FOUR PERIPHERALS IS AVAILABLE (RS04, RP04, TM, OR RK05) FOR BUS REQUEST LEVEL 5 TESTING AND IF A LINE CLOCK IS AVAILABLE FOR LEVEL 6 TESTING. IF A DEVICE IS FOUND, THE ADDRESS OF 'INT5SU' (INTERRUPT 5 SUBROUTINE) AND \$KW11L/P (LINE CLOCK INTERRUPT SUBROUTINE) IS PLACED IN LOCATIONS 'INTER5' AND 'INTER6' RESPECTIVELY.

### 5.2.9.2 INTERRUPT ENABLE ROUTINES

THESE ROUTINES ARE CALLED VIA A "JSR PC,@INTERX" (X=5 OR 6). THE ROUTINE SETS UP AND RESPONDS TO A BUS REQUEST. IF THE BR DOES NOT WORK THE RETURN PC IS INCREMENTED BY 2 AND THE RETURN IS MADE.

### 5.3 OPERATOR ACTION

-----

THE LAST TEST OF CEKBB REQUIRES OPERATOR INTERVENTION. THIS TEST IS ONLY EXECUTED ON PASS 1, IF SW<0>=0. QUESTIONS ARE TYPED ON THE TELETYPE AND THE OPERATOR MUST RESPOND EITHER ON THE CONSOLE OR ON THE TELETYPE.

IF LOCATION 42 IS NON-ZERO, INDICATING THAT THE PROGRAM WAS LOADED BY A MONITOR, THIS TEST IS SKIPPED ON ALL PASSES.

### 6. ERRORS

-----

#### 6.1 ERROR HALTS AND DESCRIPTION

-----

##### A. CEKBA

EVERY ERROR IN CEKBA HALTS THE PROCESSOR. THE COMMENT FIELD OF THE HALT INSTRUCTION CONTAINS THE NAME OF THE SIGNAL THAT WAS MOST LIKELY TO HAVE CAUSED THE ERROR. ALSO, IN THE COMMENT FIELD, IS THE OCTAL CODE THAT SHOULD REPLACE THE HALT IF LOOP ON ERROR IS DESIRED. IF THE PROGRAM HALTS AT LOCATION 6 OR 12 THE USER SHOULD LOOK IN THE TEST DESCRIPTION, OF THE TEST THAT FAILED, TO FIND THE MOST LIKELY CAUSE OF THE ERROR.

##### B. CEKBB

NONE OF THE ERRORS IN CEKBB HALT THE PROCESSOR IF SW<15>=0.

THERE ARE OVER 450(8) UNIQUE ERRORS THAT CAN OCCUR IN THIS PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR ROUTINE IS MADE AND IF SW<13> IS NOT SET, AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

THE DATA STRING WILL CONTAIN, AT A MINIMUM, THE ERROR PC AND THE TEST NUMBER. IN SOME CASES THE EXPECTED AND ACTUAL VALUES OF A REGISTER ARE ALSO INCLUDED.

REFER TO THE LISTING UNDER \$ERRTB FOR THE TYPES OF ERRORS THAT CAN OCCUR.

SEE SECTION 7.1 FOR NON-STANDARD CONFIGURATION.

## 6.2 ERROR RECOVERY

### A. CEKBA

ERROR RECOVERY IS STRICTLY BY USER INTERVENTION.

### B. CEKBB

SW<15:9>=0 - MOST ERRORS WILL CAUSE EXECUTION TO GO TO THE START OF THE NEXT TEST AFTER THE MESSAGE IS TYPED. A FEW TESTS ARE DIVIDED INTO SECTIONS. IN THESE TESTS AN ERROR WILL CAUSE EXECUTION TO GO TO THE NEXT SECTION.

SW<15> 1 - PRESSING THE CONSOLE CONTINUE WILL CAUSE THE PROGRAM TO TYPE AN ERROR MESSAGE AND HALT. PRESSING THE CONSOLE CONTINUE AGAIN WILL CAUSE THE PROGRAM TO CONTINUE AS IF SW<15>=0.

## 7. RESTRICTIONS

### 7.1 STARTING RESTRICTIONS

#### A. CEKBA

NONE

#### B. CEKBB

IF THE USER WANTS TO RUN THE BUS REQUEST 5 TEST HE MUST ENSURE THAT EITHER AN RH CONTROLLER IS ACTIVE OR THAT A UNIBUS DEVICE (RK, RS, RP, TM) IS ACTIVE.

IF A RP11-E IS SHIPPED IN PLACE OF A RP04, THIS REPRESENTS A NON-STANDARD CONFIGURATION AND LOCATION 1244 SHOULD BE CHANGED FROM 176700 TO 176714.

7.2 OPERATING RESTRICTIONS

A. CEKBA

NONE

B. CEKBB

SINCE THE PROGRAM COULD POSSIBLY DESTROY A MONITOR, IN PAGE 6, ALL LOCATIONS BETWEEN 152000 AND 157776 ARE SAVED AT THE BOTTOM OF THE PROGRAM. TO RESTORE THESE LOCTIONS A 'CONTROL C' SHOULD BE TYPED ON THE TERMINAL. THE LOCATIONS WILL BE RESTORED, A MESSAGE TYPED, AND THE PROCESSOR WILL HALT.

IF THE PROGRAM IS RUNNING UNDER A MONITOR THE LOCATIONS ARE RESTORED AND CONTROL IS RETURNED TO THE MONITOR THRU THE END OF PASS LINKAGE.

8. MISCELLANEOUS

8.1 EXECUTION TIME

A. CEKBA

FIVE(5) SECONDS PER END OF PASS MESSAGE IF RUNNING UNDER A MONITOR. 2 MINUTES IF THE PROGRAM WAS DUMPED.

B. CEKBB

THE FIRST PASS TAKES APPROXIMATELY 8 SECONDS. ALL SUBSEQUENT PASSES TAKE APPROXIMATELY 3 MINUTES.

8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

8.3 PASS COUNT

A PROGRAM PASS THRU COUNT IS KEPT IN '\$PASS'.

A. CEKBA

IF THE PROGRAM IS RUNNING UNDER A MONITOR OR WAS LOADED BY ACT 11 THE PROGRAM MAKES 144(8) PASSES FOR EACH END OF PASS MESSAGE. IF THE PROGRAM WAS DUMPED, 4000(8) PASSES ARE MADE FOR EACH END OF PASS MESSAGE. THE PASS COUNT IS DISPLAYED IN THE DATA LIGHTS WHEN DISPLAY IS SELECTED BY THE ROTARY SWITCH.

B. CEKBB

THE PROGRAM MAKES 1 PASS FOR EACH END OF PASS MESSAGE.

8.4 ITERATIONS

A. CEKBA

NONE

B. CEKBB

THE FIRST PASS OF THE PROGRAM WILL AUTOMATICALLY INHIBIT ITERATIONS. ALL SUBSEQUENT PASSES WILL PERFORM FULL, (2000 DECIMAL) ITERATIONS.

8.5 SPECIAL REGISTERS

A. CEKBA

R0 IS RESERVED FOR THE TEST NUMBER.

B. CEKBB

NONE

8.6 T BIT TRAPPING

A. CEKBA

NONE

B. CEKBB

EVERY OTHER PASS, STARTING WITH PASS 2, RUNS WITH THE T BIT ON. THIS CAUSES EVERY INSTRUCTION TO T BIT TRAP THEREFORE, IT IS NOT POSSIBLE TO 'SINGLE INSTRUCTION' THE TEST WITHOUT TURNING THE T BIT OFF.

CERTAIN TESTS AUTOMATICALLY TURN IT OFF IF IT WAS ON. THESE TESTS WILL ALSO TURN IT BACK ON UNLESS THE FOLLOWING TEST REQUIRES THAT IT ALSO BE OFF.

8.7 OSCILLOSCOPE SYNC POINTS

A. CEKBA

BEGINNING WITH TEST 24 EACH TEST HAS AN OSCILLOSCOPE SYNC INSTRUCTION. THE ADDRESS OF THE CONDITION CODE ROM STATE (44)

IS IN THE PROCESSOR MICROBREAK REGISTER (ADDRESS 17777770). THIS WILL CAUSE PIN AE1 (SLOT 10) ON THE BACKPLANE TO GO HIGH EACH TIME A CONDITION CODE (OR NOP) INSTRUCTION IS EXECUTED. THEREFORE, IF THE OSCILLOSCOPE EXTERNAL SYNC IS CONNECTED TO THIS PIN AND THE SYNC SELECT PUT ON EXTERNAL THE OSCILLOSCOPE WILL BE SYNCHRONIZED WITH THE INSTRUCTION IMMEDIATELY PRECEDING THE INSTRUCTION UNDER TEST (IUT).

B. CEKBB

ONLY TESTS 1 THRU 20 CONTAIN SYNC INSTRUCTIONS.

8.8 CACHE CONTROL

THE FIRST PASS OF BOTH PROGRAMS RUN WITH THE CACHE DISABLED (FORCING MISSES IN BOTH GROUPS). ALL SUBSEQUENT PASSES RUN WITH THE CACH ENABLED.

9. PROGRAM DESCRIPTION AND HISTORY

- CEKBBB - PROGRAM ENHANCED TO LOOP ON SUBTEST WITH SWITCH 8 SET
- CEKBBC - DOCUMENTATION OF NON-STANDARD RP11 WAS ADDED TO SECTION 6.1
- CEKBBD - ERROR POINTOUTS OF CALLS > AN EMT 377 WHICH DECODES NEXT WORD FOR THE ERROR POINTER WAS INCORRECT. MEMORY SIZING ROUTINE ENHANCED TO HANDLE UNEXPECTED MAIN MEMORY TIMEOUT TRAPS TO 114. PROGRAM ENHANCED TO RUN ON AN 11/74. ALSO, THIS SECTION OF REV HISTORY WAS ADDED TO THE DOC. (9.0) .
- CEKBBE - CHGE1 AND CHGE2 ADDED TO ELIMINATE TTY INTERRUPTS FROM OCCURRING DURING PRIORITY ARBITRATION LOGIC TESTS. (TEST 47)

DOCUMENT  
\*\*\*\*\*  
PDP 11/70-74MP CPU DIAGNOSTIC PART 2  
\*\*\*\*\*

COPYRIGHT 1975,1979  
DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASS. 01754

TABLE OF CONTENTS  
\*\*\*\*\*

41	BASIC DEFINITIONS
166	CACHE REGISTER DEFINITIONS
177	CPU REGISTER DEFINITIONS
191	MEMORY MANAGEMENT DEFINITIONS
340	UNIBUS MAP REGISTER DEFINITIONS
434	TRAP CATCHER
441	STARTING ADDRESS(ES)
447	ACT11 HOOKS
473	COMMON TAGS
547	ERROR POINTER TABLE
3188	PERIPHERAL DETERMINATOR & INTERRUPT ENABLE ROUTINES
5295	MEMORY MANAGEMENT SETUP
6074	END OF PASS ROUTINE
6146	SPURIOUS ERROR HANDLER
6211	SCOPE HANDLER ROUTINE
6277	ERROR HANDLER ROUTINE
6328	ERROR MESSAGE TYPEOUT ROUTINE
6371	STACK LIMIT TEST TYPE OUT ROUTINE
6489	MONITOR RESTORE ROUTINE
6527	CHECK TEST SEQUENCE ROUTINE
6588	TYPE ROUTINE
6661	BINARY TO OCTAL (ASCII) AND TYPE



TABLE OF CONTENTS  
\*\*\*\*\*

6739 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE  
6807 TRAP DECODER  
6822 TRAP TABLE  
6837 POWER DOWN AND UP ROUTINES  
17 COPYRIGHT (C) 1975,1979  
DIGITAL EQUIPMENT CORP.  
MAYNARD, MASS. 01754  
PROGRAM BY DONALD W. MONROE  
THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
PACKAGE (MAINDEC-11-DZQAC-A5).

41 \*\*\*\*\*  
BASIC DEFINITIONS  
\*\*\*\*\*

43 INITIAL ADDRESS OF THE STACK POINTER \*\*\* 1100 \*\*\*  
57 MISCELLANEOUS DEFINITIONS  
63 GENERAL PURPOSE REGISTER DEFINITIONS  
84 PRIORITY LEVEL DEFINITIONS  
94 'SWITCH REGISTER' SWITCH DEFINITIONS  
122 DATA BIT DEFINITIONS (BIT00 TO BIT15)  
150 BASIC 'CPU' TRAP VECTOR ADDRESSES

166 \*\*\*\*\*  
CACHE REGISTER DEFINITIONS  
\*\*\*\*\*

177 \*\*\*\*\*  
CPU REGISTER DEFINITIONS  
\*\*\*\*\*

191 \*\*\*\*\*  
MEMORY MANAGEMENT DEFINITIONS  
\*\*\*\*\*

194 MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

205 USER 'I' PAGE DESCRIPTOR REGISTERS  
216 USER 'D' PAGE DESCRIPTOR REGISTERS  
227 USER 'I' PAGE ADDRESS REGISTERS  
238 USER 'D' PAGE ADDRESS REGISTERS

249 SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS  
260 SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS  
271 SUPERVISOR 'I' PAGE ADDRESS REGISTERS  
282 SUPERVISOR 'D' PAGE ADDRESS REGISTERS  
293 KERNEL 'I' PAGE DESCRIPTOR REGISTERS  
304 KERNEL 'D' PAGE DESCRIPTOR REGISTERS  
315 KERNEL 'I' PAGE ADDRESS REGISTERS  
326 KERNEL 'D' PAGE ADDRESS REGISTERS

340 \*\*\*\*\*  
UNIBUS MAP REGISTER DEFINITIONS  
\*\*\*\*\*

343 THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'  
THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

434 \*\*\*\*\*  
TRAP CATCHER  
\*\*\*\*\*

437 ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'  
SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS  
LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

441 \*\*\*\*\*  
STARTING ADDRESS(ES)  
\*\*\*\*\*

447 \*\*\*\*\*  
ACT11 HOOKS  
\*\*\*\*\*

449 THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11  
LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL  
END OF THE PROGRAM.  
LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS  
AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS

TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:

BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING  
=0 NO POWER FAIL DESIRED

BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT  
=0 RUN TIME IS NOT MEMORY SIZE DEPENDENT

BITS 13-0 MUST BE ZERO'S

473

\*\*\*\*\*  
COMMON TAGS  
\*\*\*\*\*

475 THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
USED IN THE PROGRAM.

547

\*\*\*\*\*  
ERROR POINTER TABLE  
\*\*\*\*\*

549 THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

555 EM ::POINTS TO THE ERROR MESSAGE  
DH ::POINTS TO THE DATA HEADER  
DT ::POINTS TO THE DATA  
DF ::POINTS TO THE DATA FORMAT

1560 TEST 1 SPL

IF FORK A FAILS EXECUTION WILL GO TO ONE OF 3 STATES:  
RSD.02,SVC.70, OR D30.00.  
RSD.02 WILL CAUSE A TRAP TO LOCATION 10.  
SVC.70 WILL HANG THE PROCESSOR IN THE PAUSE STATE. THIS  
WILL ONLY OCCUR IF RACF E17(AFI04(1)\*AFI05(0)\*(RTS:CCOP))  
IS BAD.  
D30.00 WILL CAUSE A TRAP TO LOCATION 10 AFTER STATE D10.60.  
THIS FAILURE CAN BE DIFFERENTIATED FROM THE FIRST BY TESTING  
THE REGISTER ASSOCIATED WITH BITS <2:0> OF THE OP CODE TO

1571

SEE IF IT WAS INCREMENTED.

IF BOTH LEVELS 2 AND 5 COME UP AS LEVEL 4, PDRD E31(1)  
COULD BE BAD.

ONCE IT IS DETERMINED THAT THE INSTRUCTION WORKS A BIT TEST IS  
MADE ON THE PSW <7:5>.

ROM FLOW-43,361

1614 TEST 2 RESET

IF FORK A FAILS EXECUTION WILL EITHER GO TO WAT.00 OR TRP.02.  
THE LA30 PRINTER WILL BE STARTED SUCH THAT A FAILURE INTO  
WAT.00 WILL RECOVER.

IF TRP.01 IS ENTERED A TRAP SEQUENCE WILL EXECUTE  
WITH A TRAP VECTOR OF 4.

ROM FLOW-15,255,374

1672 TEST 3 MARK

FORK A CAN FAIL INTO ONE OF THE FOLLOWING:  
RSD.00, MFP.80, MTP.00, SVC.70, OR D67.01.  
STATE RSD.00 WILL CAUSE A TRAP TO LOCATION 10.  
MFP.80 WILL EXECUTE AN MFP INSTRUCTION.  
MTP.00 WILL EXECUTE AN MTP INSTRUCTION EXCEPT FOR THE ALU.

1679

SVC.70 WILL HANG THE PROCESSOR IN THE PAUSE CONDITION.  
THIS WILL ONLY HAPPEN IF RACF E8 IS BAD.  
D67.01 WILL STEP THE PCB AND TRAP TO LOCATION 10 AFTER STATE D10.60.

ROM FLOW-47,252,235,234

1733 TEST 4 ASH\*DM0

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.

IF GRAJ SC05 L DOES NOT GO LOW OR DOES NOT GET THRU TO  
RACK BRCAB04 L A SHIFT RIGHT WILL OCCUR.  
IF THE SHIFT COUNTER DOES NOT SHIFT OR GRAJ SC=0 DOES NOT GO  
LOW THE PROCESSOR WILL HANG UP IN STATE ASH.41.

ROM FLOW-52,305,257,166 LEFT SHIFT  
52,305,277 RIGHT SHIFT

1866 TEST 5 ASH\*DM1

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.

IF FORK B FAILS EXECUTION WILL EITHER GO TO RSD.00 OR  
MUL.00.  
MUL.00 WILL ONLY BE ENTERED IF EITHER B FORK MUX INPUT B2  
DOES NOT GO HIGH OR THE MUX IS BAD.

ROM FLOW-1,175,62,52,305,257

1902 TEST 6 ASH\*DM2

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.

ALL OTHER LOGIC HAS BEEN TESTED

ROM FLOW-2,175,62,52,305

1923 TEST 7 ASH\*DM4

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.

ALL OTHER LOGIC HAS BEEN TESTED.

ROM FLOW-4,122,177,62,52,305

1945 TEST 10 ASHC\*DMO

NEITHER FORK A NOR BEN03 SHOULD FAIL.

1949

IF THE INSTRUCTION FAILS, ONE OF THE ASC STATES IS BAD.

ONCE IT IS DETERMINED THAT THE INSTRUCTION WORKS,  
A BIT STUCK TEST IS PERFORMED ON THE SHIFT COUNTER.

ROM FLOW-53,306,267,227 RIGHT SHIFT  
53,306,247,176,136 LEFT SHIFT  
53,306,207 NO SHIFT

2078 TEST 11 ASHC\*DM1

THE ONLY POSSIBLE FAILURES ARE FORK B OR STATE ASC.00.

IF FORK B FAILS EXECUTION WILL EITHER GO TO RSD.00 OR ASH.00.

ASH.00 WOULD PERFORM AN ASH INSTRUCTION INSTEAD OF AN ASHC.

ROM FLOW-1,175,63,53,306,267,227

2115 TEST 12 MUL\*DM0

FORK A SHOULD NOT FAIL.

THE FOLLOWING WOULD BE BEN11 FAILURES:

IF EITHER GRAD DROO IS STUCK HIGH OR NOT GETTING THRU TO RACK E64(C1) OR RACK E64 IS BAD (INPUT C1 FAILED HIGH) THE MULTIPPLICAND WILL BE MULTITPLIED BY 177777.

IF EITHER GRAD DROO IS STUCK LOW OR NOT GETTING THRU TO RACK E64(C1) OR RACK E64 IS BAD (INPUT C1 FAILED LOW) THE MULTIPLICAND WILL BE MULTIPLIED BY ZERO.

IF GRAJ SC=0 IS NOT GETTING TO RACK E50(C1) AND RACK E50 IS BAD (INPUT C1 FAILED LOW) THE MULTIPLICAND WILL ONLY BE MULTIPLIED BY BIT 0 OF THE MULTIPLIER.

IF RACK E50(C1) FAILS HIGH THE PROCESSOR WILL HANG UP IN STATE MUL.20(266).

IF THE INSTRUCTION FAILS TO MULTIPLY CORRECTLY AND ONE OF THE ABOVE CONDITIONS CANNOT BE DETERMINED THEN THE FAILURE COULD BE IN THE INSTRUCTION DECODE ROM.

IF THE CC'S COME UP BAD THEN THE FAILURE COULD BE IN STATES MUL.40,50, OR 60, OR IN THE CONDITION CODE ROM.

ROM FLOW-50,102,266/246,226/206,310

2222 TEST 13 MUL\*DM1

FORK A SHOULD NOT FAIL.

FORK B WILL FAIL TO RSD.00 IF THE R(MUL:ASHC\*MFP) FIELD OF THE INSTRUCTION DECODE ROM IS BAD.

IF STATE MUL.00 FAILS THE RESULT WILL BE BAD.

ROM FLOW-1,175,60,102,266/246,226/206,310

2254 TEST 14 DIV\*DM0

FORK A SHOULD NOT FAIL.

SECTION 1

THE FIRST SECTION HAS A ZERO DIVISOR. IF STATE DIV.00 FAILS OR IRCF 22(1) DOES NOT GET TO RACK E49 OR RACK E49(B1) IS STUCK LOW EXECUTION WILL GO FROM DIV.10 TO DIV.20. THIS WILL CAUSE THE DIVIDE TO ABORT (GO TO DVE.20) AFTER STATE DIV.60 AND THE C BIT WILL BE CLEAR.

SECTION 2

THE NEXT SECTION DIVIDES 4 BY 2. IF RACK E49(B1) IS STUCK HIGH THE ALGORITHM WILL ABORT THINKING THAT THE DIVISOR IS ZERO. IF BEN04 FAILS THE DIVIDE WILL ABORT THINKING THAT THE DIVIDEND IS THE MOST NEGATIVE NUMBER. IF BEN16 FAILS THE DIVIDE WILL COMPLETE BUT R1 WILL CONTAIN 155776. IF BEN05 FAILS THE ALGORITHM WILL ABORT THRU STATE DVE.20. IF BEN04 (AFTER DIV.70) FAILS R0 WILL END UP WITH 177777 AND R1 WILL HAVE 177774. IF BEN03 FAILS R0 WILL END UP WITH

2273

20 AND R1 WILL HAVE 177774. IF BEN16 FAILS AFTER DVC.00 R0 WILL HAVE 2 BUT R1 WILL HAVE 177776.

SECTION 3

THE NEXT SECTION DIVIDES 6 BY 2 TO TEST BEN16\*DR0(1). A FAILURE WILL LEAVE THE REMAINDER (R1)=2 INSTEAD OF ZERO.

SECTION 4

THE NEXT SECTION DIVIDES 4 BY -2 TO TEST BEN15\*SR15(1). IF THIS FAILS R0 WILL CONTAIN 27777 AND R1 WILL CONTAIN 4.

SECTION 5

THE NEXT SECTION DIVIDES 1177777 BY 1 TO TEST BEN05\*DIV QUIT. IF THIS FAILS R0 WILL CONTAIN 177777.

SECTION 6

THE NEXT SECTION DIVIDES 1000000 B2 -2 TO TEST BEN05\*DIV QUIT. THIS SECTION WILL ONLY FAIL IF GRAJ E5 (Z2(0)\*LEFT SAVE (1)) IS BAD.

SECTION 7

THE NEXT SECTION DIVIDES 100000000000 BY 2 TO TEST BEN04\*NEGATIVE DIVIDEND.

SECTION 8

THE NEXT SECTION DIVIDED 177776 177777 BY -1 TO TEST BEN05\*DIV QUIT. THIS TEST WILL ONLY FAIL IF GRAJ E5(N(1)\*SR15(1)) IS BAD.

SECTION 9

THE NEXT SECTION DIVIDES -5 BY 2 TO ENSURE THAT THE REMAINDER IS STORED AS A NEGATIVE NUMBER.

SECTION 10

THE NEXT SECTION DIVIDES -5 BY -2 TO TEST STATES DVC.20,DVC.40, & DVC.60

SECTION 11

THE NEXT SECTION DIVIDES -2\*\*16 BY 2\*\*14 TO TEST STATE DVN.20

SECTION 12

THE NEXT SECTION DIVIDES 100 000200 BY -177 TO TEST STATES DVD.00 AND DVD.10.

2584 TEST 15 MTP\*DMO

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.

THE ONLY OTHER POSSIBLE FAILURES WOULD BE IN ROM STATES  
MTP.00 OR MTP.10.

NOTE: THIS TEST ONLY TESTS THE CPU FUNCTIONS OF THIS INSTRUCTION.  
THE MEMORY MANAGEMENT TEST VERIFIES THE INTERMODE TRANSFER.  
AS FAR AS THE CPU IS CONCERNED THERE IS NO DIFFERENCE  
BETWEEN MTPJ AND MTPD.

ROM FLOW-45,151,146,205

2627 TEST 16 MTP\*DM1

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.  
THIS WILL ONLY HAPPEN IF RACF E20(4) IS STUCK HIGH.

THIS TEST ENSURES STATE MTP.10 RELOADS THE  
DR IF THE DESTINATION IS R6 AND THAT IT PUTS THE PC IN THE  
DR IF THE DESTINATION FIELD IS R7.

ROM FLOW-45,151,146,111,155,312

2670 TEST 17 MFP\*DM0

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.  
IF ANYTHING ELSE FAILS, THEN A ROM STATE IS BAD.

ROM FLOW-46,304,250,222,300

2710 TEST 20 MFP\*DM2

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.  
IF FORK B FAILS EXECUTION WILL ALSO GO TO RSD.00 BUT THE DR  
WILL HAVE BEEN INCREMENTED. IF ANYTHING ELSE FAILS THEN  
STATE MFP.00 IS BAD.

ROM FLOW-2,175,66,250,222,300

2747 TEST 21 BPT

FORK A SHOULD NOT FAIL.  
IF THE TRAP VECTOR LOGIC FAILS THE TRAP VECTOR WOULD  
COME OUT TO BE 4.

THE ONLY OTHER FAILURE WOULD BE TRP.00.  
IF THIS STATE FAILS TO LOAD THE DR THE TRAP VECTOR WILL  
BE WHATEVER IS IN R3. IF IT FAILS TO LOAD THE BR THE OLD  
PS WILL FAIL TO BE STACKED.

2777 ALL THE LOGIC FOR (JMP+JSR)\*DM0 HAS BEEN TESTED.

2781 TEST 22 BIT TEST OF PIRQ REGISTER



IF ONE OF THE BLOCK LEVELS IS STUCK HIGH OR TMCB  
PS07(0)'S STUCK HIGH OR PDRD PRIORITY=0 IS STUCK HIGH,  
A PIRQ TRAP LOOP WILL OCCUR WHEN THAT PIR LEVEL IS ENABLED.

A COUNT PATTERN IS THEN RUN THRU THE REGISTER TO ENSURE THAT  
THE ENCODER FUNCTIONS PROPERLY.

2824 TEST 23 PIR LEVEL 1 INTERRUPT

IF BEN13 FAILS EXECUTION WOULD GO TO ONE OF THE FOLLOWING:  
PUP.00, BRK.20, OR SER.00.  
PUP.00 WOULD START THE POWER UP ROUTINE.  
BRK.20 WOULD CAUSE A TRAP TO ZERO.  
SER.00 WOULD PUT 4 IN THE SP AND PERFORM A RED ZONE TRAP.  
IF TMCB PIRQ DOES NOT GET TO DAPE OR IF DAPE TV05\*07 DOES NOT  
GO HIGH OR DOES NOT GET THRU TO THE ALU A TRAP TO 4 WILL OCCUR.  
IF THE INTERRUPT DOESN'T OCCUR AN ATTEMPT IS MADE TO ISOLATE THE  
FAILURE.

2900 TEST 24 PIR LEVEL 2 INTERRUPT

IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
THIS WILL ONLY HAPPEN IF TMCB E63(2) IS BAD.  
  
IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(2)  
IS BAD OR TMCB HONOR PIR2 IS BEING HELD HIGH.

2933 TEST 25 PIR LEVEL 3 INTERRUPT

IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
THIS WILL ONLY HAPPEN IF TMCB E63(3) IS BAD.  
  
IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(3)  
IS BAD OR TMCB HONOR PIR3 IS BEING HELD HIGH.

2966 TEST 26 PIR LEVEL 4 INTERRUPT

IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
THIS WILL ONLY HAPPEN IF TMCB E63(5) IS BAD.  
  
IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(5)  
IS BAD OR TMCA HONOR PIR 4 IS BEING HELD HIGH.

3000 TEST 27 PIR LEVEL 5 INTERRUPT

IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
THIS WILL ONLY HAPPEN IF TMCB E63(11) IS BAD.  
  
IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(11)  
IS BAD OR TMCA HONOR PIR 5 IS BEING HELD HIGH.

3035 TEST 30 PIR LEVEL 6 INTERRUPT

IF BEN13 FAILS EXECUTION WILL GO TO BRK.20.

THIS WILL ONLY HAPPEN IF EITHER TMCB E63(12)  
IS BAD, OR E61(1) IS BAD.

IF THE INTERRUPT DOES NOT OCCUR LEVEL 7 IS TRYED, TO TRY  
AND ISOLATE THE FAILURE BEFORE TMCB E55(9-8).

3077 TEST 31 PIR LEVEL 7 INTERRUPT

IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
THIS WILL ONLY HAPPEN IF TMCB E63(6) IS BAD.

IF THE INTERRUPT DOES NOT OCCUR THEN EITHER TMCB E70(6)

3083 IS BAD OR TMCA HONOR PIR7 IS BEING HELD HIGH.

3112 TEST 32 UNIBUS TIMEOUT

IF TMCC ABORT DOES NOT GO HIGH OR DOES NOT GET TO RACA  
OR IF RACA ZAP DOES NOT GO LOW THE PROCESSOR WILL NOT TRAP TO 4.

IF BEN06 FAILS THE STACKED PC WILL BE 160000 INSTEAD OF 1\$-2.

IF BEN 13 FAILS EITHER TMCC AERF(1) L IS NOT GOING LOW  
OR TMCB E53(11) IS BAD.

A TEST IS THEN MADE TO ENSURE THAT TMCC PRIORITY CLEAR GOES LOW.

3188

\*\*\*\*\*  
PERIPHERAL DETERMINATOR & INTERRUPT ENABLE ROUTINES  
\*\*\*\*\*

3189 THIS SECTION OF CODE TRYS TO FIND A DEVICE ON BR5 AND BR6.  
WHEN IT FINDS A DEVICE IT PUTS THE ADDRESS OF THAT DEVICES

3191 SUBROUTINE IN A LOCATION.

THE CODE TO INITIATE AN INTERRUPT SEQUENCE ON CERTAIN  
DEVICES IS ALSO HERE. WHEN A TEST REQUIRES AN INTERRUPT ON  
A CERTAIN LEVEL IT LOOKS AT INTERX TO DETERMINE IF A  
DEVICE IS AVAILABLE AND IF SO DOES A JSR TO THAT DEVICES  
INTERRUPT ENABLE ROUTINE.

3306 TEST 33 BR LEVEL 4 INTERRUPT

BEN 13 SHOULD NOT FAIL.  
IF THE INTERRUPT DOESN'T OCCUR AN ATTEMPT IS MADE TO  
ISOLATE THE FAILURE.

3370 TEST 34 BR LEVEL 5 INTERRUPT

THE ONLY POSSIBLE FAILURE IS THAT TMCA HONOR BR 5  
DOES NOT GO LOW OR TMCB E62(6) IS BAD.

3403 TEST 35 BR LEVEL 6 INTERRUPT

THE ONLY POSSIBLE FAILURE IS THAT TMCA HONOR BR 6 DOES NOT GO LOW  
OR TMCB E62(12) IS BAD.

3436 TEST 36 YELLOW ZONE TRAP

A YELLOW ZONE IS FIRST ATTEMPTED WITH THE SP AT 376.  
IF BEN 13 FAILS THE TRAP WILL NOT OCCUR.

IF THE PROCESSOR FAILS TO TRAP EITHER TMCD SL YEL IS  
NOT GOING HIGH OR TMCA HONOR SLY IS NOT GOING LOW  
OR TMCB E70(3) IS BAD OR BEN13 FAILED.

IF TMCC PRIORITY CLEAR DOES NOT GO LOW THE PROCESSOR WILL HANG  
UP IN A RED ZONE TRAP LOOP.

A JSR WITH A BAD SP IS THEN EXECUTED TO ENSURE TMCC  
KERNAL R6 GOES HIGH WHEN ENABLED BY 'STACK REFERENCE \* KERNAL MODE'.

IF THE TRAP WORKS TESTS WILL BE PERFORMED TO ENSURE ALL THE  
APPROPRIATE CONDITIONS DISABLE THE TRAP EXCEPT  
THE PRIORITY ARBITRATOR.

3527 TEST 37 ROM FIELD CHECK OF PC MANIPULATOR STATES

THIS TEST EXECUTES THE MACHINE STATES THAT MANIPULATE  
THE PC TO ENSURE THAT THE PCB ROM FIELD OR DRX ROM FIELD  
OR SRX ROM FIELD OF THESE STATES IS FUNCTIONAL.  
THESE STATES ARE S13.00, S45.00, MTP.10, D45.80, D45.90,  
D45.00, AND D45.01.

3618 TEST 40 RED ZONE TRAP

A RED ZONE TRAP IS FIRST ATTEMPTED WITH THE SP AT 336.  
IF BEN13 FAILS EXECUTION WILL GO TO EITHER BRK.80 OR BRK.20  
OR PUP.00.

3623

BRK.80 WILL CAUSE THE OLD PSW AND PC TO BE STACKED ON THE  
OLD STACK INSTEAD OF LOCATIONS 2 AND 0.  
BRK.20 WILL MAKE IT LOOK LIKE THE RED ZONE FAILED.  
PUP.00 WILL CAUSE A TRAP TO LOCATION 24.

IF THE PROCESSOR FAILS TO TRAP EITHER TMCD SL RED IS  
NOT GOING LOW OR TMCC ABORT IS NOT GOING LOW.

IF UBCB ABORT RESTART FAILS TO GO LOW OR E10(13)  
IS BAD THE PROCESSOR WILL HANG IN THE PAUSE STATE.

3726 TEST 41 BIT TEST OF STACK LIMIT REGISTER

FIRST A 125252 AND 52525 PATTERN IS PUT IN THE REGISTER TO ENSURE  
THAT THE REGISTER DOESN'T HAVE ANY STUCK BITS AND THAT  
THE DMUX SELECT AND INPUT LINES WORK.

3731

IF SCCE SL ADRS DOES NOT GET TO TMLD OR IF TMCD E28 OR E'4 IS BAD THE BR WILL BE SELECTED. THE PB REGISTER IS LOADED WITH 200 SO IF TMCD LO BYTE EN DOES NOT GO LOW AN ERROR WILL BE DETECTED.

3782 TEST 42 SL REGISTER COMPARATOR TEST 1

3785 THIS TEST RUNS A HIGH BYTE COUNT PATTERN THRU THE BUS ADDRESS MUX FOR EACH PATTERN OF THE STACK LIMIT REGISTER. FOR EACH PATTERN OF ADDRESSES THERE WILL BE ONE YEL TRAP AT THE ADDRESS CORRESPONDING TO THE SL REG+340 AND A RED TRAP AT EVERY ADDRESS BELOW THIS. THIS TEST ONLY TESTS ADDRESSES UP TO THE I/O PAGE. THE I/O PAGE ADDRESSES WILL BE TESTED SEPARATELY WITH MEMORY MANAGEMENT ENABLED AND THE I/O PAGE MAPPED INTO RESIDENT MEMORY

THE FOLLOWING ARE THE TYPES OF ERRORS THAT CAN OCCUR IN THIS TEST:

TYPE	DESCRIPTION
0	RED ZONE TRAP ON YELLOW ZONE ADDRESS
2	RED ZONE TRAP ON LEGAL ADDRESS
4	YELLOW ZONE TRAP ON RED ZONE ADDRESS
6	YELLOW ZONE TRAP ON LEGAL ADDRESS
10	NO TRAP ON RED ZONE ADDRESS
12	NO TRAP ON YELLOW ZONE ADDRESS

THE LOW BYTE ADDRESS IN THE STACK POINTER IS ALWAYS 340 AND WILL NOT BE TYPED ON AN ERROR.

3808 NOTE: IF THE LOOP ON ERROR SWITCH IS UP (SWITCH 9) THE TEST WILL LOOP ON THE FIRST ERROR WITH NO ERROR TYPEOUT. OTHERWISE ALL ERRORS WILL BE RECORDED IN A TABLE AND TYPED OUT AT THE END OF THE TEST. IF SWITCH 3 (DISABLE MEMORY MANAGEMENT TESTS) IS NOT ON, THIS TEST IS SKIPPED AND TEST 70 WILL EXECUTE.

3904 TEST 43 ODD ADDRESS ERROR

BEN 13 SHOULD NOT FAIL. IF THE PROCESSOR FAILS TO TRAP IN ALL SECTIONS EITHER TMCC ODD ADRS ERR IS NOT GOING LOW OR TMCC BUS ERROR IS NOT GOING LOW.

EACH TYPE OF ODD ADDRESS ERROR IS TESTED INDIVIDUALLY TO ALLOW MAXIMUM ISOLATION.

NOTE: AN ODD ADDRESS ON 'KERNEL DATI' CANNOT BE TESTED. THIS SIGNAL COMES UP WHEN A TRAP VECTOR IS READ IN FROM THE BUS.

4002 TEST 44 ILLEGAL INSTRUCTIONS  
THIS TEST ENSURES THAT ILLEGAL OP CODES TRAP TO LOCATION 10. ONLY THOSE OP CODES THAT HAVE A SINGLE BIT THAT DISTINGUISHES THEM FROM A LEGAL INSTRUCTION WILL BE TESTED.

4039 TEST 45 1 BIT TRAP

IF BEN 13 FAILS EXECUTION WOULD GO TO BRK.20.  
THIS WOULD LOOK LIKE THE TRAP DIDN'T OCCUR.

IF THE TRAP DOESN'T OCCUR THEN EITHER PDRD PS04(1) DOES NOT GET  
TO TMCB AS A HIGH OR IT DOES NOT GET TO TMCB E51(10)  
AS A LOW OR E51 IS BAD OR IRCD RTT DOES NOT GET TO TMCB AS A HIGH.

THIS TEST ALSO CHECKS THAT PS<08> SET WILL INHIBIT A T BIT TRAP IF  
THIS IS A KB11-E.

4058 TEST 46 T BIT TRAP AND RTT

IF THE INSTRUCTION AFTER THE RTT DOES NOT GET EXECUTED THEN  
EITHER IRCD RTT DOES NOT GO LOW OR IT DOES NOT GET TO  
TMCB E74(11) OR TMCB E74 IS BAD.

4174 TEST 47 PRIORITY ARBITRATION

THIS TEST ASSURES THAT EACH NECESSARY INPUT TO AN HONOR FLAG  
CAN DISABLE THAT FLAG.

EACH SECTION WILL PERFORM A SETUP SO THAT A TIGHT ERROR LOOP  
CAN BE OBTAINED.

THE FOLLOWING IS A TABLE OF CONTENTS OF THIS TEST:

SECTION NUMBER	LEVEL UNDER TEST	DISABLING FUNCTION
1	PIR 1	BR 4
2	PIR 1	SL YELLOW
3	PIR 2	SL YELLOW
4	PIR 3	SL YELLOW
5	BR 4	PIR 4
6	BR 4	PIR 5
7	BR 4	BR 5
8	BR 4	PIR 6
9	BR 4	PIR 7
10	PIR 4	BR 5
11	PIR 4	BR 6
12	PIR 4	SL YELLOW
13	BR 5	PIR 5
14	BR 5	PIR 6
15	BR 5	PIR 7
16	PIR 5	BR 6
17	PIR 5	SL YELLOW
18	BR 6	PIR 6
19	BR 6	PIR 7
20	PIR 6	SL YELLOW
21	PIR 7	SL YELLOW

4646 TEST 50 GPR SET 1 SELECT TEST

THIS TEST FIRST ENSURES THAT PSW BIT 11 SETS AND CLEARS.

- 4649 IT THEN ENSURES THAT GRAC GDREG SET 1 AND GSREG SET 1 GOES HIGH FOR THE MUX SELECTS LL, LH, AND HL. MUX SELECT HH WILL BE TESTED IN SUPERVISOR MODE.
- 4748 TEST 51 REGISTER SET 1 STUCK BIT TEST  
THIS TEST ENSURES THAT ALL BITS IN GPR'S R10 THRU R15 WORK
- 4807 TEST 52 PSW HIGH BYTE BIT TEST  
THIS TEST ENSURES THAT THE PRESENT AND PREVIOUS MODE BITS OF THE PSW CAN BE SET AND CLEARED AND THAT THEY ARE NOT STUCK TOGETHER.
- 4837 TEST 53 SP SELECTION TEST IN SUPER AND USER MODE  
THIS TEST ENSURES THAT THE CORRECT STACK POINTERS ARE SELECTED IN SUPERVISOR AND USER MODE
- 4895 TEST 54 SUPER AND USER SP BIT TEST  
THIS TEST ENSURES THAT THE SUPERVISOR AND USER STACK POINTERS DON'T HAVE ANY STUCK BITS.
- 4933 TEST 55 MTP\*DMO\*DF6\*PREVIOUS MODE(SUPER\*USER)  
THIS TEST ENSURES THAT THE CORRECT SP'S ARE SELECTED WHEN EXECUTING A MTP WITH DIFFERENT PREVIOUS MODE BITS SELECTED.
- 4991 TEST 56 MFP\*DMO\*DF6\*PREVIOUS MODE SUPER  
THE ONLY POSSIBLE WAY THIS TEST CAN FAIL IS THAT IRCC DMO (MFP+MTP) DOES NOT GO HIGH ON MFP. THIS WILL ONLY HAPPEN IF THE IR DECODE ROM HAS A BAD FIELD (R(MFP+MTP)).
- 5023 TEST 57 UPAD 7 IN USER MODE  
THIS TEST ENSURES THAT A UPAD 7(OCCURS IN RTI) CAUSES THE USER STACK POINTER TO BE USED TO FETCH THE NEW PS AND PC.
- 5027 IF PDRD PS14(1) DOES NOT GET TO THE GSAM(ON GRAC) THE TEST WILL BLOW UP.
- 5074 TEST 60 SPL\*SUPERVISOR MODE  
THIS TEST ENSURES THAT SPL DOES NOT LOAD THE PSW IN SUPER+USER MODE.
- 5090 TEST 61 PSW CLOCKING TEST  
THIS TEST ENSURES THAT ALL THE BITS IN THE PSW GET LOADED WHEN THE FOLLOWING SIGNALS ARE TRUE:  
1) LOAD PS\*KERNEL MODE, AND 2) LOAD PS\*KERNEL DATI.

IT ALSO ENSURES THAT THE PRESET LOGIC ON BITS 11, 12, 13, 14, AND 15 FUNCTIONS PROPERLY.

FOLLOWING IS A TABLE TO DESCRIBE THE PSW FOR EACH SECTION

SECTION	PSW AT START	PSW ON STK(OR VECTOR)	EXPEC PSW
1	000XXX	174XXX	174XXX
2	174XXX	000XXX	174XXX
3	040XXX	134XXX	174XXX
4	144XXX	000XXX	030XXX
5	030XXX	000XXX	000XXX

5196 TEST 62 ILLEGAL HALT  
THIS TEST ENSURES THAT A HALT IN SUPER OR USER MODE WILL TRAP TO LOCATION 4.

IF BFN6 FAILS EXECUTION WOULD GO TO FET.04 WHICH WOULD CAUSE THE HALT TO LOOK LIKE A NOP.  
IF TMCE SET CONF GOES LOW THE PROCESSOR WILL HALT AT LOCALLOCATION 1\$.  
THE CPU ERROR REGISTER BIT 7 IS ALSO TESTED HERE.

5229 TEST 63 WAIT

THIS TEST ENSURES THAT THE WAIT INSTRUCTION WORKS PROPERLY. IT FIRST EXECUTES WITH A LEVEL 4 INTERRUPT. THEN THE T BIT IS ENABLED TO ENSURE THAT THE INTERRUPT OCCURS AND NOT THE T BIT TRAP.

5250 TEST 64 CHECK MFPT INSTRUCTION (KB-11E/EM ONLY)

6011 TEST 65 OPERATOR INTERVENTION TEST

THIS TEST ENSURES THAT THE RESET AND WAIT FLOWS PUT RO AND THE PC IN THE LIGHTS. THE TEST IS ONLY EXECUTED ON PASS 1 AND CAN BE DISABLED ALTOGETHER WITH SWITCH 0.

THE RESET LOOP IS STOPED BY CHANGING THE POSITION OF SWITCH 7.

THE WAIT IS EXITED BY TYPING A CHARACTER ON THE TERMINAL.

6074

\*\*\*\*\*  
END OF PASS ROUTINE  
\*\*\*\*\*

6076 INCREMENT THE PASS NUMBER (\$PASS)  
INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM  
TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'  
WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS  
IF SW12=1 INHIBIT TRACE TRAP  
IF THERES A MONITOR GO TO IT  
IF THERE ISN'T JUMP TO LOOP

6146 \*\*\*\*\*  
SPURIOUS ERROR HANDLER  
\*\*\*\*\*

6147 THIS ROUTINE IS ENTERED BY AN UNEXPECTED TRAP TO 4 OR 114.  
IF SWITCH 13 IS OFF, AN ERROR MESSAGE, THE ERROR REGISTER,  
THE ERROR PC, AND THE TEST NUMBER ARE TYPED OUT.  
IF SWITCH 13 IS ON, ONLY THE ERROR MESSAGE WILL BE TYPED.

6211 \*\*\*\*\*  
SCOPE HANDLER ROUTINE  
\*\*\*\*\*

6213 THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT  
AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)

6215 AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>  
THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
SW14=1 LOOP ON TEST  
SW11=1 INHIBIT ITERATIONS  
SW09=1 LOOP ON ERROR  
SW08=1 LOOP ON TEST IN SWR<7:0>  
CALL SCOPE ;SCOPE=IOT

6277 \*\*\*\*\*  
ERROR HANDLER ROUTINE  
\*\*\*\*\*

6279 THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,  
SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL  
AND GO TO ETYPDM ON ERROR  
THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:  
SW15=1 HALT ON ERROR  
SW13=1 INHIBIT ERROR TYPEOUTS  
SW10=1 BELL ON ERROR  
SW09=1 LOOP ON ERROR  
CALL ERROR N ;:ERROR=EMT AND N-ERROR ITEM NUMBER

6328 \*\*\*\*\*  
ERROR MESSAGE TYPEOUT ROUTINE  
\*\*\*\*\*

6329 THIS ROUTINE USES THE 'ITEM CONTROL BYTE' (\$ITEMB) TO DETERMINE WHICH  
ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' (\$ERRTB),  
AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

6371 \*\*\*\*\*  
STACK LIMIT TEST TYPE OUT ROUTINE  
\*\*\*\*\*



6372 THIS ROUTINE TYPES THE ADDRESS AND STACK LIMIT REGISTER  
VALUES THAT FAILED IN TEST 153 OR 201 IN OCTAL AND BINARY.

6489

\*\*\*\*\*  
MONITOR RESTORE ROUTINE  
\*\*\*\*\*

6490 THIS ROUTINE IS ENTERED BY TYPING A CHARACTER ON THE KEYBOARD  
IF THE CHARACTER IS NOT A CTRL C EXECUTION IS RETURNED TO THE  
TEST FOLLOWING THE ONE THAT WAS INTERRUPTED.  
IF IT IS A CTRL C THE MONITOR IS RESTORED AND THE  
PROCESSOR HALTS.

6527

\*\*\*\*\*  
CHECK TEST SEQUENCE ROUTINE  
\*\*\*\*\*

6528 THIS ROUTINE IS CALLED IN THE SCOPE ROUTINE. IT VERIFYS  
THAT A TEST HAS NOT BEEN SKIPPED.

6588

\*\*\*\*\*  
TYPE ROUTINE  
\*\*\*\*\*

6590 ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.  
THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.  
NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.

6593 NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.  
NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.

CALL:

1) USING A TRAP INSTRUCTION  
TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING  
OR

TYPE  
MESADR

2) USING A JSR INSTRUCTION

MOV PS,-(SP) ;:PUSH PROCESSOR STATUS WORD ON THE STACK  
JSR PC,\$TYPE ;:CALL TYPE ROUTINE  
MESADDR ;:FIRST ADRESS OF MESSAGE

6661

\*\*\*\*\*  
BINARY TO OCTAL (ASCII) AND TYPE  
\*\*\*\*\*

6663 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT  
OCTAL (ASCII) NUMBER AND TYPE IT.  
\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE  
CALL:

MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
TYPOS ;:CALL FOR TYPEOUT

.BYTE N ;:N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE  
.BYTE M ;:M=1 OR 0  
;:1=TYPE LEADING ZEROS  
;:0=SUPPRESS LEADING ZEROS

\$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST  
\$TYPOS OR \$TYPOC

CALL:  
MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
TYPON ;:CALL FOR TYPEOUT

\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

CALL:  
MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
TYPOC ;:CALL FOR TYPEOUT

6739 \*\*\*\*\*  
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE  
\*\*\*\*\*

6741 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT  
SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE  
NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED  
BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE  
REPLACED WITH SPACES.

CALL:  
MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK  
TYPDS ;:GO TO THE ROUTINE

6807 \*\*\*\*\*  
TRAP DECODER  
\*\*\*\*\*

6809 THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION  
AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS  
OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
GO TO THAT ROUTINE.

6822 \*\*\*\*\*  
TRAP TABLE  
\*\*\*\*\*

6824 THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
BY THE 'TRAP' INSTRUCTION.

6837 \*\*\*\*\*  
POWER DOWN AND UP ROUTINES  
\*\*\*\*\*

1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710

000001  
177400

.TITLE CEKBBE0 11/70-74MP CPU #2  
:\*COPYRIGHT (C) 1975,1979  
:\*DIGITAL EQUIPMENT CORP.  
:\*MAYNARD, MASS. 01754  
:\*  
:\*PROGRAM BY DONALD W. MONROE  
:\*  
:\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
:\*PACKAGE (MAINDEC-11-DZQAC-A5).  
:\*  
\$TN-1 \$SWR-177400

1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733

```
.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH      USE
:*      -----
:*      15          HALT ON ERROR
:*      14          LOOP ON TEST
:*      13          INHIBIT ERROR TYPEOUTS
:*      12          INHIBIT TRACE TRAP
:*      11          INHIBIT ITERATIONS
:*      10          BELL ON ERROR
:*      9           LOOP ON ERROR
:*      8           LOOP ON TEST IN SWR<7:0>
:*      7           NOT USED
:*      6           SKIP BR6 TEST
:*      5           SKIP BR5 TEST
:*      4           SKIP BR4 TEST
:*      3           NOT USED
:*      2           NOT USED
:*      1           NOT USED
:*      0           DISABLE OPERATOR INTERVENTION TEST
```

```
1734  
1735 .SBTTL BASIC DEFINITIONS  
1736  
1737 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***  
1738 001100 STACK= 1100 ;:FIRST ADDRESS OF THE STACK  
1739 001100 KERSTK= STACK ;:KERNEL STACK  
1740 000700 SUPSTK= STACK-200 ;:SUPERVISOR STACK  
1741 000600 USESTK= STACK-300 ;:USER STACK  
1742 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL  
1743 .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL  
1744 177776 PS= 177776 ;:PROCESSOR STATUS WORD  
1745 .EQUIV PS,PSW  
1746 177774 STKLMT= 177774 ;:STACK LIMIT REGISTER  
1747 177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER  
1748 177570 SWR= 177570 ;:SWITCH REGISTER  
1749 177570 DISPLAY=SWR  
1750  
1751 ;*MISCELLANEOUS DEFINITIONS  
1752 000011 HT= 11 ;:CODE FOR HORIZONTAL TAB  
1753 000012 LF= 12 ;:CODE LINE FEED  
1754 000015 CR= 15 ;:CODE CARRIAGE RETURN  
1755 000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED  
1756  
1757 ;*GENERAL PURPOSE REGISTER DEFINITIONS  
1758 000000 R0= %0 ;:GENERAL REGISTER  
1759 000001 R1= %1 ;:GENERAL REGISTER  
1760 000002 R2= %2 ;:GENERAL REGISTER  
1761 000003 R3= %3 ;:GENERAL REGISTER  
1762 000004 R4= %4 ;:GENERAL REGISTER  
1763 000005 R5= %5 ;:GENERAL REGISTER  
1764 000006 R6= %6 ;:GENERAL REGISTER  
1765 000007 R7= %7 ;:GENERAL REGISTER  
1766 .EQUIV R0,R10 ;:GENERAL REGISTER  
1767 .EQUIV R1,R11 ;:GENERAL REGISTER  
1768 .EQUIV R2,R12 ;:GENERAL REGISTER  
1769 .EQUIV R3,R13 ;:GENERAL REGISTER  
1770 .EQUIV R4,R14 ;:GENERAL REGISTER  
1771 .EQUIV R5,R15 ;:GENERAL REGISTER  
1772 000006 SP=%6 ;:STACK POINTER  
1773 .EQUIV SP,KSP ;:KERNEL STACK POINTER  
1774 .EQUIV SP,SSP ;:SUPERVISOR STACK POINTER  
1775 .EQUIV SP,USP ;:USER STACK POINTER  
1776 000007 PC=%7 ;:PROGRAM COUNTER  
1777  
1778 ;*PRIORITY LEVEL DEFINITIONS  
1779 000000 PR0= 0 ;:PRIORITY LEVEL 0  
1780 000040 PR1= 40 ;:PRIORITY LEVEL 1  
1781 000100 PR2= 100 ;:PRIORITY LEVEL 2  
1782 000140 PR3= 140 ;:PRIORITY LEVEL 3  
1783 000200 PR4= 200 ;:PRIORITY LEVEL 4  
1784 000240 PR5= 240 ;:PRIORITY LEVEL 5  
1785 000300 PR6= 300 ;:PRIORITY LEVEL 6  
1786 000340 PR7= 340 ;:PRIORITY LEVEL 7  
1787  
1788 ;*'SWITCH REGISTER' SWITCH DEFINITIONS  
1789 100000 SW15= 100000
```

1790 040000  
1791 020000  
1792 010000  
1793 004000  
1794 002000  
1795 001000  
1796 000400  
1797 000200  
1798 000100  
1799 000040  
1800 000020  
1801 000010  
1802 000004  
1803 000002  
1804 000001  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817 100000  
1818 040000  
1819 020000  
1820 010000  
1821 004000  
1822 002000  
1823 001000  
1824 000400  
1825 000200  
1826 000100  
1827 000040  
1828 000020  
1829 000010  
1830 000004  
1831 000002  
1832 000001  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845 000004

SW14= 40000  
SW13= 20000  
SW12= 10000  
SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

;\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

;\*BASIC 'CPU' TRAP VECTOR ADDRESSES

ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS

1846	000010	RESVEC= 10	::RESERVED AND ILLEGAL INSTRUCTIONS
1847	000014	TBITVEC=14	::'T' BIT
1848	000014	TRTVEC= 14	::TRACE TRAP
1849	000014	BPTVEC= 14	::BREAKPOINT TRAP (BPT)
1850	000020	IOTVEC= 20	::INPUT/OUTPUT TRAP (IOT) **SCOPE**
1851	000024	PWRVEC= 24	::POWER FAIL
1852	000030	EMTVEC= 30	::EMULATOR TRAP (EMT) **ERROR**
1853	000034	TRAPVEC=34	::'TRAP' TRAP
1854	000060	TKVEC= 60	::TTY KEYBOARD VECTOR
1855	000064	TPVEC= 64	::TTY PRINTER VECTOR
1856	000114	CACHVEC=114	::CACHE ERROR INTERRUPT VECTOR
1857	000240	PIRQVEC=240	::PROGRAM INTERRUPT REQUEST VECTOR
1858	000250	MMVEC= 250	::MEMORY MANAGEMENT VECTOR

.SBTTL CACHE REGISTER DEFINITIONS

1863	177740	LOADRS = 177740	::LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
1864	177742	HIADRS = 177742	::UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
1865	177744	MEMERR = 177744	::CACHE ERROR REGISTER
1866	177746	CONTRL = 177746	::MEMORY CONTROL REGISTER
1867	177750	MAINT = 177750	::MEMORY MAINTENENCE REGISTER
1868	177752	HITMIS = 177752	::HIT MISS REGISTER '1' IMPLIES HIT IN CACHE

.SBTTL CPU REGISTER DEFINITIONS

1874	177760	SIZELO = 177760	::MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
1875	177762	SIZEHI = 177762	::TO GET TO THE LAST 32 WORDS OF MEMORY
1876			::HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
1877			::CURRENTLY ALL ZERO
1878	177764	SYSTID = 177764	::SYSTEM ID REGISTER
1879	177766	CPUERR = 177766	::CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
1880			::THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;\*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

1890	177572	MMR0= 177572
1891	177574	MMR1= 177574
1892	177576	MMR2= 177576
1893	172516	MMR3= 172516
1894		.EQUIV MMR0,SR0
1895		.EQUIV MMR1,SR1
1896		.EQUIV MMR2,SR2
1897		.EQUIV MMR3,SR3

;\*USER 'I' PAGE DESCRIPTOR REGISTERS

1901	177600	UIPDRO= 177600
------	--------	----------------

1902	177602	UIPDR1= 177602
1903	177604	UIPDR2= 177604
1904	177606	UIPDR3= 177606
1905	177610	UIPDR4= 177610
1906	177612	UIPDR5= 177612
1907	177614	UIPDR6= 177614
1908	177616	UIPDR7= 177616
1909		
1910		;*USER 'D' PAGE DESCRIPTOR REGISTORS
1911		
1912	177620	UDPDR0= 177620
1913	177622	UDPDR1= 177622
1914	177624	UDPDR2= 177624
1915	177626	UDPDR3= 177626
1916	177630	UDPDR4= 177630
1917	177632	UDPDR5= 177632
1918	177634	UDPDR6= 177634
1919	177636	UDPDR7= 177636
1920		
1921		;*USER 'I' PAGE ADDRESS REGISTERS
1922		
1923	177640	UIPAR0= 177640
1924	177642	UIPAR1= 177642
1925	177644	UIPAR2= 177644
1926	177646	UIPAR3= 177646
1927	177650	UIPAR4= 177650
1928	177652	UIPAR5= 177652
1929	177654	UIPAR6= 177654
1930	177656	UIPAR7= 177656
1931		
1932		;*USER 'D' PAGE ADDRESS REGISTERS
1933		
1934	177660	UDPAR0= 177660
1935	177662	UDPAR1= 177662
1936	177664	UDPAR2= 177664
1937	177666	UDPAR3= 177666
1938	177670	UDPAR4= 177670
1939	177672	UDPAR5= 177672
1940	177674	UDPAR6= 177674
1941	177676	UDPAR7= 177676
1942		
1943		;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS
1944		
1945	172200	SIPDR0= 172200
1946	172202	SIPDR1= 172202
1947	172204	SIPDR2= 172204
1948	172206	SIPDR3= 172206
1949	172210	SIPDR4= 172210
1950	172212	SIPDR5= 172212
1951	172214	SIPDR6= 172214
1952	172216	SIPDR7= 172216
1953		
1954		;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS
1955		
1956	172220	SDPDR0= 172220
1957	172222	SDPDR1= 172222



1958	172224	SDPDR2= 172224
1959	172226	SDPDR3= 172226
1960	172230	SDPDR4= 172230
1961	172232	SDPDR5= 172232
1962	172234	SDPDR6= 172234
1963	172236	SDPDR7= 172236
1964		
1965		;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS
1966		
1967	172240	SIPAR0= 172240
1968	172242	SIPAR1= 172242
1969	172244	SIPAR2= 172244
1970	172246	SIPAR3= 172246
1971	172250	SIPAR4= 172250
1972	172252	SIPAR5= 172252
1973	172254	SIPAR6= 172254
1974	172256	SIPAR7= 172256
1975		
1976		;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS
1977		
1978	172260	SDPAR0= 172260
1979	172262	SDPAR1= 172262
1980	172264	SDPAR2= 172264
1981	172266	SDPAR3= 172266
1982	172270	SDPAR4= 172270
1983	172272	SDPAR5= 172272
1984	172274	SDPAR6= 172274
1985	172276	SDPAR7= 172276
1986		
1987		;*KERNEL 'i' PAGE DESCRIPTOR REGISTERS
1988		
1989	172300	KIPDR0= 172300
1990	172302	KIPDR1= 172302
1991	172304	KIPDR2= 172304
1992	172306	KIPDR3= 172306
1993	172310	KIPDR4= 172310
1994	172312	KIPDR5= 172312
1995	172314	KIPDR6= 172314
1996	172316	KIPDR7= 172316
1997		
1998		;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS
1999		
2000	172320	KDPDR0= 172320
2001	172322	KDPDR1= 172322
2002	172324	KDPDR2= 172324
2003	172326	KDPDR3= 172326
2004	172330	KDPDR4= 172330
2005	172332	KDPDR5= 172332
2006	172334	KDPDR6= 172334
2007	172336	KDPDR7= 172336
2008		
2009		;*KERNEL 'I' PAGE ADDRESS REGISTERS
2010		
2011	172340	KIPAR0= 172340
2012	172342	KIPAR1= 172342
2013	172344	KIPAR2= 172344

2014	172346	KIPAR3= 172346
2015	172350	KIPAR4= 172350
2016	172352	KIPAR5= 172352
2017	172354	KIPAR6= 172354
2018	172356	KIPAR7= 172356

;\*KERNEL 'D' PAGE ADDRESS REGISTERS

2022	172360	KDPAR0= 172360
2023	172362	KDPAR1= 172362
2024	172364	KDPAR2= 172364
2025	172366	KDPAR3= 172366
2026	172370	KDPAR4= 172370
2027	172372	KDPAR5= 172372
2028	172374	KDPAR6= 172374
2029	172376	KDPAR7= 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

;\*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'  
;\*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

2041	170200	MAPL00 = 170200
2042	170202	MAPH00 = 170202
2043	170204	MAPL01 = 170204
2044	170206	MAPH01 = 170206
2045	170210	MAPL02 = 170210
2046	170212	MAPH02 = 170212
2047	170214	MAPL03 = 170214
2048	170216	MAPH03 = 170216
2049	170220	MAPL04 = 170220
2050	170222	MAPH04 = 170222
2051	170224	MAPL05 = 170224
2052	170226	MAPH05 = 170226
2053	170230	MAPL06 = 170230
2054	170232	MAPH06 = 170232
2055	170234	MAPL07 = 170234
2056	170236	MAPH07 = 170236
2057	170240	MAPL10 = 170240
2058	170242	MAPH10 = 170242
2059	170244	MAPL11 = 170244
2060	170246	MAPH11 = 170246
2061	170250	MAPL12 = 170250
2062	170252	MAPH12 = 170252
2063	170254	MAPL13 = 170254
2064	170256	MAPH13 = 170256
2065	170260	MAPL14 = 170260
2066	170262	MAPH14 = 170262
2067	170264	MAPL15 = 170264
2068	170266	MAPH15 = 170266
2069	170270	MAPL16 = 170270

2070	170272	MAPH16 = 170272
2071	170274	MAPL17 = 170274
2072	170276	MAPH17 = 170276
2073	170300	MAPL20 = 170300
2074	170302	MAPH20 = 170302
2075	170304	MAPL21 = 170304
2076	170306	MAPH21 = 170306
2077	170310	MAPL22 = 170310
2078	170312	MAPH22 = 170312
2079	170314	MAPL23 = 170314
2080	170316	MAPH23 = 170316
2081	170320	MAPL24 = 170320
2082	170320	MAPH24 = 170320
2083	170324	MAPL25 = 170324
2084	170326	MAPH25 = 170326
2085	170330	MAPL26 = 170330
2086	170332	MAPH26 = 170332
2087	170334	MAPL27 = 170334
2088	170336	MAPH27 = 170336
2089	170340	MAPL30 = 170340
2090	170342	MAPH30 = 170342
2091	170344	MAPL31 = 170344
2092	170346	MAPH31 = 170346
2093	170350	MAPL32 = 170350
2094	170352	MAPH32 = 170352
2095	170354	MAPL33 = 170354
2096	170356	MAPH33 = 170356
2097	170360	MAPL34 = 170360
2098	170362	MAPH34 = 170362
2099	170364	MAPL35 = 170364
2100	170366	MAPH35 = 170366
2101	170370	MAPL36 = 170370
2102	170372	MAPH36 = 170372
2103	170374	MAPL37 = 170374
2104	170376	MAPH37 = 170376
2105		.EQUIV MAPL00,MAPL0
2106		.EQUIV MAPH00,MAPH0
2107		.EQUIV MAPL01,MAPL1
2108		.EQUIV MAPH01,MAPH1
2109		.EQUIV MAPL02,MAPL2
2110		.EQUIV MAPH02,MAPH2
2111		.EQUIV MAPL03,MAPL3
2112		.EQUIV MAPH03,MAPH3
2113		.EQUIV MAPL04,MAPL4
2114		.EQUIV MAPH04,MAPH4
2115		.EQUIV MAPL05,MAPL5
2116		.EQUIV MAPH05,MAPH5
2117		.EQUIV MAPL06,MAPL6
2118		.EQUIV MAPH06,MAPH6
2119		.EQUIV MAPL07,MAPL7
2120		.EQUIV MAPH07,MAPH7
2121		
2122		
2123		
2124		
2125	172544	PLKC=172544

```
2126  
2127  
2128 .SBTTL TRAP CATCHER  
2129  
2130 000000 . =0  
2131 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '.+2,HALT'  
2132 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS  
2133 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS  
2134  
2135 .SBTTL STARTING ADDRESS(ES)  
2136 000200 . =200  
2137  
2138 000200 000137 004742 JMP @#START ;: JUMP TO STARTING ADDRESS OF PROGRAM  
2139 ;:*****  
2140  
2141 .SBTTL ACT11 HOOKS  
2142  
2143 ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11  
2144 ;*  
2145 ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL  
2146 ;*END OF THE PROGRAM.  
2147 ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS  
2148 ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS  
2149 ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:  
2150 ;*  
2151 ;* BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING  
2152 ;* =0 NO POWER FAIL DESIRED  
2153 ;*  
2154 ;* BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT  
2155 ;* =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT  
2156 ;*  
2157 ;* BITS 13-0 MUST BE ZERO'S  
2158 ;*  
2159 000204 $SVPC= . ;: SAVE LOCATION COUNTER  
2160 000046 . =46 ;: SET LOCATION COUNTER  
2161 000046 .WORD $ENDAD ;: SET LOC.46 TO ADDRESS $ENDAD  
2162 000052 . =52 ;: SET LOCATION COUNTER  
2163 000052 .WORD 0 ;: SET LOC.52 TO ZERO  
2164 000204 .-$SVPC ;: RESTORE LOCATION COUNTER
```

```
2165 ;:*****
2166
2167 .SBTTL COMMON TAGS
2168
2169 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
2170 ;*USED IN THE PROGRAM.
2171
2172 001100 . =1100
2173
2174 001100 $CMTAG: ;:START OF COMMON TAGS
2175 001100 000000 $PASS: .WORD 0 ;:CONTAINS PASS COUNT
2176 001102 000 $STSTM: .BYTE 0 ;:CONTAINS THE TEST NUMBER
2177 001103 000 $ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG
2178 001104 000000 $ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT
2179 001106 000000 $LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP
2180 001110 000000 $LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS
2181 001112 000000 $ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED
2182 001114 000 $ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE
2183 001115 001 $ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST
2184 001116 000000 $ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION
2185 001120 000000 $GDADR: .WORD 0 ;:CONTAINS OF 'GOOD' DATA
2186 001122 000000 $BDADR: .WORD 0 ;:CONTAINS OF 'BAD' DATA
2187 001124 000000 $GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA
2188 001126 000000 $BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA
2189 001130 000000 000000 000000 .WORD 0,0,0 ;:RESERVED--NOT TO BE USED
2190 001136 177560 $TKS: 177560 ;:TTY KBD STATUS
2191 001140 177562 $TKB: 177562 ;:TTY KBD BUFFER
2192 001142 177564 $TPS: 177564 ;:TTY PRINTER STATUS REG.
2193 001144 177566 $TPB: 177566 ;:TTY PRINTER BUFFER REG.
2194 001146 000 $NULL: .BYTE 0 ;:CONTAINS NULL CHARACTER FOR FILLS
2195 001147 002 $FILLS: .BYTE 2 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
2196 001150 012 $FILLC: .BYTE 12 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
2197 001151 000 $TPFLG: .BYTE 0 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
2198 001152 000000 $REGAD: .WORD 0 ;:CONTAINS THE FROM
2199 ;:WHICH ($REGO) WAS OBTAINED
2200 001154 000000 $REGO: .WORD 0 ;:CONTAINS (($REGAD)+0)
2201 001156 000000 $REG1: .WORD 0 ;:CONTAINS (($REGAD)+2)
2202 001160 000000 $REG2: .WORD 0 ;:CONTAINS (($REGAD)+4)
2203 001162 000000 $TMP0: .WORD 0 ;:USER DEFINED
2204 001164 000000 $TMP1: .WORD 0 ;:USER DEFINED
2205 001166 000000 $TMP2: .WORD 0 ;:USER DEFINED
2206 001170 000000 $TMP3: .WORD 0 ;:USER DEFINED
2207 001172 000000 $TIMES: 0 ;:MAX. NUMBER OF ITERATIONS
2208 001174 000000 $ESCAPE: 0 ;:ESCAPE ON ERROR
2209 001176 177607 000377 $BELL: .ASCIZ <207><377><377> ;:CODE FOR BELL
2210 001202 077 $QUES: .ASCII /?/ ;:QUESTION MARK
2211 001203 015 $CRLF: .ASCII <15> ;:CARRIAGE RETURN
2212 001204 000012 $LF: .ASCIZ <12> ;:LINE FEED
2213 001206 000000 $ERPSW: .WORD 0 ;:ERROR PSW
2214 001210 000000 $STSTM: .WORD 0 ;:TEST NUMBER STORAGE
2215 001212 000100 $PR2: .WORD PR2 ;:PRIORITY LEVEL 2
2216 001214 000240 $PR5: .WORD PR5 ;:PRIORITY LEVEL 5
2217 001216 000000 $EPIRQ: .WORD 0 ;:ERROR PIRQ
2218 001220 000000 E1STKLM: .WORD 0 ;:STACK LIMIT REGISTER ERROR 1
2219 001222 000000 E2STKLM: .WORD 0 ;:STACK LIMIT REGISTER ERROR 2
2220 001224 000000 INTER5: .WORD 0 ;:ADDRESS OF BR5 INTER SUBROUTINE
```

2221	001226	000000	INT5VEC: .WORD	0	:BR 5 INTERRUPT VECTOR
2222	001230	000000	INT5ST: .WORD	0	:BR 5 STATUS REG
2223	001232	000000	INTER6: .WORD	0	:ADDRESS OF BR6 INTERRUPT SUBROUTINE
2224	001234	000000	INT6VEC: .WORD	0	:BR 6 INTERRUPT VECTOR
2225	001236	000000	INT6ST: .WORD	0	:BR 6 STATUS REG
2226	001240	172040	RSCS1: .WORD	172040	:ADDRESS OF RS STATUS REGISTER
2227	001242	000204	RSVEC: .WORD	204	:ADDRESS OF RS VECTOR
2228	001244	176700	RPCS1: .WORD	176700	:ADDRESS OF RP STATUS REGISTER
2229	001246	000254	RPVEC: .WORD	254	:ADDRESS OF RP VECTOR
2230	001250	177404	RKCS1: .WORD	177404	:ADDRESS OF RK STATUS REGISTER
2231	001252	000220	RKVEC: .WORD	220	:ADDRESS OF RK VECTOR
2232	001254	172440	TMCS1: .WORD	172440	:ADDRESS OF TM STATUS REG
2233	001256	000224	TMVEC: .WORD	224	:ADDRESS OF TM VECTOR
2234	001260	177546	LKSTAT: .WORD	177546	:ADDRESS OF LINE CLOCK STATUS REGISTER
2235	001262	000100	LKVEC: .WORD	100	:ADDRESS OF LINE CLOCK VECTOR
2236	001264	172540	PLKSTAT: .WORD	172540	:ADDRESS OF PROG LINE CLOCK STATUS REG
2237	001266	000104	PLKVEC: .WORD	104	:ADDRESS OF PROG LINE CLOCK VECTOR
2238	001270	000000	NEXTTST: .WORD	0	:ADDRESS OF NEXT TEST
2239	001272	000	KB11E: .BYTE	0	:KB-11E/EM WITHOUT MP CACHE
2240	001273	000	KB11EM: .BYTE	0	:KB-11E/EM WITH MP MODS
2241					
2242					
2243	000007		:OPCODE FOR MFPT INSTRUCTION (AVAILABLE ON KB11-E AND KB11-EM ONLY)		
			MFPT 7		

```
2244 ;:*****
2245
2246 .SBTTL ERROR POINTER TABLE
2247
2248 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
2249 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
2250 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
2251 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
2252 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
2253
2254 ;* EM ;:POINTS TO THE ERROR MESSAGE
2255 ;* DH ;:POINTS TO THE DATA HEADER
2256 ;* DT ;:POINTS TO THE DATA
2257 ;* DF ;:POINTS TO THE DATA FORMAT
2258
2259
2260 001274 $ERRTB:
2261
2262 001274 036560 ITEM1: EM1 ;:EITHER SPL FAILED OR BITS STUCK
2263 001276 036626 DH1 ;:ERRORPC SPL 5 PSW SPL 2 PSW TEST NUMBER
2264 ;: EXPECT ACTUAL EXPECT ACTUAL
2265 001300 036746 DT1 ;:$ERRPC,$PR5,$ERPSW,$PR2,$TMPO
2266 001302 036764 ITEM2: EM2 ;:PR5 LOADS OK BUT PR2 DOESN'T
2267 001304 036626 DH1
2268 001306 036746 DT1
2269 001310 037005 ITEM3: EM3 ;:PR2 LOADS OK BUT PR5 DOESN'T
2270 001312 036626 DH1
2271 001314 036746 DT1
2272 001316 037032 ITEM4: EM4 ;:FORK A FAILED TO D30.00
2273 001320 037075 DH4 ;:ERRORPC TEST NUMBER
2274 001322 037122 DT4 ;:$ERRPC,$$STSTM
2275 001324 037130 ITEM5: EM5 ;:FORK A FAILED TO RSD.02
2276 001326 037075 DH4
2277 001330 037122 DT4
2278 001332 037216 ITEM6: EM6 ;:RESET DID NOT WORK
2279 001334 037075 DH4
2280 001336 037122 DT4
2281 001340 037251 ITEM7: EM7 ;:FORK A FAILED INTO TRP.02
2282 001342 037075 DH4
2283 001344 037122 DT4
2284 001346 037265 ITEM10: EM10 ;:FORK A FAILED INTO WAT.00
2285 001350 037075 DH4
2286 001352 037122 DT4
2287 001354 037301 ITEM11: EM11 ;:FORK A FAILED TO MTP.00
2288 001356 037075 DH4
2289 001360 037122 DT4
2290 001362 037340 ITEM12: EM12 ;:FORK A FAILED TO MFP.80
2291 001364 037075 DH4
2292 001366 037122 DT4
2293 001370 037377 ITEM13: EM13 ;:PCB DID NOT LOAD FROM RS
2294 001372 037075 DH4
2295 001374 037122 DT4
2296 001376 037427 ITEM14: EM14 ;:MARK DID NOT LOAD SP PROPERLY
2297 001400 037457 DH14 ;:ERRORPC SP TEST NUMBER
2298 ;: EXPECT ACTUAL
2299 001402 037540 DT14 ;:$ERRPC,$REG1,$REG0,$$STSTM
```

2300	001404	037552	ITEM15: EM15	:R5 DID NOT LOAD PROPERLY
2301	001406	037602	DH15	:ERRORPC R5 TEST NUMBER
2302	001410	037540	DT14	
2303	001412	037663	ITEM16: EM16	:FORK A FAILED TO RSD.00
2304	001414	037075	DH4	
2305	001416	037122	DT4	
2306	001420	037717	ITEM17: EM17	:FORK A FAILED TO D67.01
2307	001422	037075	DH4	
2308	001424	037122	DT4	
2309	001426	037774	ITEM20: EM20	:R1 SHIFTED RIGHT INSTEAD OF LEFT
2310	001430	037075	DH4	
2311	001432	037122	DT4	
2312	001434	040065	ITEM21: EM21	:R1 DID NOT SHIFT
2313	001436	037075	DH4	
2314	001440	037122	DT4	
2315	001442	040105	ITEM22: EM22	:R1 SHIFTED BUT CARRY DID NOT SET
2316	001444	037075	DH4	
2317	001446	037122	DT4	
2318	001450	040202	ITEM23: EM23	:R1 SHIFTED LEFT INSTEAD OF RIGHT
2319	001452	037075	DH4	
2320	001454	037122	DT4	
2321	001456	040272	ITEM24: EM24	:SHIFT RIGHT DID NOT SIGN FILL
2322	001460	037075	DH4	
2323	001462	037122	DT4	
2324	001464	040325	ITEM25: EM25	:R1 SHIFTED BUT DON'T KNOW WHERE
2325	001466	040357	DH25	:ERRORPC R5 TEST NUMBER
2326				: EXPECT ACTUAL
2327	001470	040432	DT25	:SERRPC,\$TMP1,\$REG1,\$STSTNM
2328	001472	040444	ITEM26: EM26	:SHIFT OK BUT CARRY DID NOT LOAD
2329	001474	037075	DH4	
2330	001476	037122	DT4	
2331	001500	040504	ITEM27: EM27	:ASH.20 DID NOT LOAD CC'S CORRECTLY
2332	001502	041470	DH42	:ERRORPC PSW TEST NUMBER
2333				: EXPECT ACTUAL
2334	001504	041546	DT42	:SERRPC,\$TMP0,\$ERPSW,\$STSTNM
2335	001506	040546	ITEM30: EM30	:R1 SHIFTED WITH A SHIFT COUNT OF 0
2336	001510	037075	DH4	
2337	001512	037122	DT4	
2338	001514	040604	ITEM31: EM31	:ASH.40 DID NOT LOAD CC'S CORRECTLY
2339	001516	041470	DH42	
2340	001520	041546	DT42	
2341	001522	040646	ITEM32: EM32	:FORK A FAILED TO RSD.00
2342	001524	037075	DH4	
2343	001526	037122	DT4	
2344	001530	040712	ITEM33: EM33	:STATE ASH.00 FAILED
2345	001532	037075	DH4	
2346	001534	037122	DT4	
2347	001536	040730	ITEM34: EM34	:FORK B FAILED INTO MUL.00
2348	001540	037075	DH4	
2349	001542	037122	DT4	
2350	001544	040763	ITEM35: EM35	:FORK B FAILED TO RSD.00
2351	001546	037075	DH4	
2352	001550	037122	DT4	
2353	001552	041101	ITEM36: EM36	:FORK A FAILED TO RSD.00
2354	001554	037075	DH4	
2355	001556	037122	DT4	



2356	001560	041164	ITEM37:	EM37	:RACE E45 BAD (AFIRO4(*)*MUL:ASH(+MFP))
2357	001562	037075		DH4	
2358	001564	037122		DT4	
2359	001566	041234	ITEM40:	EM40	:RACE E33 BAD (AFIRO5(1)*(MUL:ASH(+MFP))
2360	001570	037075		DH4	
2361	001572	037122		DT4	
2362	001574	041304	ITEM41:	EM41	:RO DID NOT SIGN FILL ON RIGHT SHIFT
2363	001576	041347		DH41	:ERROR PC RO TEST NUMBER
2364					: EXPECT ACTUAL
2365	001600	041426		DT41	:SERRPC,\$TMP0,\$REG0,\$STSTNM
2366	001602	041440	ITEM42:	EM42	:BAD CC'S ON RIGHT SHIFT
2367	001604	041470		DH42	:ERRORPC PSW TEST NUMBER
2368					: EXPECT ACTUAL
2369	001606	041546		DT42	:SERRPC,\$TMP0,\$ERPSW,\$STSTNM
2370	001610	041560	ITEM43:	EM43	:RO<0> DID NOT GO TO R1<15>
2371	001612	041612		DH43	:ERRORPC RO R1 TEST NUMBER
2372					: EXPECT ACTUAL EXPECT ACTUAL
2373	001614	041722		DT43	:SERRPC,\$TMP0,\$REG0,\$TMP1,\$REG1,\$STSTNM
2374	001616	041740	ITEM44:	EM44	:RO DID NOT SHIFT LEFT PROPERLY
2375	001620	041612		DH43	
2376	001622	041722		DT43	
2377	001624	042004	ITEM45:	EM45	:BAD CC'S ON LEFT SHIFT
2378	001626	041470		DH42	
2379	001630	041546		DT42	
2380	001632	040325	ITEM46:	EM25	:R1 DID NOT SHIFT LEFT PROPERLY
2381	001634	041612		DH43	
2382	001636	041722		DT43	
2383	001640	042162	ITEM47:	EM47	:BAD CC'S ON NO SHIFT
2384	001642	041470		DH42	
2385	001644	041546		DT42	
2386	001646	042207	ITEM50:	EM50	:R1 DID NOT ROTATE PROPERLY
2387	001650	041612		DH43	
2388	001652	041722		DT43	
2389	001654	042241	ITEM51:	EM51	:BITS STUCK IN SC (52 PATTERN)
2390	001656	042301		DH51	:ERRORPC RO R1 C BIT
2391					: EXPECT ACTUAL EXPECT ACTUAL EXPECT ACTUAL
2392	001660	042440		DT51	:SERRPC,\$TMP0,\$REG0,\$TMP1,\$REG1,\$TMP2,\$ERPSW,\$STSTNM
2393	001662	042462	ITEM52:	EM52	:BITS STUCK IN SC (25 PATTERN)
2394	001664	042301		DH51	
2395	001666	042440		DT51	
2396	001670	042535	ITEM53:	EM53	:IRCB B FORK MUX INPUT B3 NOT GOING LOW
2397	001672	042301		DH51	
2398	001674	042440		DT51	
2399	001676	042570	ITEM54:	EM54	:STATE ASC.00 FAILED
2400	001700	041612		DH43	
2401	001702	041722		DT43	
2402	001704	042606	ITEM55:	EM55	:FORK A FAILED TO RSD.00
2403	001706	037075		DH4	
2404	001710	037122		DT4	
2405	001712	042664	ITEM56:	EM56	:EITHER GRAD DROO STUCK H OR RACK E64 BAD
2406	001714	041612		DH43	

2407	001716	041722		
2408	001720	042735		
2409	001722	041612		
2410	001724	041722		
2411	001726	043005	ITEM57:	DT43 EM57 ;EITHER GRAD DROO STUCK LOW OR RACK E64 BAD DH43
2412	001730	041612		
2413	001732	041722		
2414	001734	043106	ITEM60:	DT43 EM60 ;EITHER GRAJ SC=0 NOT GETTING TO RACK E50 DH43 ;OR RACK E50 BAD DT43
2415	001736	041612		
2416	001740	041722		
2417	001742	043164	ITEM61:	EM61 ;INSTRUCTION FAILED TO LOAD R0 & R1 CORRECTLY DH43 ;ON POSITIVE DT43
2418	001744	041470		
2419	001746	041546	ITEM62:	EM62 ;BAD CC'S DH42
2420	001750	043226		
2421	001752	041612		
2422	001754	041722	ITEM63:	DT42 EM63 ;R0 DID NOT LOAD ON NEG. DH43
2423	001756	043266		
2424	001760	041612	ITEM64:	DT43 EM64 ;R1 DID NOT LOAD ON NEG DH43
2425	001762	041722		
2426	001764	043326	ITEM65:	DT43 EM65 ;BAD CC'S DUE TO STATE MUL.50 DH42
2427	001766	041470		
2428	001770	041546		
2429	001772	043363	ITEM66:	DT42 EM66 ;C DID NOT SET ON OVERFLOW DH4
2430	001774	037075		
2431	001776	037122		
2432	002000	043414	ITEM67:	DT4 EM67 ;C DID NOT SET ON UNDERFLOW DH4
2433	002002	037075		
2434	002004	037122		
2435	002006	043446	ITEM70:	DT4 EM70 ;STATE MUL.00 FAILED DH43
2436	002010	041612		
2437	002012	041722		
2438	002014	042606	ITEM71:	DT43 EM55 ;BAD FIELD IN IR DECODE ROM DH4
2439	002016	037075		
2440	002020	037122		
2441	002022	043464	ITEM72:	DT4 EM72 ;STATE ASH.30 DID NOT LOAD CC'S CORRECTLY DH42
2442	002024	041470		
2443	002026	041546		
2444	002030	043526	ITEM73:	DT42 EM73 ;STATE ASH.41 FAILED DH25
2445	002032	040357		
2446	002034	040432		
2447	002036	043544	ITEM74:	DT25 EM74 ;BAD CC'S DUE TO STATE ASH.41 DH42
2448	002040	041470		
2449	002042	041546		
2450	002044	043606	ITEM75:	DT42 EM75 ;BEN2 FAILED DH4
2451	002046	037075		
2452	002050	037122		
2453	002052	043673	ITEM76:	DT4 EM76 ;BAD CC'S DUE TO DVE.00 DH42
2454	002054	041470		
2455	002056	041546		
2456	002060	043716	ITEM77:	DT42 EM77 ;BAD CC'S DUE TO DVC.70 DH42
2457	002062	041470		
2458	002064	041546		
2459	002066	043741	ITE100:	DT42 EM100 ;BEN16 FAILED DH4
2460	002070	037075		
2461	002072	037122		
2462	002074	043771	ITE101:	DT4 EM101 ;BEN4 FAILED

2463	002076	037075	DH4		
2464	002100	037122	DT4		
2465	002102	044115	ITE102:	EM102	:QUOTIENT OK REMAINDER BAD
2466	002104	041612		DH43	
2467	002106	041722		DT43	
2468	002110	044167	ITE103:	EM103	:BEN3 FAILED
2469	002112	037075		DH4	
2470	002114	037122		DT4	
2471	002116	044216	ITE104:	EM104	:BEN04 STUCK TO DIV SUB
2472	002120	037075		DH4	
2473	002122	037122		DT4	
2474	002124	044306	ITE105:	EM105	:CAN'T DETERMINE CAUSE OF FAILURE
2475	002126	041612		DH43	
2476	002130	041722		DT43	
2477	002132	044337	ITE106:	EM106	:BEN16 FAILED TO DIV.50
2478	002134	037075		DH4	
2479	002136	037122		DT4	
2480	002140	044306	ITE107:	EM105	:EM107 SAME AS EM105
2481	002142	041612		DH43	
2482	002144	041722		DT43	
2483	002146	044427	ITE110:	EM110	:BEN04*-N FAILED
2484	002150	037075		DH4	
2485	002152	037122		DT4	
2486	002154	044517	ITE111:	EM111	:BEN02 FAIL:J
2487	002156	037075		DH4	
2488	002160	037122		DT4	
2489	002162	044547	ITE112:	EM112	:BEN05 FAILED
2490	002164	037075		DH4	
2491	002166	037122		DT4	
2492	002170	044667	ITE113:	EM113	:BEN16 FAILED (RACK E50(B0))
2493	002172	037075		DH4	
2494	002174	037122		DT4	
2495	002176	044716	ITE114:	EM114	:BEN16 FAILED (RACK E64(B0))
2496	002200	037075		DH4	
2497	002202	037122		DT4	
2498	002204	044745	ITE115:	EM115	:ROM STATE FAILED
2499	002206	041612		DH43	
2500	002210	041722		DT43	
2501	002212	044115	ITE116:	EM102	:QUOTIENT OK, REMAINDER BAD
2502	002214	041612		DH43	
2503	002216	041722		DT43	
2504	002220	045013	ITE117:	EM117	:BAD CC'S IN DVC.90 OR RACK E63 BAD
2505	002222	041470		DH42	
2506	002224	041546		DT42	
2507	002226	045070	ITE120:	EM120	:BEN05 DIV QUIT (N(0)*SR15(0)) DID NOT
2508	002230	037075		DH4	:GO LOW OR RACK E63(C0) STUCK HIGH
2509	002232	037122		DT4	
2510	002234	045157	ITE121:	EM121	:CC'S BAD DUE TO DIV.30 OR DVE.20
2511	002236	041470		DH42	
2512	002240	041546		DT42	
2513	002242	045227	ITE122:	EM122	:GRAJ E5 BAD
2514	002244	037075		DH4	
2515	002246	037122		DT4	
2516	002250	045267	ITE123:	EM123	:RACK E63(D0) STUCK LOW
2517	002252	037075		DH4	
2518	002254	037122		DT4	

2519	002256	045316	ITE124:	EM124	;CC'S DID NOT LOAD PROPERLY
2520	002260	041470		DH42	
2521	002262	041546		DT42	
2522	002264	045350	ITE125:	EM125	;EITHER GRAJ E5(N(1)*SR15(1)) BAD
2523	002266	041612		DH43	;OR ROM STATE BAD
2524	002270	041722		DT43	
2525	002272	044115	ITE126:	EM102	;QUOT. OK REMAINDER BAD
2526	002274	041612		DH43	
2527	002276	041722		DT43	
2528	002300	045423	ITE127:	EM127	;QUOT. BAD, REMAINDER OK
2529	002302	041612		DH43	
2530	002304	041722		DT43	
2531	002306	045475	ITE130:	EM130	;QUOTIENT BAD
2532	002310	041612		DH43	
2533	002312	041722		DT43	
2534	002314	044115	ITE131:	EM102	;QUOT. OK, REMAINDER BAD
2535	002316	041612		DH43	
2536	002320	041722		DT43	
2537	002322	045475	ITE132:	EM130	;QUOT. BAD
2538	002324	041612		DH43	
2539	002326	041722		DT43	
2540	002330	044115	ITE133:	EM102	;QUOT. OK, REMAIN. BAD
2541	002332	041612		DH43	
2542	002334	041722		DT43	
2543	002336	045536	ITE134:	EM134	;BAD CC'S ON DIVISION OVERFLOW
2544	002340	041470		DH42	
2545	002342	041546		DT42	
2546	002344	045605	ITE135:	EM135	;RO DID NOT LOAD CORRECTLY
2547	002346	041347		DH41	
2548	002350	041426		DT41	
2549	002352	045614	ITE136:	EM136	;THE SP DID NOT INCREMENT
2550	002354	037075		DH4	
2551	002356	037122		DT4	
2552	002360	045316	ITE137:	EM124	;CC'S DID NOT LOAD CORRECTLY
2553	002362	041470		DH42	
2554	002364	041546		DT42	
2555	002366	045647	ITE140:	EM140	;FORK A FAILED
2556	002370	037075		DH4	
2557	002372	037122		DT4	
2558	002374	045704	ITE141:	EM141	;STATE MTP.10 FAILED
2559	002376	037075		DH4	
2560	002400	037122		DT4	
2561	002402	045743	ITE142:	EM142	---;SP-LOADED-INCORRECTLY----
2562	002404	037457		DH14	
2563	002406	037540		DT14	
2564	002410	045771	ITE143:	EM143	;STATE MTP.00 DID NOT PUT PCB IN DR
2565	002412	037075		DH4	
2566	002414	037122		DT4	
2567	002416	046025	ITE144:	EM144	;FORK A FAILED
2568	002420	037075		DH4	
2569	002422	037122		DT4	
2570	002424	046054	ITE145:	EM145	;STATE MFP.10 DID NOT DEC. THE SP
2571	002426	037075		DH4	
2572	002430	037122		DT4	
2573	002432	046113	ITE146:	EM146	;RO DID NOT GET PUT ON THE STACK
2574	002434	037075		DH4	

2575	002436	037122		DT4	
2576	002440	046152	ITE147:	EM147	:BAD CC'S
2577	002442	041470		DH42	
2578	002444	041546		DT42	
2579	002446	045647	ITE150:	EM140	:RACF X/CLASS DOES NOT GO HIGH
2580	002450	037075		DH4	
2581	002452	037122		DT4	
2582	002454	046203	ITE151:	EM151	:STATE MFP.00 IS BAD
2583	002456	037075		DH4	
2584	002460	037122		DT4	
2585	002462	046216	ITE152:	EM152	:BAD CC'S
2586	002464	041470		DH42	
2587	002466	041546		DT42	
2588	002470	037663	ITE153:	EM16	:RACE X/CLASS DOES NOT GO HIGH
2589	002472	037075		DH4	
2590	002474	037122		DT4	
2591	002476	042606	ITE154:	EM55	:R(NUL:ASHC+MFP) FIELD OF IR ROM BAD
2592	002500	037075		DH4	
2593	002502	037122		DT4	
2594	002504	046245	ITE155:	EM155	:STATE TRP.01 FAILED
2595	002506	037075		DH4	
2596	002510	037122		DT4	
2597	002512	046276	ITE156:	EM156	:TRAP VECTOR DECODE FAILED
2598	002514	037075		DH4	
2599	002516	037122		DT4	
2600	002520	046416	ITE157:	EM157	:STATE TRP.01 FAILED
2601	002522	037075		DH4	
2602	002524	037122		DT4	
2603	002526	046447	ITE160:	EM160	:STATE TRP.01 FAILED
2604	002530	037075		DH4	
2605	002532	037122		DT4	
2606	002534	046500	ITE161:	EM161	:TRAP VECTOR DECODE FAILED
2607	002536	037075		DH4	
2608	002540	037122		DT4	
2609	002542	046575	ITE162:	EM162	:STATE TRP.01 FAILED
2610	002544	037075		DH4	
2611	002546	037122		DT4	
2612	002550	046626	ITE163:	EM163	:BIT FAILED IN PIRQ REG
2613	002552	046655		DH163	:ERRORPC PIRQ TEST NUMBER
2614					: EXPECT ACTUAL
2615	002554	046734		DT163	:SERRPC,\$TMPO,\$EPIRQ,\$STSTNM
2616	002556	046746	ITE164:	EM164	:STATE TST.10 HAS BAD BEN FIELD
2617	002560	037075		DH4	
2618	002562	037122		DT4	
2619	002564	046777	ITE165:	EM165	:HONOR PIR 1 DOES NOT GO LOW
2620	002566	037075		DH4	
2621	002570	037122		DT4	
2622	002572	047061	ITE166:	EM166	:PIR 6 WORKS BUT 4 & 1 DON'T
2623	002574	037075		DH4	
2624	002576	037122		DT4	
2625	002600	047166	ITE167:	EM167	:TMCA ABOVE BR7 MIGHT BE STUCK LOW
2626	002602	037075		DH4	
2627	002604	037122		DT4	
2628	002606	047230	ITE170:	EM170	:TMCE BRQ CLDCK MIGH BE STUCH LOW
2629	002610	037075		DH4	
2630	002612	037122		DT4	

2631	002614	047272	ITE171:	EM171	:BEN 13 FAILED TO PUP.00
2632	002616	037075		DH4	
2633	002620	037122		DT4	
2634	002622	047413	ITE172:	EM172	:BEN 13 FAILED TO SER.00
2635	002624	037075		DH4	
2636	002626	037122		DT4	
2637	002630	042033	ITE173:	EM46	:MFPT FAILED TO LOAD R0 CORRECTLY
2638	002632	042066		DH46	
2639	002634	042150		DT46	
2640	002636	047666	ITE174:	EM174	:PIR 2 DID NOT INTERRUPT
2641	002640	037075		DH4	
2642	002642	037122		DT4	
2643	002644	047750	ITE175:	EM175	:BEN 13 FAILED
2644	002646	037075		DH4	
2645	002650	037122		DT4	
2646	002652	047765	ITE176:	EM176	:LEVEL 1 INT. WHEN CPU LEVEL 1 ENABLED
2647	002654	050157		DH201	:ERRORPC PIRQ TEST NUMBER
2648	002656	050210		DT201	:\$ERRPC,\$EPIRQ,\$\$STSTNM
2649	002660	050027	ITE177:	EM177	:PIR 3 DID NOT INTERRUPT
2650	002662	037075		DH4	
2651	002664	037122		DT4	
2652	002666	047750	ITE200:	EM175	:BEN 13 FAILED
2653	002670	037075		DH4	
2654	002672	037122		DT4	
2655	002674	050111	ITE201:	EM201	:LEVEL 2 WHEN CPU LEVEL 2 ENABLED
2656	002676	050157		DH201	
2657	002700	050210		DT201	
2658	002702	050220	ITE202:	EM202	:PIR 4 DID NOT INTERRUPT
2659	002704	037075		DH4	
2660	002706	037122		DT4	
2661	002710	047750	ITE203:	EM175	:BEN 13 FAILED
2662	002712	037075		DH4	
2663	002714	037122		DT4	
2664	002716	050302	ITE204:	EM204	:LEVEL 3 WHEN CPU LEVEL 3 ENABLED
2665	002720	050157		DH201	
2666	002722	050210		DT201	
2667	002724	050350	ITE205:	EM205	:PIR 5 DID NOT INTERRUPT
2668	002726	037075		DH4	
2669	002730	037122		DT4	
2670	002732	047750	ITE206:	EM175	:BEN 13 FAILED
2671	002734	037075		DH4	
2672	002736	037122		DT4	
2673	002740	050433	ITE207:	EM207	:LEVEL 4 INTERRUPT WHEN NOT SUPPOSE TO
2674	002742	050157		DH201	
2675	002744	050210		DT201	
2676	002746	050501	ITE210:	EM210	:FAILURE AFTER TMCB E70
2677	002750	037075		DH4	
2678	002752	037122		DT4	
2679	002754	050547	ITE211:	EM211	:FAILURE IN TMCB E70 OR BEFORE
2680	002756	037075		DH4	
2681	002760	037122		DT4	
2682	002762	050630	ITE212:	EM212	:BEN 13 FAILED
2683	002764	037075		DH4	
2684	002766	037122		DT4	
2685	002770	050674	ITE213:	EM213	:LEVEL 5 INTERRUPT WHEN NOT SUPPOSE TO
2686	002772	050157		DH201	

2687	002774	050210		DT201	
2688	002776	050742	ITE214:	EM214	;LEVEL 7 DID NOT INTERRUPT
2689	003000	037075		DH4	
2690	003002	037122		DT4	
2691	003004	047750	ITE215:	EM175	;BEN 13 FAILED
2692	003006	037075		DH4	
2693	003010	037122		DT4	
2694	003012	051024	ITE216:	EM216	;LEVEL 6 INTERRUPT WHEN NOT SUPPOSE TO
2695	003014	050157		DH201	
2696	003016	050210		DT201	
2697	003020	051072	ITE217:	EM217	;NO TIMEOUT ON DATI
2698	003022	037075		DH4	
2699	003024	037122		DT4	
2700	003026	051126	ITE220:	EM220	;BEN 13 FAILED
2701	003030	037075		DH4	
2702	003032	037122		DT4	
2703	003034	051207	ITE221:	EM221	;NO TIMEOUT ON DATO
2704	003036	037075		DH4	
2705	003040	037122		DT4	
2706	003042	051243	ITE222:	EM222	;BR 4 INTERRUPTS WHEN CPU AT LEVEL 7
2707	003044	037075		DH4	
2708	003046	037122		DT4	
2709	003050	051323	ITE223:	EM223	;BOTH BR 4 & BR 6 FAILED
2710	003052	037075		DH4	
2711	003054	037122		DT4	
2712	003056	051344	ITE224:	EM224	;BR 4 FAILED
2713	003060	037075		DH4	
2714	003062	037122		DT4	
2715	003064	051513	ITE225:	EM225	;BR 4 FAILED BUT BR 6 OK
2716	003066	037075		DH4	
2717	003070	037122		DT4	
2718	003072	051624	ITE226:	EM226	;BR 5 INTERRUPT FAILED
2719	003074	037075		DH4	
2720	003076	037122		DT4	
2721	003100	051704	ITE227:	EM227	;BR 6 INTERRUPT FAILED
2722	003102	037075		DH4	
2723	003104	037122		DT4	
2724	003106	051765	ITE230:	EM230	;YEL ZONE TRAP FAILED
2725	003110	037075		DH4	
2726	003112	037122		DT4	
2727	003114	000000	ITE231:	0	;DELETED
2728	003116	037075		DH4	
2729	003120	037122		DT4	
2730	003122	052252	ITE232:	EM232	;JSR WITH BAD STACK FAILED
2731	003124	037075		DH4	
2732	003126	037122		DT4	
2733	003130	052324	ITE233:	EM233	;STACK LIMIT REG DID NOT DISABLE YEL ZONE
2734	003132	037075		DH4	
2735	003134	037122		DT4	
2736	003136	052421	ITE234:	EM234	;TMCC KERNAL R6 DID NOT DISABLE YEL ZONE
2737	003140	037075		DH4	
2738	003142	037122		DT4	
2739	003144	052526	ITE235:	EM235	;RED ZONE REFERENCE FAILED
2740	003146	037075		DH4	
2741	003150	037122		DT4	
2742	003152	000000	ITE236:	0	;DELETED

2743	003154	037075	DH4	
2744	003156	037122	DT4	
2745	003160	052673	ITE237:	EM237 ;BEN 13 FAILED TO PUP.00
2746	003162	037075	DH4	
2747	003164	037122	DT4	
2748	003166	052766	ITE240:	EM240 ;BEN 13 FAILED TO BRK.80
2749	003170	037075	DH4	
2750	003172	037122	DT4	
2751	003174	053070	ITE241:	EM241 ;NO RED ZONE ON STACK OVERFLOW
2752	003176	037075	DH4	
2753	003200	037122	DT4	
2754	003202	053201	ITE242:	EM242 ;NO RED ZONE WHEN SL REG>BADDR
2755	003204	037075	DH4	
2756	003206	037122	DT4	
2757	003210	053335	ITE243:	EM243 ;52400 PATTERN FAILED IN SL REG
2758	003212	053405	DH243	;
2759				ERRORPC SL REG TEST NUMBER
2760	003214	053470	DT243	;
2761	003216	053502	ITE244:	EM244 ;\$ERRPC,\$TMP0,E2STKLMT,\$\$STNM
2762	003220	053405	DH243	;
2763	003222	053552	DT244	;
2764	003224	053564	ITE245:	EM245 ;\$ERRPC,\$TMP0,E1STKLMT,\$\$STNM
2765	003226	037075	DH4	;
2766	003230	037122	DT4	;
2767	003232	053657	ITE246:	EM246 ;SL AND PB BOTH SELECTED
2768	003234	037075	DH4	
2769	003236	037122	DT4	
2770	003240	054000	ITE247:	EM247 ;PSW SELECTED INSTEAD OF SL
2771	003242	037075	DH4	
2772	003244	037122	DT4	
2773	003246	054111	ITE250:	EM250 ;WHAT HAPPENED?
2774	003250	054136	DH250	;
2775	003252	054254	DT250	;
2776	003254	054272	ITE251:	EM251 ;YEL ZONE IN RED REGION (SP=330)
2777	003256	037075	DH4	
2778	003260	037122	DT4	
2779	003262	054437	ITE252:	EM252 ;YEL ZONE IN RED REGION (SP=240)
2780	003264	037075	DH4	
2781	003266	037122	DT4	
2782	003270	054507	ITE253:	EM253 ;YEL ZONE IN RED REGION (SP=140)
2783	003272	037075	DH4	
2784	003274	037122	DT4	
2785	003276	054557	ITE254:	EM254 ;RED ZONE IN YELLOW REGION (SP=376)
2786	003300	037075	DH4	
2787	003302	037122	DT4	
2788	003304	054647	ITE255:	EM255 ;CPU ERR REG BIT 4 DOES NOT SET
2789	003306	054726	DH255	
2790	003310	041426	DT41	
2791	003312	055007	ITE256:	EM256 ;CPU ERR REG DOES NOT CLEAR
2792	003314	037075	DH4	
2793	003316	037122	DT4	
2794	003320	055044	ITE257:	EM257 ;CPU ERROR BIT 3 DOES NOT SET
2795	003322	054726	DH255	
2796	003324	041426	DT41	
2797	003326	055115	ITE260:	EM260 ;CPU ERROR BIT 3 DOES NOT CLEAR
2798	003330	037075	DH4	



2799	003332	037122		DT4	
2800	003334	055160	ITE261:	EM261	;CPU ERROR BIT 2 DOES NOT SET
2801	003336	054726		DH255	
2802	003340	041426		DT41	
2803	003342	055232	ITE262:	EM262	;CPU ERROR BIT 2 DOES NOT CLEAR
2804	003344	037075		DH4	
2805	003346	037122		DT4	
2806	003350	055274	ITE263:	EM263	
2807	003352	055463		DH263	
2808	003354	000000		0	
2809	003356	056106	ITE264:	EM264	;GOING TO NEXT TEST
2810	003360	000000		0	
2811	003362	000000		0	
2812	003364	056131	ITE265:	EM265	;NEITHER - BYIN NOR DATI CAUSED ODD ADDR.
2813	003366	037075		DH4	
2814	003370	037122		DT4	
2815	003372	056447	ITE266:	EM266	;DATI TRAPPED BUT - BYIN DIDN'T
2816	003374	037075		DH4	
2817	003376	037122		DT4	
2818	003400	056565	ITE267:	EM267	;BOTH DATI & DATO FAILED
2819	003402	037075		DH4	
2820	003404	037122		DT4	
2821	003406	056645	ITE270:	EM270	;DATO WORKS BUT DATI FAILED
2822	003410	037075		DH4	
2823	003412	037122		DT4	
2824	003414	056743	ITE271:	EM271	;NO TRAP ON DATO
2825	003416	037075		DH4	
2826	003420	037122		DT4	
2827	003422	057073	ITE272:	EM272	;NO TRAP ON SM357*SRC1 DATI
2828	003424	037075		DH4	
2829	003426	037122		DT4	
2830	003430	057132	ITE273:	EM273	;ODD ADDR BIT IN CPU ERROR FAILED
2831	003432	037075		DH4	
2832	003434	037122		DT4	
2833	003436	057200	ITE274:	EM274	;NO TRAP ON DATO
2834	003440	037075		DH4	
2835	003442	037122		DT4	
2836	003444	057220	ITE275:	EM275	;T BIT TRAP FAILED
2837	003446	037075		DH4	
2838	003450	037122		DT4	
2839	003452	057570	ITE276:	EM276	;T BIT NEVER SET
2840	003454	037075		DH4	
2841	003456	037122		DT4	
2842	003460	057621	ITE277:	EM277	;PS<08> DID NOT DISABLE T TRAP (KB-11E/EM ONLY)
2843	003462	037075		DH4	
2844	003464	037122		DT4	
2845	003466	057675	ITE300:	EM300	;TRAP VECTOR DECODE FAILED
2846	003470	037075		DH4	
2847	003472	037122		DT4	
2848	003474	057764	ITE301:	EM301	;RTT DID NOT DISABLE T BIT
2849	003476	037075		DH4	
2850	003500	037122		DT4	
2851	003502	060051	ITE302:	EM302	;PIR 1 DID NOT DISABLE ON BR4
2852	003504	037075		DH4	
2853	003506	037122		DT4	
2854	003510	060116	ITE303:	EM303	;PIR 1 CAME THRU ON YELLOW ZONE

2855	003512	037075	DH4	
2856	003514	037122	DT4	
2857	003516	060247	ITE304:	EM304 ;FIR 2 CAME THRU ON YELLOW ZONE
2858	003520	037075	DH4	
2859	003522	037122	DT4	
2860	003524	060334	ITE305:	EM305 ;PIR 3 CAME THRU ON YELLOW ZONE
2861	003526	037075	DH4	
2862	003530	037122	DT4	
2863	003532	060422	ITE306:	EM306 ;BR4 CAME THRU ON PIR 5
2864	003534	037075	DH4	
2865	003536	037122	DT4	
2866	003540	060544	ITE307:	EM307 ;BR4 CAME THRU ON PIR 5
2867	003542	037075	DH4	
2868	003544	037122	DT4	
2869	003546	060617	ITE310:	EM310 ;BR4 CAME THRU ON BR 5
2870	003550	037075	DH4	
2871	003552	037122	DT4	
2872	003554	060671	ITE311:	EM311 ;BR4 CAME IN ON PIR 6
2873	003556	037075	DH4	
2874	003560	037122	DT4	
2875	003562	060742	ITE312:	EM312 ;BR4 CAME IN ON PIR 7
2876	003564	037075	DH4	
2877	003566	037122	DT4	
2878	003570	061066	ITE313:	EM313 ;PIR 4 CAME IN ON BR5
2879	003572	037075	DH4	
2880	003574	037122	DT4	
2881	003576	061136	ITE314:	EM314 ;PIR 4 CAME IN ON BR6
2882	003600	037075	DH4	
2883	003602	037122	DT4	
2884	003604	061247	ITE315:	EM315 ;PIR 4 CAME THRU ON SL YELLOW
2885	003606	037075	DH4	
2886	003610	037122	DT4	
2887	003612	061371	ITE316:	EM316 ;BR5 CAME IN ON PIR 5
2888	003614	037075	DH4	
2889	003616	037122	DT4	
2890	003620	061504	ITE317:	EM317 ;BR5 CAME IN ON PIR 6
2891	003622	037075	DH4	
2892	003624	037122	DT4	
2893	003626	061524	ITE320:	EM320 ;BR5 CAME IN ON PIR 7
2894	003630	037075	DH4	
2895	003632	037122	DT4	
2896	003634	061642	ITE321:	EM321 ;PIR 5 CAME IN ON BR6
2897	003636	037075	DH4	
2898	003640	037122	DT4	
2899	003642	061712	ITE322:	EM322 ;PIR 5 CAME IN ON SL YELLOW
2900	003644	037075	DH4	
2901	003646	037122	DT4	
2902	003650	061767	ITE323:	EM323 ;BR6 CAME IN ON PIR 6
2903	003652	037075	DH4	
2904	003654	037122	DT4	
2905	003656	062077	ITE324:	EM324 ;BR6 CAME IN ON PIR 7
2906	003660	037075	DH4	
2907	003662	037122	DT4	
2908	003664	062160	ITE325:	EM325 ;PIR 6 CAME IN ON SL YELLOW
2909	003666	037075	DH4	
2910	003670	037122	DT4	

2911	003672	062235	ITE326:	EM326	;PIR 7 CAME IN ON SL YELLOW
2912	003674	037075		DH4	
2913	003676	037122		DT4	
2914	003700	062316	ITE327:	EM327	;PSW BIT 11 DID NOT SET
2915	003702	037075		DH4	
2916	003704	037122		DT4	
2917	003706	062461	ITE330:	EM330	;R12 CLEARED R2
2918	003710	037075		DH4	
2919	003712	037122		DT4	
2920	003714	062572	ITE331:	EM331	;R2 SRC WAS AFFECTED BY CLR R12
2921	003716	037075		DH4	
2922	003720	037122		DT4	
2923	003722	062705	ITE332:	EM332	;R2 DST WAS AFFECTED BY (R12)+
2924	003724	037075		DH4	
2925	003726	037122		DT4	
2926	003730	062766	ITE333:	EM333	;R2 SRC WAS AFFECTED BY (R12)+
2927	003732	037075		DH4	
2928	003734	037122		DT4	
2929	003736	063021	ITE334:	EM334	;R15 DST AFTER UPAD 2
2930	003740	037075		DH4	
2931	003742	037122		DT4	
2932	003744	063054	ITE335:	EM335	;R15 SRC DOES NOT SELECT ON UPAD 2
2933	003746	037075		DH4	
2934	003750	037122		DT4	
2935	003752	063111	ITE336:	EM336	;PDRD PS11 DOES NOT GET TO GRAC
2936	003754	037075		DH4	
2937	003756	037122		DT4	
2938	003760	063160	ITE337:	EM337	;BAD BITS IN GPR SET 1 SRC
2939	003762	063225		DH337	;ERRORPC PATTERN TESTNUMBER
2940	003764	063262		DT337	;SERRPC,\$TMP1,\$STSTNM
2941	003766	063272	ITE340:	EM340	;BAD BITS IN GPR SET 1 DST
2942	003770	063225		DH337	
2943	003772	063262		DT337	
2944	003774	063344	ITE341:	EM341	;GRAB DST SET 1 DOES NOT GO LOW ON R14
2945	003776	037075		DH4	
2946	004000	037122		DT4	
2947	004002	063411	ITE342:	EM342	;GRAB SRC SET 1 DOES NOT GO LOW ON R14
2948	004004	037075		DH4	
2949	004006	037122		DT4	
2950	004010	063456	ITE343:	EM343	;50000 PATTERN FAILED IN PSW
2951	004012	041470		DH42	
2952	004014	041546		DT42	
2953	004016	063512	ITE344:	EM344	;164000 PATTERN FAILED IN PSW
2954	004020	041470		DH42	
2955	004022	041546		DT42	
2956	004024	063547	IRE345:	EM345	;PSW HIGH BYTE DID NOT CLEAR
2957	004026	041470		DH42	
2958	004030	041546		DT42	
2959	004032	063602	ITE346:	EM346	;SUPER SP DOES NOT SELECT ON UPAD 5
2960	004034	037075		DH4	
2961	004036	037122		DT4	
2962	004040	063662	ITE347:	EM347	;SUPER SP DOES NOT SELECT ON UPAD 0
2963	004042	037075		DH4	
2964	004044	037122		DT4	
2965	004046	063742	ITE350:	EM350	;USER SP DOES NOT SELECT ON UPAD 5
2966	004050	037075		DH4	

2967	004052	037122		
2968	004054	064017	ITE351:	DT4
2969	004056	037075		
2970	004060	037122		
2971	004062	064074	ITE352:	EM352
2972	004064	063225		
2973	004066	063262		
2974	004070	064141	ITE353:	EM353
2975	004072	063225		
2976	004074	063262		
2977	004076	064206	ITE354:	EM354
2978	004100	037075		
2979	004102	037122		
2980	004104	064240	ITE355:	EM355
2981	004106	037075		
2982	004110	037122		
2983	004112	064317	ITE356:	EM356
2984	004114	037075		
2985	004116	037122		
2986	004120	064352	ITE357:	EM357
2987	004122	064443		
2988				
2989	004124	037540		
2990	004126	064515	ITE360:	EM360
2991	004130	037075		
2992	004132	037122		
2993	004134	064574	ITE361:	EM361
2994	004136	037075		
2995	004140	037122		
2996	004142	064644	ITE362:	EM362
2997	004144	037075		
2998	004146	037122		
2999	004150	064701	ITE363:	EM363
3000	004152	037075		
3001	004154	037122		
3002	004156	064751	ITE364:	EM364
3003	004160	037075		
3004	004162	037122		
3005	004164	065033	ITE365:	EM365
3006	004166	037075		
3007	004170	037122		
3008	004172	065070	ITE366:	EM366
3009	004174	037075		
3010	004176	037122		
3011	004200	065124	ITE367:	EM367
3012	004202	041470		
3013	004204	041546		
3014	004206	065240	ITE370:	EM370
3015	004210	041470		
3016	004212	041546		
3017	004214	065365	ITE371:	EM371
3018	004216	041470		
3019	004220	041546		
3020	004222	065472	ITE372:	EM372
3021	004224	041470		
3022	004226	041546		

;USER SP DOES NOT SELECT ON UPAD 0

;EITHER USER OR SUPER SP DST FAILED

;EITHER USER OR SUPER SP SRC FAILED

;KSP SRC CHANGED ON MTP

;KSP SRC & DST CHANGED ON MTP

;KSP DST CHANGED ON MTP

;SSP DID NOT LOAD PROPERLY ON MTPD  
 ;ERRORPC SSP TEST NUMBER  
 ; EXPECT ACTUAL

;USER SP DID NOT LOAD ON MTP

;BAD FIELD IN IR DECODE ROM

;SSP WAS READ AND USP WAS WRITTEN

;SSP WAS READ AND WRITTEN

;CAN'T DETERMINE WHAT HAPPENED  
 ;SSP WAS READ BUT THE WRITE FAILED

;USP WAS READ BUT SSP WAS WRITTEN

;USP WAS READ BUT REG 7 WAS WRITTEN

;SPL WORKED IN SUPERVISOR MODE

;BIT FAILED TO SET IN PSW

;BIT FAILED TO CLEAR IN PSW

;BITS <13:11> DID NOT PRESET

3023	004230	065622	ITE373: EM373	:IOT DID NOT CHANGE PSW CORRECTLY
3024	004232	041470	DH42	
3025	004234	041546	DT42	
3026	004236	065734	ITE374: EM374	:PREVIOUS MODE BITS DID NOT CLEAR
3027	004240	041470	DH42	
3028	004242	041546	DT42	
3029	004244	066024	ITE375: EM375	:NO KT ABORT
3030	004246	037075	DH4	
3031	004250	037122	DT4	
3032	004252	066177	ITE376: EM376	:PS<08> FAILED TO SET
3033	004254	037075	DH4	
3034	004256	037122	DT4	
3035	004260	000000	ITE377: 0	
3036	004262	000000	0	
3037	004264	000000	0	
3038	004266	066227	ITE400: EM400	:KT ABORT TRAPPED TO 4
3039	004270	037075	DH4	
3040	004272	037122	DT4	
3041	004274	066330	ITE401: EM401	:KT ABORT TRAPPED TO 10
3042	004276	037075	DH4	
3043	004300	037122	DT4	
3044	004302	066375	ITE402: EM402	:KT ABORT TRAPPED TO 240
3045	004304	037075	DH4	
3046	004306	037122	DT4	
3047	004310	066437	ITE403: EM403	:KT TRAP DID NOT WORK
3048	004312	037075	DH4	
3049	004314	037122	DT4	
3050	004316	000000	ITE404: 0	:DELETED
3051	004320	037075	DH4	
3052	004322	037122	DT4	
3053	004324	066543	ITE405: EM405	:NO KT TRAP ON SOURCE OPERAND
3054	004326	037075	DH4	
3055	004330	037122	DT4	
3056	004332	066661	ITE406: FM406	:NO ABORT ON NEXM
3057	004334	037075	DH4	
3058	004336	037122	DT4	
3059	004340	000000	ITE407: 0	:DELETED
3060	004342	037075	DH4	
3061	004344	037122	DT4	
3062	004346	067011	ITE410: EM410	:NEXM BIT DID NOT SET IN CPUERR REG
3063	004350	054726	DH255	
3064	004352	041426	DT41	
3065	004354	067114	ITE411: EM411	:NEXM BIT DID NOT CLEAR IN CPUERR REG
3066	004356	037075	DH4	
3067	004360	037122	DT4	
3068	004362	067163	ITE412: EM412	:KT ABORT ON NEXM (KB11-B/C)
3069	004364	037075	DH4	
3070	004366	037122	DT4	
3071	004370	067240	ITE413: EM413	:KT ABORT ON SL RED
3072	004372	037075	DH4	
3073	004374	037122	DT4	
3074	004376	067313	ITE414: EM414	:KT ABORT ON ODD ADDRESS
3075	004400	037075	DH4	
3076	004402	037122	DT4	
3077	004404	067374	ITE415: EM415	:KT ABORT FAILED TO OVER-RIDE NEXM (KB11-E/EM)
3078	004406	037075	DH4	

3079	004410	037122		DT4	
3080	004412	067457	ITE416:	EM416	:TMCE CACHE BEND DID NOT GO
3081	004414	037075		DH4	:HIGH ON KT ABORT
3082	004416	037122		DT4	
3083	004420	067532	ITE417:	EM417	:ILLEGAL HALT DID NOT TRAP
3084	004422	037075		DH4	
3085	004424	037122		DT4	
3086	004426	067623	ITE420:	EM420	:CPU ERROR REG BIT 5 DID NOT SET
3087	004430	054726		DH255	
3088	004432	041426		DT41	
3089	004434	067700	ITE421:	EM421	:BEN 6 FAILED ON PS RESTORE
3090	004436	037075		DH4	
3091	004440	037122		DT4	
3092	004442	067771	ITE422:	EM422	:NO PE ABORT
3093	004444	037075		DH4	
3094	004446	037122		DT4	
3095	004450	070201	ITE423:	EM423	:PE ABORTED TO 4
3096	004452	037075		DH4	
3097	004454	037122		DT4	
3098	004456	070273	ITE424:	EM424	:PE ABORTED TO 14
3099	004460	037075		DH4	
3100	004462	037122		DT4	
3101	004464	070352	ITE425:	EM425	:PE ABORTED TO 104
3102	004466	037075		DH4	
3103	004470	037122		DT4	
3104	004472	070371	ITE426:	EM426	:NO PE TRAP
3105	004474	037075		DH4	
3106	004476	037122		DT4	
3107	004500	070455	ITE427:	EM427	:PE TRAP, TRAPPED TO
3108	004502	037075		DH4	:WRONG VECTOR
3109	004504	037122		DT4	
3110	004506	070561	ITE430:	EM430	:PIR 6 CAME IN ON MEM MGT TRAP
3111	004510	037075		DH4	
3112	004512	037122		DT4	
3113	004514	070624	ITE431:	EM431	:PIR 3 CAME IN ON MEM MGT TRAP
3114	004516	037075		DH4	
3115	004520	037122		DT4	
3116	004522	070670	ITE432:	EM432	:YEL ZONE CAME IN ON PE TRAP
3117	004524	037075		DH4	
3118	004526	037122		DT4	
3119	004530	070754	ITE433:	EM433	:MEM MGT TRAP CAME IN ON PE
3120	004532	037075		DH4	
3121	004534	037122		DT4	
3122	004536	000000	ITE434:	0	:DELETED
3123	004540	037075		DH4	
3124	004542	037122		DT4	
3125	004544	071022	ITE435:	EM435	:TMCC PRIORITY CLEAR DID NOT WORK
3126	004546	037075		DH4	
3127	004550	037122		DT4	
3128	004552	071142	ITE436:	EM436	:UNIBUS PE ABORT DID NOT HAPPEN
3129	004554	037075		DH4	
3130					
3131	004556	037122		DT4	
3132	004560	071205	ITE437:	EM437	:UNIBUS PE TRAPPED TO 0
3133	004562	037075		DH4	
3134	004564	037122		DT4	

3135					
3136	004566	071255	ITE440:	EM440	;WAIT INSTRUCTION FAILED
3137	004570	037075		DH4	
3138	004572	037122		DT4	
3139	004574	071330	ITE441:	EM441	;T BIT INTERRUPTED WAIT
3140	004576	037075		DH4	
3141	004600	037122		DT4	
3142	004602	071401	ITE442:	EM442	;PIRQ DID NOT INTERRUPT WAIT
3143	004604	037075		DH4	
3144	004606	037122		DT4	
3145	004610	071456	ITE443:	EM443	;STATE S13.00 FAILED
3146	004612	037075		DH4	
3147	004614	037122		DT4	
3148	004616	071474	ITE444:	EM444	;STATE S45.00 FAILED
3149	004620	037075		DH4	
3150	004622	037122		DT4	
3151	004624	071512	ITE445:	EM445	;STATE MTP.10 FAILED
3152	004626	037075		DH4	
3153	004630	037122		DT4	
3154	004632	071530	ITE446:	EM446	;STATE NEG.00 FAILED
3155	004634	037075		DH4	
3156	004636	037122		DT4	
3157	004640	071546	ITE447:	EM447	;STATE D45.00 FAILED
3158	004642	037075		DH4	
3159	004644	037122		DT4	
3160	004646	071564	ITE450:	EM450	;STATE D45.90 FAILED
3161	004650	037075		DH4	
3162	004652	037122		DT4	
3163	004654	071602	ITE451:	EM451	;STATE D45.00 FAILED
3164	004656	037075		DH4	
3165	004660	037122		DT4	
3166	004662	071620	ITE452:	EM452	;STATE D45.01 FAILED
3167	004664	037075		DH4	
3168	004666	037122		DT4	
3169	004670	071636	ITE453:	EM453	;RESERVED INSTRUCTION TRAP FAILED
3170	004672	037075		DH4	
3171	004674	037122		DT4	
3172	004676	071704	ITE454:	EM454	;TMCB PIRQ DOES NOT GO LOW
3173	004700	037075		DH4	
3174	004702	037122		DT4	
3175	004704	071771	ITE455:	EM455	;BEN06 FAILED
3176	004706	037075		DH4	
3177	004710	037122		DT4	
3178	004712	072107	ITE456:	EM456	;ODD ADDRESS FAILED ON DATI & DATO
3179	004714	037075		DH4	
3180	004716	037122		DT4	
3181	004720	072163	ITE457:	EM457	;PSW BIT 11 DOES NOT CLEAR
3182	004722	037075		DH4	
3183	004724	037122		DT4	
3184	004726	072214	ITE460:	EM460	;PSW CHANGED ON RESET
3185	004730	041470		DH42	
3186	004732	041546		DT42	
3187	004734	072260	ITE461:	EM461	;PSW CHANGES ON RESET IN SUPER MODE
3188	004736	041470		DH42	
3189	004740	041546		DT42	
3190	004742	012737	000014 177746 START:	MOV #14,@#CONTRL	;FORCE MISSES IN CACHE

```
3191 004750 005037 170200 CLR @MAPLO ;SETUP MAP 0 AND 1 TO PHISICAL CORE
3192 004754 005037 170202 CLR @MAPHO
3193 004760 012737 020000 170204 MOV #20000,@MAPL1
3194 004766 005037 170206 CLR @MAPH1
3195 004772 012737 000340 177776 MOV #340,@#PS ;:LOCK OUT ALL INTERRUPTS
3196 005000 012706 001100 MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
3197 005004 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
3198 005006 022706 001136 CMP #STKS,R6 ;:DONE?
3199 005012 001374 BNE .-6 ;:LOOP BACK IF NO
3200 005014 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER
3201 005020 012737 033302 000020 MOV #SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
3202 005026 012737 000340 000022 MOV #340,@#IOTVEC+2 ;:LEVEL 7
3203 005034 012737 033550 000030 MOV #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
3204 005042 012737 000340 000032 MOV #340,@#EMTVEC+2 ;:LEVEL 7
3205 005050 012737 036246 000034 MOV #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
3206 005056 012737 000340 000036 MOV #340,@#TRAPVEC+2 ;:LEVEL 7
3207 005064 012737 036300 000024 MOV #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
3208 005072 012737 000340 000026 MOV #340,@#PWRVEC+2 ;:LEVEL 7
3209 005100 016767 025316 025306 MOV #SENDCT,#EOPCT ;:SETUP END-OF-PROGRAM COUNTER
3210 005106 005067 174060 CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
3211 005112 005067 174056 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
3212 005116 112767 000001 173771 MOV #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
3213 005124 012737 032652 000014 MOV #SRTN,@#TBITVEC ;:SET 'T' BIT VECTOR TO $RTRN
3214 005132 012737 000340 000016 MOV #340,@#TBITVEC+2 ;:LEVEL 7
3215 005140 012767 000002 025504 MOV #RTI,$RTRN ;:SET $RTRN TO A RTI
3216 005146 012737 005174 000010 MOV #65,$@#RESVEC ;:TRY TO DO A RTT
3217 005154 005046 CLR -(SP) ;:DUMMY PS
3218 005156 012746 005164 MOV #64,$-(SP) ;:AND PC
3219 005162 000006 RTT ;:TRY THE RTT
3220 005164 012767 000006 025460 64$: MOV #RTT,$RTRN ;:RTT IS LEGAL--SET $RTRN TO A RTT
3221 005172 000402 BR 66$
3222 005174 062706 000010 65$: ADD #10,SP ;:RTT ILLEGAL--CLEAN OFF THE STACK
3223 005200 012737 000012 000010 66$: MOV #RESVEC+2,@#RESVEC ;:RESTORE TRAP CATCHER
3224 005206 005067 025446 CLR $TBIT ;:CLEAR 'T' BIT SWITCH
3225 005212 012767 005212 173666 MOV #,$SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
3226 005220 012767 005220 173662 MOV #,$SLPERR ;:SETUP THE ERROR LOOP ADDRESS
3227
3228 ;:*****
3229 005226 005227 177777 INC #-1 ;:FIRST TIME?
3230 005232 001037 BNE 67$ ;:BRANCH IF NO
3231 005234 022737 032604 000042 CMP #SENDAD,@#42 ;:ACT-11?
3232 005242 001433 BEQ 67$ ;:BRANCH IF YES
3233 005244 104400 005252 TYPE 68$ ;:TYPE ASCIZ STRING
3234 005250 000430 BR 67$ ;:GET OVER THE ASCIZ
3235 ;:68$: .ASCIZ <CRLF>'CEKBEO PDP11/70-74MP CPU DIAGNOSTIC PART 2'<CRLF>
3236 005332 67$: INC #-1 ;:FIRST TIME?
3237 005332 005227 177777 BNE 100$ ;:BR IF NO
3238 005336 001000
3239 005340 100$:
3240 ;:*****
3241 ;:SAVE THE MONITOR IF INITIAL LOAD
3242 005340 005737 000042 LOOP: TST @#42 ;:RUNNING UNDER XXDP CHAIN?
3243 005344 001003 BNE 3$ ;:BRANCH IF YES
3244 005346 005227 177777 INC #-1 ;:INITIAL LOAD?
3245 005352 001010 BNE 2$ ;:BRANCH IF NO
3246 005354 012700 002734 3$: MOV #^D1500,R0 ;:SETUP SOB COUNT
```



```

3247 005360 012701 073312      MOV      #SUBTAB+2,R1      :GET END ADDRESS OF PROGRAM
3248 005364 012702 160000      MOV      #160000,R2      :GET END ADDRESS OF MONITOR
3249 005370 014221      1$: MOV      -(R2),(R1)+    :SAVE 1500 DECIMAL WORDS
3250 005372 077002      SOB      R0,1$
3251 005374 012737 034656 000060 2$: MOV      #QUIT,@#TKVEC   :SETUP KEYBOARD VECTOR
3252 005402 012737 000340 000062  MOV      #PR7,@#TKVEC+2  :SETUP KEYBOARD PSW
3253 005410 005077 173524      CLR      @#STKB          :ENSURE BUFFER CLEAR
3254 005414 152777 000100 173514  BISB     #BIT6,@#STKS     :SET INTERRUPT ENABLE BIT
3255 005422 012737 032666 000004  MOV      #CPUSPUR,@#ERRVEC :
3256 005430 012737 000340 000006  MOV      #PR7,@#ERRVEC+2  :
3257 005436 012737 033072 000114  MOV      #CACHSPU,@#CACHVEC :
3258 005444 012737 000340 000116  MOV      #PR7,@#CACHVEC+2  :
3259 005452 005037 177766      CLR      @#CPUERR        :ENSURE ERROR REGISTER CLEAR
3260 005456 012737 177777 177744  MOV      #-1,@#MEMERR     :ENSURE MEMORY ERROR REG CLEAR
3261 005464 005767 173410      TST      $PASS           :FIRST PASS?
3262 005470 001402      BEQ      TST1            :BRANCH IF YES
3263 005472 005037 177746      CLR      @#CONTRL        :ENABLE CACHE
3264
3265
3266      :*****
3267      :*TEST 1      SPL
3268      :*
3269      :* IF FORK A FAILS EXECUTION WILL GO TO ONE OF 3 STATES:
3270      :* RSD.02,SVC.70, OR D30.00.
3271      :* RSD.02 WILL CAUSE A TRAP TO LOCATION 10.
3272      :* SVC.70 WILL HANG THE PROCESSOR IN THE PAUSE STATE. THIS
3273      :* WILL ONLY OCCUR IF RACF E17(AFIRO4(1)*AFIRO5(0)*(RTS:CCOP))
3274      :* IS BAD.
3275      :* D30.00 WILL CAUSE A TRAP TO LOCATION 10 AFTER STATE D10.60.
3276      :* THIS FAILURE CAN BE DIFFERENTIATED FROM THE FIRST BY TESTING
3277      :* THE REGISTER ASSOCIATED WITH BITS <2:0> OF THE OP CODE TO
3278      :* SEE IF IT WAS INCREMENTED.
3279      :*
3280      :* IF BOTH LEVELS 2 AND 5 COME UP AS LEVEL 4, PDRD E31(1)
3281      :* COULD BE BAD.
3282      :*
3283      :* ONCE IT IS DETERMINED THAT THE INSTRUCTION WORKS A BIT TEST IS
3284      :* MADE ON THE PSW <7:5>.
3285      :*
3286      :* ROM FLOW-43,361
3287      :******
3287 005476 000004      TST1: SCOPE
3288 005500 012767 005660 173466  MOV      #TST2,$ESCAPE   :SAVE START ADDRESS OF NEXT TEST
3289 005506 012767 005660 173554  MOV      #TST2,NEXTTST   :SAVE START ADDRESS OF NEXT TEST
3290 005514 012706 001100      MOV      #STACK,SP      :INITIALIZE THE SP
3291 005520 005005      CLR      R5             :INITIALIZE ERROR RECORD
3292 005522 012737 005642 000010  MOV      #1$,@#RESVEC    :SETUP RESERVED VECTOR TO TRAP TO THIS ROUTINE
3293 005530 012737 000113 177770  MOV      #113,@#177770   :SETUP MICROPROCESSOR BREAK REG
3294 005536 000235      SPL                    :EXECUTE INSTRUCTION UNDER TEST
3295 005540 013767 177776 173440  MOV      @#PSW,$ERPSW    :SAVE PSW
3296 005546 042767 177437 173432  BIC      #177437,$ERPSW  :MASK OFF THE PRIORITY
3297 005554 022767 000240 173424  CMP      #PR5,$ERPSW    :DID PRIORITY LEVEL 5 GET SET?
3298 005562 001401      BEQ      2$             :BRANCH IF YES
3299 005564 005205      INC      R5             :SET ERROR RECORD
3300 005566 012737 000044 177770 2$: MOV      #44,@#177770   :SETUP MICROPROCESSOR BREAK REG
3301 005574 000240      NOP
3302 005576 000232      SPL                    :TEST TO SEE IF BITS 5&7 CLEAR & BIT 6 SETS

```

```
3303 005600 013767 177776 173354 MOV @#PSW,$TMP0 ;SAVE PSW
3304 005606 042767 177437 173346 BIC #177437,$TMP0 ;MASK OFF THE PRIORITY
3305 005614 022767 000100 173340 CMP #PR2,$TMP0 ;DID BIT 6 SET & 5&7 CLEAR?
3306 005622 001404 BEQ 3$ ;BRANCH IF YES
3307 005624 005705 TST R5 ;DID SPL 5 FAIL?
3308 005626 001401 BEQ 4$ ;BRANCH IF NO
3309 005630 104001 ERROR 1 ;EITHER SPL DOES NOT LOAD PSW OR BITS STUCK
3310 005632 104002 4$: ERROR 2 ;PR5 OK BUT PR2 BAD
3311 005634 005705 3$: TST R5 ;DID SPL 5 FAIL?
3312 005636 001410 BEQ TST2 ;BRANCH IF NO
3313 005640 104003 ERROR 3 ;PR2 OK BUT PR5 FAILED
3314 005642 012737 000012 000010 1$: MOV #12,@#RESVEC ;RESTORE RESERVEC VECTOR ADDRESS
3315 005650 005705 TST R5 ;DID R5 GET INCREMENTED?
3316 005652 001401 BEQ 5$ ;BRANCH OF NO
3317 005654 104004 ERROR 4 ;FORK A FAILED TO D30.00
3318 005656 104005 5$: ERROR 5 ;FORK A FAILED TO RSD.02
3319 *****
3320 *TEST 2 RESET
3321 *
3322 * IF FORK A FAILS EXECUTION WILL EITHER GO TO WAT.00 OR TRP.02.
3323 * THE LA30 PRINTER WILL BE STARTED SUCH THAT A FAILURE INTO
3324 * WAT.00 WILL RECOVER.
3325 *
3326 * IF TRP.01 IS ENTERED A TRAP SEQUENCE WILL EXECUTE
3327 * WITH A TRAP VECTOR OF 4.
3328 *
3329 * ROM FLOW-15,255,374
3330 *****
3331 005660 000004 TST2: SCOPE
3332 005662 012767 003674 173214 MOV #^D1980,$ICNT ;SET ITERATION COUNT
3333 005670 012767 006154 173276 MOV #TST3,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
3334 005676 012767 006154 173364 MOV #TST3,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3335 005704 013767 177776 173256 MOV @#PSW,$TMP3 ;SAVE PSW
3336 005712 012767 005742 173166 MOV #4$,$LPADR ;SETUP LOOP ADR
3337 005720 012767 005742 173162 MOV #4$,$LPERR ;SETUP ERROR LOOP
3338 005726 012737 006144 000064 MOV #2$,@#TPVEC ;PUT ADDRESS OF 2$ IN PRINTER INTERRUPT VEC
3339 005734 012737 006134 000004 MOV #1$,@#ERRVEC ;PUT ADDRESS OF 1$ IN LOCATION 4
3340 005742 012706 001100 4$: MOV #STACK,SP ;INITIALIZE THE SP
3341 ;TO CATCH A FAILURE TO THE TRP.02 STATE
3342 005746 000230 SPL 0 ;SET CPU AT 0
3343 005750 012777 000015 173166 MOV #15,@$TPB ;SEND A CR TO THE PRINTER
3344 005756 105777 173160 7$: TSTB @$TPS ;WAIT FOR CR TO FINISH
3345 005762 100375 BPL 7$ ;INCASE TP DOUBLE BUFFERED
3346 005764 012777 000101 173152 MOV #101,@$TPB ;SEND AN 'A'
3347 005772 052777 000100 173142 BIS #BIT6,@$TPS ;SET THE INTERRUPT FLAG
3348 006000 012737 034157 177776 MOV #34157,@#PSW ;SET AS MANY BITS AS POSSIBLE IN PSW
3349 006006 000240 NOP ;SYNC POINT
3350 006010 000005 RESET ;EXECUTE INSTRUCTION UNDER TEST
3351 006012 013767 177776 173166 MOV @#PSW,$ERPSW ;SAVE PSW
3352 006020 017700 173116 MOV @$TPS,R0 ;GET THE PRINTER STATUS
3353 006024 042777 000100 173112 BIC #BIT6,@$TPB ;CLEAR INTERRUPT FLAG INCASE RESET FAILED
3354 006032 032700 000100 BIT #BIT6,R0 ;DID INTERRUPT FLAG CLEAR?
3355 006036 001401 BEQ 3$ ;BRANCH IF YES
3356 006040 104006 ERROR 6 ;RESET DID NOT WORK
3357 006042 105777 173074 3$: TSTB @$TPS ;WAIT FOR CHARACTER TO FINISH
3358 006046 100375 BPL 3$
```

```

3359 006050 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
3360 006056 012767 034157 173076 MOV #34157,$TMP0 ;SAVE EXPECTED VALUE
3361 006064 032737 000020 177776 BIT #BIT4,@#PSW ;IS T BIT ON?
3362 006072 001407 BEQ 5$ ;BRANCH IF NO
3363 006074 012767 034177 173060 MOV #34177,$TMP0 ;SAVE EXPECTED VALUE
3364 006102 022767 034177 173076 CMP #34177,$ERPSW ;DID PSW COME OUT OK?
3365 006110 000403 BR 6$
3366 006112 022767 034157 173066 5$: CMP #34157,$ERPSW ;DID PSW CHANGE?
3367 006120 6$:
3368 006120 001415 BEQ TST3 ;:BRANCH IF NO
3369 006122 012767 034157 173032 MOV #34157,$TMP0 ;SAVE EXPECTED VALUE
3370 006130 104377 ERROR 377 ;PSW CHANGED ON RESET
3371 006132 000460 460
3372 006134 042777 000100 173000 1$: BIC #BIT6,@$TPS ;CLEAR PRINTER INTERRUPT FLAG
3373 006142 104007 ERROR 7 ;FORK A FAILED INTO TRP.02
3374 006144 042777 000100 172770 2$: BIC #BIT6,@$TPS ;CLEAR PRINTER INTERRUPT FLAG
3375 006152 104010 ERROR 10 ;FORK A FAILED TO WAT.00
    
```

\*\*\*\*\*  
 \*TEST 3 MARK

\*  
 \* FORK A CAN FAIL INTO ONE OF THE FOLLOWING:  
 \* RSD.00, MFP.80, MTP.00, SVC.70, OR D67.01.  
 \* STATE RSD.00 WILL CAUSE A TRAP TO LOCATION 10.  
 \* MFP.80 WILL EXECUTE AN MFP INSTRUCTION.  
 \* MTP.00 WILL EXECUTE AN MTP INSTRUCTION EXCEPT FOR THE ALU.  
 \* SVC.70 WILL HANG THE PROCESSOR IN THE PAUSE CONDITION.  
 \* THIS WILL ONLY HAPPEN IF RACF E8 IS BAD.  
 \* D67.01 WILL STEP THE PCB AND TRAP TO LOCATION 10 AFTER STATE D10.60.  
 \*  
 \* ROM FLOW-47,252,235,234

```

3390 *****
3391 006154 000004 TST3: SCOPE
3392 006156 152777 000100 172752 BISB #BIT6,@$TKS ;RESTORE INTER FLAG AFTER RESET
3393 006164 012767 006252 172714 MOV #9$,$LPADR ;SETUP LOOP ADDRESS
3394 006172 012767 006252 172710 MOV #9$,$LPERR ;SETUP ERROR LOOP
3395 006200 013767 177776 172762 MOV @#PSW,$TMP3 ;SAVE PSW
3396 006206 032737 000020 177776 BIT #BIT4,@#PSW ;IS T BIT ON?
3397 006214 001416 BEQ 9$ ;BRANCH IF NO
3398 006216 012746 000340 MOV #PR7,-(SP) ;PUT NEW PSW ON STACK
3399 006222 012746 006252 MOV #9$,-(SP) ;PUT RETURN ADDR ON STACK
3400 006226 000006 RTT ;TURN T BIT OFF
3401 006230 012767 006436 172736 MOV #TST4,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
3402 006236 012767 006436 173024 MOV #TST4,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3403 006244 012737 006406 000010 MOV #2$,@#RESVEC ;SETUP LOC 10 TO TRAP TO THIS ROUTINE
3404 006252 012706 001100 9$: MOV #STACK,SP ;INITIALIZE STACK
3405 006256 012746 100000 MOV #BIT15,-(SP) ;SET SIGN BIT ON STACK
3406 006262 012766 100000 177776 MOV #BIT15,-2(SP) ;SET SIGN BIT AT LOCATION 1074
3407 006270 012705 006326 MOV #1$,R5 ;SETUP R5 FOR MARK INSTRUCTION
3408 006274 000240 NOP ;SYNC POINT
3409 006276 006401 MARK 1 ;EXECUTE INSTRUCTION UNDER TEST
3410 006300 022701 006300 8$: CMP #8$,R1 ;DID MTP OCCUR?
3411 006304 001401 BEQ 3$ ;BRANCH IF NO
3412 006306 104011 5$: ERROR 11 ;FORK A FAILED TO MTP
3413 006310 013701 001074 3$: MOV @#1074,R1 ;DID MFP OCCUR?
3414 006314 100401 BMI 4$ ;BRANCH IF NO
    
```

```

3415 006316 104012          ERROR 12          ;FORK A FAILED TO MFP
3416 006320 012706 001076 4$:  MOV #1076,SP    ;INITIALIZE SP INCASE MARK CHANGED IT
3417 006324 104013          ERROR 13          ;PCB DID NOT LOAD FROM R5
3418 006326 020627 006304 1$:  CMP SP,#5$-2    ;DID SP GET LOADED PROPERLY?
3419 006332 001410          BEQ 6$           ;BRANCH IF YES
3420 006334 010667 172614          MOV SP,$REG0    ;SAVE SP FOR TYPEOUT
3421 006340 012767 006306 172610  MOV #5$,$REG1   ;STORE ADDRESS OF 5$ FOR TYPEOUT
3422 006346 012706 001100          MOV #STACK,SP  ;REINITIALIZE SP
3423 006352 104014          ERROR 14          ;MARK DID NOT LOAD SP PROPERLY
3424 006354 012706 001100 6$:  MOV #STACK,SP  ;RESTORE THE SP
3425 006360 020527 006300          CMP R5,#8$     ;DID R5 GET LOADED PROPERLY?
3426 006364 001424          BEQ TST4       ;BRANCH IF YES
3427 006366 010567 172562          MOV R5,$REG0   ;SAVE R5 FOR TYPEOUT
3428 006372 013767 006304 172556  MOV @#5$-2,$REG1 ;STORE ADDRESS OF 5$-2 FOR TYPEOUT
3429 006400 012706 001076          MOV #1076,SP   ;RESTORE THE SP
3430 006404 104015          ERROR 15          ;R5 DID NOT LOAD PROPERLY
3431 006406 012737 000012 000010 2$:  MOV #12,@#RESVEC ;RESTORE RESERVED INSTR VECTOR
3432 006414 021627 006300          CMP (SP),#8$   ;DID PCB GET INCREMENTED BY D67.01?
3433 006420 001003          BNE 7$        ;BRANCH IF NO
3434 006422 012706 001100          MOV #STACK,SP  ;RESTORE THE SP
3435 006426 104016          ERROR 16          ;FORK A FAILED TO RSD.00
3436 006430 012706 001100 7$:  MOV #STACK,SP  ;RESTORE THE SP
3437 006434 104017          ERROR 17          ;FORK A FAILED TO D67.01
3438
3439 :*****
3440 :*TEST 4 ASH*DMO
3441 :*
3442 :* IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
3443 :*
3444 :* IF GRAJ SC05 L DOES NOT GO LOW OR DOES NOT GET THRU TO
3445 :* RACK BRCAB04 L A SHIFT RIGHT WILL OCCUR.
3446 :* IF THE SHIFT COUNTER DOES NOT SHIFT OR GRAJ SC=0 DOES NOT GO
3447 :* LOW THE PROCESSOR WILL HANG UP IN STATE ASH.41.
3448 :*
3449 :* ROM FLOW-52,305,257,166 LEFT SHIFT
3450 :* 52,305,277 RIGHT SHIFT
3451 :*****
3451 006436 000004 TST4: SCOPE
    
```

```
3452 006440 012767 007200 172622      MOV      #TST5,NEXTTST      ;SAVE ADDRESS OF NEXT TEST
3453 006446 012767 006512 172432      MOV      #19$, $LPADR      ;SETUP LOOP ADR
3454 006454 012767 006512 172426      MOV      #19$, $LPERR      ;SETUP ERROR LOOP
3455 006462 032767 000020 172500      BIT      #BIT4, $TMP3      ;WAS T BIT ON?
3456 006470 001405                BEQ      18$                ;BRANCH IF NO
3457 006472 012746 000360                MOV      #360, -(SP)        ;PUT NEW PSW ON STACK
3458 006476 012746 006504                MOV      #18$, -(SP)        ;PUT RETURN ADR ON STACK
3459 006502 000006                RTT                          ;TURN T BIT ON
3460
3461                ;
3462 006504 012737 007066 000010      ;ARITHMETIC LEFT SHIFT
18$: MOV      #1$, @WRESVEC      ;SET UP FOR FORK A FAILURE
19$: MOV      #STACK, SP        ;INITIALIZE THE SP
3463 006512 012706 001100                MOV      #177701, R0        ;PUT SHIFT COUNT IN R0 (+1)
3464 006516 012700 177701                MOV      #BIT15, R1        ;SET BIT 15 IN REGISTER TO BE SHIFTED
3465 006522 012701 100000                MOV      #BIT15, R1        ;SET BIT 15 IN REGISTER TO BE SHIFTED
3466 006526 000240                NOP                          ;SYNC POINT
3467 006530 072100                ASH      R0, R1              ;EXECUTE INSTRUCTION UNDER TEST
3468 006532 103414                BCS      2$                 ;BRANCH IF SHIFT WORKED
3469 006534 020127 140000                CMP      R1, #140000        ;DID SHIFT GO IN WRONG DIRECTION?
3470 006540 001002                BNE      3$                 ;BRANCH IF NO
3471 006542 104020                ERROR    20                 ;SHIFTED RIGHT INSTEAD OF LEFT
3472 006544 000425                BR       9$                 ;
3473 006546 020127 100000      3$: CMP      R1, #BIT15      ;DID R1 SHIFT AT ALL?
3474 006552 001002                BNE      4$                 ;BRANCH IF YES
3475 006554 104021                ERROR    21                 ;R1 DID NOT SHIFT
3476 006556 000420                BR       9$                 ;
3477 006560 104022                4$: ERROR    22                 ;R1 SHIFTED BUT CARRY DID NOT SET
3478                ;SHIFT COUNTER COULD BE STUCK
3479 006562 000416                BR       9$                 ;
3480 006564 102003                2$: BVC      10$              ;BRANCH IF V DID NOT SET
3481 006566 100402                BMI      10$              ;BRANCH IF N DID NOT CLEAR
3482 006570 001001                BNE      10$              ;BRANCH IF Z DID NOT SET
3483 006572 000412                BR       9$                 ;CC'S OK
3484 006574 013767 177776 172404      10$: MOV      @WPSW, $ERPSW    ;SAVE PSW
3485 006602 042767 177760 172376      BIC      #177760, $ERPSW    ;MASK OFF CC'S
3486 006610 012767 000007 172344      MOV      #7, $TMP0          ;PUT EXPECTED CC'S IN STORAGE
3487 006616 104031                ERROR    31                 ;STATE ASH.40 DID NOT LOAD CC'S CORRECTLY
3488                ;*****
3489                ;ARITHMETIC RIGHT SHIFT TEST
3490 006620 012767 006626 172262      9$: MOV      #64$, $LPERR      ;SETUP ERROR LOOP
3491 006626 012701 100001      64$: MOV      #100001, R1      ;SETUP R1 FOR RIGH SHIFT
3492 006632 012700 000077                MOV      #77, R0           ;PUT SHIFT COUNT IN R0 (-1)
3493 006636 005002                CLR      R2                 ;ENSURE R2 CLEAR
3494 006640 000240                NOP                          ;SYNC POINT
3495 006642 072100                ASH      R0, R1              ;EXECUTE RIGHT SHIFT
3496 006644 005502                ADC      R2                 ;SAVE CARRY IN R2
3497 006646 020127 140000                CMP      R1, #140000        ;DID R1 SHIFT IN RIGHT DIRECTION?
3498 006652 001417                BEQ      5$                 ;BRANCH IF YES
3499 006654 005701                TST      R1                 ;DID R1 SHIFT LEFT INSTEAD OF RIGHT?
3500 006656 001002                BNE      6$                 ;BRANCH IF NO
3501 006660 104023                ERROR    23                 ;R1 SHIFTED WRONG DIRECTION
3502 006662 000416                BR       8$                 ;
3503 006664 022701 040000      6$: CMP      #40000, R1        ;DID SHIFT FAIL TO SIGN FILL?
3504 006670 001001                BNE      7$                 ;BRANCH IF NO
3505 006672 104024                ERROR    24                 ;SHIFT RIGHT DID NOT SIGN FILL
3506 006674 010167 172256      7$: MOV      R1, $REG1        ;SAVE R1 FOR TYPE OUT
3507 006700 012767 140000 172256      MOV      #140000, $TMP1     ;PUT EXPECTED VALUE IN STORAGE
```

```
3508 006706 104025          ERROR 25          ;R1 SHIFTED, BUT DON'T KNOW WHERE
3509 006710 000403          BR      8$
3510 006712 005702          5$:  TST  R2          ;DID CARRY GET SET ON SHIFT?
3511 006714 001001          BNE    8$          ;BRANCH IF YES
3512 006716 104026          ERROR 26          ;SHIFT OK BUT CARRY DID NOT LOAD
3513                                     ;*****
3514                                     ;CONDITION CODE LOAD TEST(STATE ASH.30)
3515 006720 012767 006726 172162 8$:  MOV  #63$, $LPERR ;SETUP ERROR LOOP
3516 006726 012701 100000 63$:  MOV  #BIT15,R1    ;SETUP R0 TO TEST CC'S IN STATE ASH.30
3517                                     ;R0 HAS SHIFT COUNT (-1)
3518 006732 000250          CLN
3519 006734 000266          +SEV.SEZ        ;SET UP CC'S TO
3520                                     ;COMPLIMENT OF EXPECTED RESULT
3521 006736 072100          ASH  R0,R1        ;AND OSCILLOSCOPE SYNC POINT
3522 006740 013767 177776 172240 MOV  @PSW,$ERPSW  ;EXECUTE THE SHIFT
3523 006746 042767 177760 172232 BIC  #177760,$ERPSW ;SAVE PSW
3524 006754 022767 000010 172224 CMP  #10,$ERPSW   ;MASK OFF THE CC'S
3525 006762 001404          BEQ   12$         ;DID CC'S COME OUT OK?
3526 006764 012767 000010 172170 MOV  #10,$TMP0    ;BRANCH IF YES
3527 006772 104072          ERROR 72         ;PUT EXPECTED VALUE IN STORAGE
3528                                     ;STATE ASH.30 DID NOT LOAD CC'S CORRECTLY
3529                                     ;*****
3530                                     ;SHIFT COUNT=0
3531 006774 012767 007002 172106 12$:  MOV  #62$, $LPERR ;SETUP ERROR LOOP
3532 007002 012701 100000 62$:  MCV  #BIT15,R1    ;SET SIGN BIT IN REGISTER TO BE SHIFTED
3533 007006 005000          CLR   R0          ;SET SHIFT COUNT TO ZERO
3534 007010 000263          +SEC!SEV        ;SETUP CC'S TO COMPLIMENT
3535 007012 000250          CLN              ;OF EXPECTED VALUE
3536                                     ;AND OSCILLOSCOPE SYNC POINT
3537 007014 072100          ASH  R0,R1        ;EXECUTE INSTRUCTION
3538 007016 013767 177776 172162 MOV  @PSW,$ERPSW  ;SAVE PSW
3539 007024 042767 177760 172154 BIC  #177760,$ERPSW ;MASK OFF THE CC'S
3540 007032 022767 000010 172146 CMP  #10,$ERPSW   ;DID CC'S COME OUT OK?
3541 007040 001416          BEQ   16$         ;BRANCH IF YES
3542 007042 022701 100000 13$:  CMP  #BIT15,R1    ;DID R1 SHIFT?
3543 007046 001005          BNE   15$         ;BRANCH IF YES
3544 007050 012767 000010 172104 MOV  #10,$TMP0    ;SAVE EXPECTED VALUE
3545 007056 104027          ERROR 27         ;STATE ASH.20 DID NOT LOAD CC'S CORRECTLY
3546 007060 000406          BR    16$
3547 007062 104030          15$:  ERROR 30          ;R1 SHIFTED WHEN NOT SUPPOSE TO
3548 007064 000404          BR    16$
3549 007066 012737 000012 000010 1$:  MOV  #12,@RESVEC ;RESTORE RESERVED VECTOR
3550 007074 104032          ERROR 32         ;FORK A FAILED
3551                                     ;*****
3552                                     ;CONDITION CODE TEST OF STATE ASH.41
3553 007076 012767 007104 172004 16$:  MOV  #61$, $LPERR ;SETUP ERROR LOOP
3554 007104 012700 000002 61$:  MOV  #2,R0        ;PUT SHIFT COUNT IN R0
3555 007110 012701 060000          MOV  #60000,R1   ;TO TEST STATE ASH.41
3556 007114 000274          +SEN!SEZ        ;SETUP R1 FOR SHIFT LEFT
3557 007116 000243          +CLV!CLC        ;SETUP CC'S TO COMPLEMENT
3558                                     ;OF EXPECTED VALUE
3559                                     ;AND OSCILLOSCOPE SYNC POINT
3560 007120 072100          ASH  R0,R1        ;EXECUTE INSTRUCTION UNDER TEST
3561 007122 013767 177776 172056 MOV  @PSW,$ERPSW  ;SAVE PSW
3562 007130 020127 100000          CMP  R1,#BIT15   ;DID R1 SHIFT CORRECTLY?
3563 007134 001406          BEQ   17$         ;BRANCH IF YES
3564 007136 010167 172014          MOV  R1,$REG1    ;SAVE R1 FOR TYPEOUT
```

```
3564 007142 012767 100000 172014      MOV    #BIT15,$TMP1    ;STORE EXPECTED VALUE
3565 007150 104073                ERROR  73              ;STATE ASH.41 FAILED
3566 007152 042767 177760 172026 17$:  BIC    #177760,$ERPSW  ;MASK OFF CC'S
3567 007160 022767 000013 172020      CMP    #13,$ERPSW     ;DID CC'S LOAD CORRECTLY?
3568 007166 001404                BEQ    TST5            ;:BRANCH IF YES
3569 007170 012767 000013 171764      MOV    #13,$TMP0      ;STORE EXPECTED VALUE
3570 007176 104074                ERROR  74              ;BAD CC'S DUE TO ASH.41
3571
3572      ;*****
3573      ;*TEST 5      ASH*DM1
3574      ;*
3575      ;*      IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
3576      ;*
3577      ;*      IF FORK B FAILS EXECUTION WILL EITHER GO TO RSD.00 OR
3578      ;*      MUL.00.
3579      ;*      MUL.00 WILL ONLY BE ENTERED IF EITHER B FORK MUX INPUT B2
3580      ;*      DOES NOT GO HIGH OR THE MUX IS BAD.
3581      ;*
3582      ;*      ROM FLOW-1,175,62,52,305,257
3583      ;*      ;*****
3584      ;*      TST5:  SCOPE
3585      ;*      MOV    #TST6,$ESCAPE  ;SAVE START ADDRESS OF NEXT TEST
3586      ;*      MOV    #TST6,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3587      ;*      MOV    #STACK,SP      ;INITIALIZE THE STACK
3588      ;*      MOV    #1,$@#RESVEC   ;SETUP RESERVED VECTOR
3589      ;*      MOV    #TMP1,R0       ;PUT ADDRESS OF SHIFT COUNT IN R0
3590      ;*      MOV    #1,(R0)        ;SET SHIFT COUNT TO ONE
3591      ;*      MOV    #BIT15,R1      ;SETUP REGISTER TO BE SHIFTED
3592      ;*      NOP                    ;OSCILLOSCOPE SYNC POINT
3593      ;*      ASH    (R0),R1        ;EXECUTE INSTRUCTION UNDER TEST
3594      ;*      BCS    TST6            ;:TEST OK, GO TO NEXT TEST
3595      ;*      TST    R1              ;DID MULTIPLY OCCUR?
3596      ;*      BMI    2$              ;BRANCH IF YES
3597      ;*      ERROR  33              ;STATE ASH.00 FAILED
3598      ;*      ERROR  34              ;FORK B FAILED INTO MUL.00
3599      ;*      MOV    #3,$@#RESVEC   ;SETUP RESVEC
3600      ;*      CLR    R0              ;PUT EVEN ADDRESS IN R0
3601      ;*      NOP                    ;OSCILLOSCOPE SYNC POINT
3602      ;*      ASH    (R0)+,R1       ;EXECUTE DM2
3603      ;*      MOV    #12,$@#RESVEC  ;RESTORE RESERVED VECTOR
3604      ;*      TST    R0              ;DID R0 AUTO INCREMENT?
3605      ;*      BEQ    4$              ;BRANCH IF NO
3606      ;*      ERROR  35              ;FORK B FAILED
3607      ;*      ERROR  36              ;FORK A FAILED
3608      ;*      ;*****
3609      ;*      ;*TEST 6      ASH*DM2
3610      ;*
3611      ;*      IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
3612      ;*
3613      ;*      ALL OTHER LOGIC HAS BEEN TESTED
3614      ;*
3615      ;*      ROM FLOW-2,175,62,52,305
3616      ;*      ;*****
3617      ;*      TST6:  SCOPE
3618      ;*      MOV    #TST7,$ESCAPE  ;SAVE START ADDRESS OF NEXT TEST
3619      ;*      MOV    #TST7,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3619      ;*      MOV    #STACK,SP      ;INITIALIZE THE SP
```

```
3620 007336 012737 007364 000010      MOV    #1$,@#RESVEC    ;PUT ADDRESS OF 1$ IN RESERVED VECTOR
3621 007344 012700 001164              MOV    #STMP1,RO      ;PUT ADDRESS OF SHIFT COUNT IN RO
3622 007350 005010              CLR    (RO)           ;PUT SHIFT COUNT OF ZERO IN $TMP1
3623 007352 072120              ASH    (RO)+,R1      ;EXECUTE INSTRUCTION UNDER TEST
3624 007354 012737 000012 000010      MOV    #12,@#RESVEC  ;RESTORE RESVEC
3625 007362 000404              BR     TST7          ;GO TO NEXT TEST
3626 007364 012737 000012 000010 1$:  MOV    #12,@#RESVEC  ;RESTORE RESERVED VECTOR
3627 007372 104037              ERROR  37            ;FORK A FAILED
3628
3629      :*****
3630      :*TEST 7      ASH*DM4
3631      :*
3632      :*      IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
3633      :*
3634      :*      ALL OTHER LOGIC HAS BEEN TESTED.
3635      :*
3636      :*      ROM FLOW-4,122,177,62,52,305
3637      :*****
3637 007374 000004      TST7:  SCOPE
3638 007376 012767 007460 171570      MOV    #TST10,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
3639 007404 012767 007460 171656      MOV    #TST10,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3640 007412 012706 001100              MOV    #STACK,SP     ;INITIALIZE THE SP
3641 007416 012737 007450 000010      MOV    #1$,@#RESVEC  ;PUT ADDRESS OF 1$ IN RESERVED VECTOR
3642 007424 012700 001166      MOV    #STMP2,RO     ;PUT ADDRESS OF SHIFT COUNT+2 IN RO
3643 007430 005060 177776      CLR    -2(RO)        ;PUT SHIFT COUNT IN $TMP1
3644 007434 000240              NOP                   ;OSCILLOSCOPE SYNC POINT
3645 007436 072140              ASH    -(RO),R1      ;EXECUTE INSTRUCTION UNDER TEST
3646 007440 012737 000012 000010      MOV    #12,@#RESVEC  ;RESTORE RESVEC
3647 007446 000404              BR     TST10         ;GO TO NEXT TEST
3648 007450 012737 000012 000010 1$:  MOV    #12,@#RESVEC  ;RESTORE RESVEC
3649 007456 104040              ERROR  40            ;FORK A FAILED
3650
3651      :*****
3652      :*TEST 10     ASHC*DM0
3653      :*
3654      :*      NEITHER FORK A NOR BEN03 SHOULD FAIL.
3655      :*
3656      :*      IF THE INSTRUCTION FAILS, ONE OF THE ASC STATES IS BAD.
3657      :*
3658      :*      ONCE IT IS DETERMINED THAT THE INSTRUCTION WORKS,
3659      :*      A BIT STUCK TEST IS PERFORMED ON THE SHIFT COUNTER.
3660      :*
3661      :*      ROM FLOW-53,306,267,227      RIGHT SHIFT
3662      :*      53,306,247,176,136      LEFT SHIFT
3663      :*      53,306,207                NO SHIFT
3664      :*****
3664 007460 000004      TST10: SCOPE
3665 007462 012767 010302 171600      MOV    #TST11,NEXTTST ;SAVE ADDRESS OF NEXT TEST
3666
3667      :ARITHMETIC RIGHT SHIFT COMBINED
3668 007470 012700 100001      MOV    #100001,RO    ;SETUP RO FOR RIGHT SHIFT
3669 007474 012701 000001      MOV    #BIT0,R1      ;SETUP R1 FOR RIGH SHIFT
3670 007500 012702 000077      MOV    #77,R2        ;PUT SHIFT COUNT (-1) IN R2
3671 007504 000262              SEV                   ;ENSURE V SET
3672
3673 007506 073002              ASHC   R2,RO         ;AND OSCILLOSCOPE SYNC POINT
3674 007510 013767 177776 171470      MOV    @#PSW,$ERPSW  ;SAVE PSW
3675 007516 042767 177760 171462      BIC    #177760,$ERPSW ;MASK OFF THE CC'S
```



```
3676 007524 022767 000011 171454      CMP      #11,$ERPSW      ;DID CC'S LOAD PROPERLY?
3677 007532 001014      BNE      1$           ;BRANCH IF NO
3678 007534 005701      TST      R1           ;DID R1 GET SIGN BIT SET
3679 007536 100017      BPL      2$           ;BRANCH IF NO
3680 007540 022700 140000      CMP      #140000,R0    ;DID R0 SIGN FILL?
3681 007544 001427      BEQ      3$           ;BRANCH IF YES
3682 007546 012767 140000 171406      MOV      #140000,$TMP0 ;EXPECTED VALUE
3683 007554 010067 171374      MOV      R0,$REG0     ;SAVE R0 FOR TYPEOUT
3684 007560 104041      ERROR   41           ;R0 DID NOT SIGN FILL
3685 007562 000420      BR       3$
3686 007564 012767 000011 171370 1$:      MOV      #11,$TMP0    ;EXPECTED VALUE
3687 007572 104042      ERROR   42           ;BAD CC'S
3688 007574 000413      BR       3$
3689 007576 012767 140000 171356 2$:      MOV      #140000,$TMP0 ;GET EXPECTED VALUES
3690 007604 012767 100000 171352      MOV      #100000,$TMP1 ;FOR TYPEOUT
3691 007612 010067 171336      MOV      R0,$REG0     ;SAVE R0 & R1 FOR TYPEOUT
3692 007616 010167 171334      MOV      R1,$REG1
3693 007622 104043      ERROR   43           ;R0<0> DID NOT GO TO R1<15>
3694
3695      ;*****
3695      ;ARITHMETIC LEFT SHIFT
3696 007624 012767 007632 171256 3$:      MOV      #64,$SLPERR  ;SETUP ERROR LOOP
3697 007632 012700 100000 64$:      MOV      #BIT15,R0    ;SETUP R0 FOR LEFT SHIFT
3698 007636 012701 100001      MOV      #100001,R1   ;SETUP R1 FOR LEFT SHIFT
3699 007642 012702 000001      MOV      #BIT0,R2     ;PUT SHIFT COUNT (+1) IN R2
3700 007646 000274      +SEZ!SEN             ;ENSURE Z AND N SET
3701
3702      ;AND OSCILLOSCOPE SYNC POINT
3702 007650 073002      ASHC    R2,R0         ;EXECUTE INSTRUCTION UNDER TEST
3703 007652 013767 177776 171326      MOV      @PSW,$ERPSW  ;SAVE PSW
3704 007660 042767 177760 171320      BIC      #177760,$ERPSW ;MASK OFF THE CC'S
3705 007666 010067 171262      MOV      R0,$REG0     ;SAVE REG0
3706 007672 010167 171260      MOV      R1,$REG1     ;SAVE REG1
3707 007676 012767 000001 171256      MOV      #1,$TMP0     ;GET EXPECTED VALUES
3708 007704 012767 000002 171252      MOV      #2,$TMP1     ;OF REGISTERS
3709 007712 022767 000003 171266      CMP      #3,$ERPSW    ;ARE CC'S CORRECT?
3710 007720 001010      BNE      7$           ;BRANCH IF NO
3711 007722 022701 000002      CMP      #2,R1        ;DID R1 SHIFT PROPERLY?
3712 007726 001012      BNE      11$          ;BRANCH IF NO
3713 007730 022700 000001      CMP      #1,R0        ;DID R0 SHIFT PROPERLY?
3714 007734 001410      BEQ      12$          ;BRANCH IF YES
3715 007736 104044      ERROR   44           ;R0 DID NOT GET SHIFTED PROPERLY
3716 007740 000406      BR       12$
3717 007742 012767 000003 171212 7$:      MOV      #3,$TMP0     ;SAVE EXPECTED VALUE
3718 007750 104045      ERROR   45           ;BAD CC'S
3719 007752 000401      BR       12$
3720 007754 104046      ERROR   46           ;R1 DID NOT SHIFT PROPERLY
3721
3722      ;*****
3722      ;NO SHIFT
3723 007756 012767 007764 171124 12$:      MOV      #63,$SLPERR  ;SETUP ERROR LOOP
3724 007764 012701 040000 63$:      MOV      #40000,R1    ;SETUP R1 FOR NO SHIFT
3725 007770 005002      CLR     R2           ;PUT SHIFT COUNT (0) IN R2
3726 007772 012700 100000      MOV      #BIT15,R0    ;SET SIGN BIT IN R0 & SET N
3727 007776 000277      SCC
3728 010000 000250      CLN
3729      ;ENSURE ALL CC'S SET
3729      ;ENSURE N CLEAR
3729      ;AND OSCILLOSCOPE SYNC POINT
3730 010002 073002      ASHC    R2,R0         ;EXECUTE INSTRUCTION UNDER TEST
3731 010004 013767 177776 171174      MOV      @PSW,$ERPSW  ;SAVE PSW
```

```
3732 010012 012767 000010 171142      MOV      #10,$TMP0      ;SAVE EXPECTED VALUE
3733 010020 042767 177760 171160      BIC      #177760,$ERPSW ;MASK OFF THE CC'S
3734 010026 022767 000010 171152      CMP      #10,$ERPSW    ;DID THE CC'S COME OUT CORRECT?
3735 010034 001401                BEQ      14$           ;BRANCH IF YES
3736 010036 104047                13$:      ERROR      47           ;BAD CC'S ON NO SHIFT
3737                ;*****
3738                ;CHECK ROTATE
3739 010040 012767 010046 171042      14$:      MOV      #62$,$LPERR    ;SETUP ERROR LOOP
3740 010046 012701 100000      62$:      MOV      #BIT15,R1     ;SETUP R1 FOR A ROTATE
3741 010052 010700                MOV      PC,R0        ;PUT RANDOM NUMBER IN R0
3742 010054 010067 171102      MOV      R0,$TMP0     ;SAVE R0
3743 010060 012702 177761      MOV      #-17,R2      ;PUT SHIFT COUNT (-17) IN R2
3744 010064 012767 000001 171072      MOV      #1,$TMP1     ;EXPECTED VALUE OF R1 IN $TMP1
3745 010072 000240                NOP                    ;OSCILLOSCOPE SYNC POINT
3746 010074 073102      ASHC     R2,R1        ;EXECUTE INSTRUCTION UNDER TEST
3747 010076 022701 000001      CMP      #1,R1        ;DID R1 ROTATE CORRECTLY?
3748 010102 001405                BEQ      15$           ;BRANCH IF YES
3749 010104 010067 171044      MOV      R0,$REG0     ;SAVE R0 FOR TYPEOUT
3750 010110 010167 171042      MOV      R1,$REG1     ;SAVE R1 FOR TYPEOUT
3751 010114 104050                ERROR      50           ;R1 DID NOT ROTATE PROPERLY
3752                ;*****
3753                ;THE FOLLOWING CODE CHECKS THE BITS IN THE SC
3754 010116 012767 010124 170764      15$:      MOV      #61$,$LPERR    ;SETUP ERROR LOOP
3755 010124 012700 000040      61$:      MOV      #40,R0        ;SETUP R1 FOR RIGHT SHIFT
3756 010130 012702 000052      MOV      #52,R2        ;PUT SHIFT COUNT (-26) IN R2
3757 010134 005001                CLR      R1            ;ENSURE R0 CLEAR
3758 010136 005067 171020      CLR      $TMP0        ;PUT EXPECTED VALUES OF
3759 010142 005067 171016      CLR      $TMP1        ;R0 & R1 IN STORAGE
3760 010146 012767 000001 171012      MOV      #1,$TMP2     ;C BIT EXPECTED
3761 010154 000240                NOP                    ;OSCILLOSCOPE SYNC POINT
3762 010156 073002      ASHC     R2,R0        ;EXECUTE SHIFT
3763 010160 013767 177776 171020      MOV      @PSW,$ERPSW  ;SAVE PSW
3764 010166 103410                BCS     16$           ;BRANCH IF CORRECT SHIFT
3765 010170 042767 177776 171010      BIC      #177776,$ERPSW ;MASK OFF C BIT
3766 010176 010067 170752      MOV      R0,$REG0
3767 010202 010167 170750      MOV      R1,$REG1
3768 010206 104051                ERROR      51           ;STUCK BITS IN SC
3769 010210 012767 010216 170672      16$:      MOV      #60$,$LPERR    ;SETUP ERROR LOOP
3770 010216 012701 004000      60$:      MOV      #4000,R1     ;SET UP R1 FOR LEFT SHIFT
3771 010222 005000                CLR      R0            ;ENSURE R0 CLEAR
3772 010224 012702 000025      MOV      #25,R2        ;PUT SHIFT COUNT (+25) IN R2
3773 010230 000240                NOP                    ;OSCILLOSCOPE SYNC POINT
3774 010232 073002      ASHC     R2,R0        ;EXECUTE SHIFT
3775 010234 013767 177776 170744      MOV      @PSW,$ERPSW
3776 010242 042767 177776 170736      BIC      #177776,$ERPSW ;MASK OFF C BIT
3777 010250 103414                BCS     TST11         ;GO TO NEXT TEST IF CORRECT SHIFT
3778 010252 012767 000000 170674      MOV      #R0,$REG0    ;SAVE R0 FOR TYPEOUT
3779 010260 012767 000001 170670      MOV      #R1,$REG1    ;SAVE R1 FOR TYPEOUT
3780 010266 005067 170670      CLR      $TMP0        ;SAVE EXPECTED
3781 010272 012767 000001 170664      MOV      #1,$TMP1     ;VALUES
3782 010300 104052                ERROR      52           ;STUCK BITS IN SC
3783                ;*****
3784                ;TEST 11      ASHC*DM1
3785                ;*
3786                ;*      THE ONLY POSSIBLE FAILURES ARE FORK B OR STATE ASC.00.
3787                ;*
```

```
3788 :* IF FORK B FAILS EXECUTION WILL EITHER GO TO RSD.00 OR
3789 :* ASH.00.
3790 :* ASH.00 WOULD PERFORM AN ASH INSTRUCTION INSTEAD OF AN ASHC.
3791 :*
3792 :* ROM FLOW-1,175,63,53,306,267,227
3793 :*****
3794 010302 000004 TST11: SCOPE
3795 010304 012767 010442 170662 MOV #TST12,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
3796 010312 012767 010442 170750 MOV #TST12,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
3797 010320 012706 001100 MOV #STACK,SP ;INITIALIZE THE SP
3798 010324 012737 010432 000010 MOV #1$,@#RESVEC ;SETUP RESERVED VECTOR
3799 010332 012702 001166 MOV #STMP2,R2 ;PUT ADDRESS OF SHIFT COUNT IN R2
3800 010336 012700 000001 MOV #1,R0 ;SETUP R0 FOR RIGHT SHIFT
3801 010342 005001 CLR R1 ;SETUP R1 FOR RIGHT SHIFT
3802 010344 012712 000077 MOV #77,(R2) ;PUT SHIFT COUNT (-1) IN STMP2
3803 010350 005067 170606 CLR $TMP0 ;PUT EXPECTED VALUES OF
3804 010354 012767 100000 170602 MOV #BIT15,$TMP1 ;R0 & R1 IN STORAGE
3805 010362 000240 NOP ;OSCILLOSCOPE SYNC POINT
3806 010364 073012 ASHC (R2),R0 ;EXECUTE INSTRUCTION UNDER TEST
3807 010366 103012 BCC 2$ ;BRANCH IF FORK B DID NOT FAIL
3808 010370 013767 177776 170610 MOV @#PSW,$ERPSW ;SAVE PSW FOR TYPEOUT
3809 010376 005067 170560 CLR $TMP0 ;SAVE EXPECTED VALUE
3810 010402 010067 170546 MOV R0,$REG0
3811 010406 010167 170544 MOV R1,$REG1
3812 010412 104053 ERROR 53 ;FORK B FAILED TO ASH.00
3813 010414 005701 2$: TST R1 ;DID R1 SIGN BIT SET?
3814 010416 100411 BMI TST12 ;BRANCH IF YES
3815 010420 010067 170530 MOV R0,$REG0 ;SAVE R0 & R1
3816 010424 010167 170526 MOV R1,$REG1 ;FOR TYPEOUT
3817 010430 104054 ERROR 54 ;STATE ASC.00 FAILED
3818 010432 012737 000012 000010 1$: MOV #12,@#RESVEC ;RESTORE RESVEC
3819 010440 104055 ERROR 55 ;FORK A FAILED TO RSD.00
3820 :*****
3821 :*TEST 12 MUL*DMO
3822 :*
3823 :* FORK A SHOULD NOT FAIL.
3824 :* THE FOLLOWING WOULD BE BEN11 FAILURES:
3825 :* IF EITHER GRAD DROO IS STUCK HIGH OR NOT GETTING THRU TO
3826 :* RACK E64(C1) OR RACK E64 IS BAD (INPUT C1 FAILED HIGH)
3827 :* THE MULTPLICAND WILL BE MULTITPLIED BY 177777.
3828 :* IF EITHER GRAD DROO IS STUCK LOW OR NOT GETTING THRU TO
3829 :* RACK E64(C1) OR RACK E64 IS BAD (INPUT C1 FAILED LOW)
3830 :* THE MULTPLICAND WILL BE MULTIPLIED BY ZERO.
3831 :* IF GRAJ SC=0 IS NOT GETTING TO RACK E50(C1) AND RACK E50
3832 :* IS BAD (INPUT C1 FAILED LOW) THE MULTIPLICAND WILL ONLY
3833 :* BE MULTIPLIED BY BIT 0 OF THE MULTIPLIER.
3834 :* IF RACK E50(C1) FAILS HIGH THE PROCESSOR WILL HANG UP IN
3835 :* STATE MUL.20(266).
3836 :* IF THE INSTRUCTION FAILS TO MULTIPLY CORRECTLY AND ONE
3837 :* OF THE ABOVE CONDITIONS CANNOT BE DETERMINED THEN THE
3838 :* FAILURE COULD BE IN THE INSTRUCTION DECODE ROM.
3839 :*
3840 :* IF THE CC'S COME UP BAD THEN THE FAILURE COULD BE IN
3841 :* STATES MUL.40,50, OR 60, OR IN THE CONDITION CODE ROM.
3842 :*
3843 :* ROM FLOW-50,102,266/246,226/206,310
```

```

3844
3845 010442 000004
3846 010444 012767 011004 170616
3847
3848
3849 010452 012700 000002
3850 010456 012702 000003
3851 010462 005001
3852 010464 005067 170472
3853 010470 012767 000006 170466
3854 010476 000277
3855
3856 010500 070002
3857 010502 013767 177776 170476
3858 010510 010067 170440
3859 010514 010167 170436
3860 010520 020127 000006
3861 010524 001002
3862 010526 005700
3863 010530 001423
3864 010532 020127 177776 1$:
3865 010536 001002
3866 010540 104056
3867 010542 000430
3868 010544 005701 3$:
3869 010546 001401
3870 010550 000404
3871 010552 005700 4$:
3872 010554 001002
3873 010556 104057
3874 010560 000421
3875 010562 020127 100000 5$:
3876 010566 001002
3877 010570 104060
3878 010572 000414
3879 010574 104061 6$:
3880 010576 000412
3881 010600 042767 177760 170400 2$:
3882 010606 022767 000000 170372
3883 010614 001403
3884 010616 005067 170340 7$:
3885 010622 104062
3886
3887
3888 010624 012767 010632 170256 8$:
3889 010632 012700 177777 64$:
3890
3891 010636 000277
3892 010640 000250
3893
3894 010642 070002
3895 010644 013767 177776 170334
3896 010652 010067 170276
3897 010656 010167 170274
3898 010662 012767 177777 170272
3899 010670 012767 177775 170266

```

```

:*****
TST12: SCOPE
MOV #TST13,NEXTTST ;SAVE ADDRESS OF NEXT TEST
;MULTIPLY 3 TIMES 2
MOV #2,R0 ;PUT MULTIPLICAND IN R0
MOV #3,R2 ;PUT MULTIPLIER IN R2
CLR R1 ;ENSURE R1 CLEAR
CLR $TMP0 ;PUT EXPECTED VALUE OF
MOV #6,$TMP1 ;R0 & R1 IN STORAGE
SCC ;MULTIPLY SHOULD CLEAR ALL OF THEM
;AND OSCILLOSCOPE SYNC POINT
MUL R2,R0 ;EXECUTE INSTRUCTION UNDER TEST
MOV @PSW,$ERPSW ;SAVE PSW
MOV R0,$REG0 ;SAVE R0 & R1
MOV R1,$REG1 ;FOR TYPEOUT
CMP R1,#6 ;DID R1 GET LOW PRODUCT
BNE 1$ ;BRANCH IF NO
TST R0 ;DID R0 GET UPPER PRODUCT?
BEQ 2$ ;BRANCH IF YES
CMP R1,#177776 ;DID R0 GET MULTIPLIED BY 177777?
BNE 3$ ;BRANCH IF NO
ERROR 56 ;EITHER GRAD DROO STUCK H OR RACK E64 BAD
BR 8$
3$: TST R1 ;DID R1 STAY ZERO?
BEQ 4$ ;BRANCH IF YES
BR 5$ ;CONTINUE ERROR ANALYSIS
4$: TST R0 ;DID R1 & R0 GO TO ZERO?
BNE 5$ ;BRANCH IF NO
ERROR 57 ;EITHER GRAD DROO STUCK L OR RACK E64 BAD
BR 8$
5$: CMP R1,#BIT15 ;DID R0 ONLY GET MULTIPLIED BY BIT 0?
BNE 6$ ;BRANCH IF NO
ERROR 60 ;RACK E50(C1) FAILED LOW
BR 8$
6$: ERROR 61 ;CAN'T DETERMINE WHAT HAPPENED
BR 8$
2$: BIC #177760,$ERPSW ;MASK OFF THE CC'S
CMP #0,$ERPSW
BEQ 8$ ;BRANCH IF YES
7$: CLR $TMP0 ;PUT EXPECTED PSW IN $TMO
ERROR 62 ;BAD CC'S
:*****
;THE FOLLOWING SECTION TESTS STATE MUL.50 (-1 TIMES 3)
8$: MOV #64,$LPERR ;SETUP ERROR LOOP
64$: MOV #177777,R0 ;PUT -1 (MULTIPLICAND) IN R0
;R2 HAS MULTIPLIER (+3)
;
;CC'S=0111
;AND OSCILLOSCOPE SYNC POINT
MUL R2,R0 ;EXECUTE INSTRUCTION UNDER TEST
MOV @PSW,$ERPSW ;SAVE PSW
MOV R0,$REG0 ;SAVE R0 & R1
MOV R1,$REG1 ;FOR TYPEOUT
MOV #-1,$TMP0 ;SAVE EXPECTED
MOV #-3,$TMP1 ;VALUES

```



3956 011110 012737 000012 000010 1\$: MOV #12,@#RESVEC ;RESTORE RESVEC  
3957 011116 104071 ERROR 71 ;FORK B FAILED

3958  
3959  
3960  
3961  
3962  
3963  
3964  
3965  
3966  
3967  
3968  
3969  
3970  
3971  
3972  
3973  
3974  
3975  
3976  
3977  
3978  
3979  
3980  
3981  
3982  
3983  
3984  
3985  
3986  
3987  
3988  
3989  
3990  
3991  
3992  
3993  
3994  
3995  
3996  
3997  
3998  
3999  
4000  
4001  
4002  
4003  
4004  
4005  
4006  
4007  
4008  
4009  
4010  
4011

```
*****
*TEST 14      DIV*DMO
*
*   FORK A SHOULD NOT FAIL.
*   SECTION 1
*   THE FIRST SECTION HAS A ZERO DIVISOR. IF STATE DIV.00
*   FAILS OR IRCF Z2(1) DOES NOT GET TO RACK E49 OR RACK
*   E49(B1) IS STUCK LOW EXECUTION WILL GO FROM DIV.10 TO
*   DIV.20. THIS WILL CAUSE THE DIVIDE TO ABORT (GO TO DVE.20)
*   AFTER STATE DIV.60 AND THE C BIT WILL BE CLEAR.
*
*   SECTION 2
*   THE NEXT SECTION DIVIDES 4 BY 2. IF RACK E49(B1) IS STUCK
*   HIGH THE ALGORITHM WILL ABORT THINKING THAT THE DIVISOR
*   IS ZERO. IF BEN04 FAILS THE DIVIDE WILL ABORT THINKING
*   THAT THE DIVIDEND IS THE MOST NEGATIVE NUMBER.
*   IF BEN16 FAILS THE DIVIDE WILL COMPLETE BUT R1 WILL CONTAIN
*   155776. IF BEN05 FAILS THE ALGORITHM WILL ABORT THRU STATE
*   DVE.20. IF BEN04 (AFTER DIV.70) FAILS R0 WILL END UP WITH 177777
*   AND R1 WILL HAVE 177774. IF BEN03 FAILS R0 WILL END UP WITH
*   20 AND R1 WILL HAVE 177774. IF BEN16 FAILS AFTER DVC.00 R0
*   WILL HAVE 2 BUT R1 WILL HAVE 177776.
*
*   SECTION 3
*   THE NEXT SECTION DIVIDES 6 BY 2 TO TEST BEN16*DR0(1).
*   A FAILURE WILL LEAVE THE REMAINDER (R1)=2 INSTEAD OF ZERO.
*
*   SECTION 4
*   THE NEXT SECTION DIVIDES 4 BY -2 TO TEST BEN15*SR15(1).
*   IF THIS FAILS R0 WILL CONTAIN 27777 AND R1 WILL CONTAIN 4.
*
*   SECTION 5
*   THE NEXT SECTION DIVIDES 1177777 BY 1 TO TEST BEN05*DIV QUIT.
*   IF THIS FAILS R0 WILL CONTAIN 177777.
*
*   SECTION 6
*   THE NEXT SECTION DIVIDES 1000000 B2 -2 TO TEST BEN05*DIV QUIT.
*   THIS SECTION WILL ONLY FAIL IF GRAJ E5 (Z2(0)*LEFT SAVE (1)) IS BAD.
*
*   SECTION 7
*   THE NEXT SECTION DIVIDES 100000000000 BY 2 TO TEST
*   BEN04*NEGATIVE DIVIDEND.
*
*   SECTION 8
*   THE NEXT SECTION DIVIDED 177776 177777 BY -1 TO TEST BEN05*DIV QUIT.
*   THIS TEST WILL ONLY FAIL IF GRAJ E5(N(1)*SR15(1)) IS BAD.
*
*   SECTION 9
*   THE NEXT SECTION DIVIDES -5 BY 2 TO ENSURE THAT THE REMAINDER IS
*   STORED AS A NEGATIVE NUMBER.
*
*   SECTION 10
*   THE NEXT SECTION DIVIDES -5 BY -2 TO TEST STATES DVC.20,DVC.40, & DVC.60
```

```
4012 :*  
4013 :* SECTION 11  
4014 :* THE NEXT SECTION DIVIDES -2**16 BY 2**14 TO TEST STATE DVN.20  
4015 :*  
4016 :* SECTION 12  
4017 :* THE NEXT SECTION DIVIDES 100 000200 BY -177 TO TEST STATES  
4018 :* DVD.00 AND DVD.10.  
4019 :*  
4020 011120 000004 TST14: SCOPE  
4021 011122 012767 012600 170140 MOV #TST15,NEXTTST ;SAVE ADDRESS OF NEXT TEST  
4022 :DIVISOR=0 SECTION (ALSO TESTS CONDITION CODE ROM FOR DIV)  
4023 011130 005000 CLR R0 ;PUT DIVIDEND IN  
4024 011132 012701 000004 MOV #4,R1 ;R0 AND R1  
4025 011136 005002 CLR R2 ;MAKE DIVISOR=0  
4026 011140 000270 SEN ;ENSURE N SET  
4027 :AND OSCILLOSCOPE SYNC POINT  
4028 011142 071002 DIV R2,R0 ;EXECUTE INSTRUCTION UNDER TEST  
4029 011144 013767 177776 170034 MOV @#PSW,$ERPSW ;SAVE PSW  
4030 011152 042767 177760 170026 BIC #177760,$ERPSW ;MASK OFF THE CC'S  
4031 011160 022767 000007 170020 CMP #7,$ERPSW ;DID THE CC'S COME OUT OK?  
4032 011166 001412 BEQ 2$ ;BRANCH IF YES  
4033 011170 022767 000002 170010 CMP #2,$ERPSW ;DID INSTR GO THROUGH DVE.20?  
4034 011176 001002 BNE 3$ ;BRANCH IF NO  
4035 011200 104075 ERROR 75 ;BEN02 FAILED  
4036 011202 000404 BR 2$  
4037 011204 012767 000007 167750 3$: MOV #7,$TMP0 ;SAVE EXPECTED VALUE  
4038 011212 104076 ERROR 76 ;CC'S BAD IN STATE DVE.00  
4039 :*****  
4040 :SECTION TWO (ALSO CHECKS CC LOAD OF STATE DVC.70)  
4041 011214 012767 011222 167666 2$: MOV #64,$LPERR ;SETUP ERROR LOOP  
4042 011222 012702 000002 64$: MOV #2,R2 ;PUT DIVISOR IN R2 (R0 & R1 HOLD DIVIDEND)  
4043 011226 000277 SCC ;ENSURE CC'S SET  
4044 :AND OSCILLOSCOPE SYNC POINT  
4045 011230 071002 DIV R2,R0 ;EXECUTE INSTRUCTION UNDER TEST  
4046 011232 013767 177776 167746 MOV @#PSW,$ERPSW ;SAVE PSW  
4047 011240 020027 000002 CMP R0,#2 ;DID QUOTIENT COME OUT CORRECT?  
4048 011244 001042 BNE 4$ ;BRANCH IF NO  
4049 011246 005701 TST R1 ;DID REMAINDER COME OUT CORRECT?  
4050 011250 001013 BNE 5$ ;BRANCH IF NO  
4051 011252 042767 177760 167726 BIC #177760,$ERPSW ;MASK OFF CC'S  
4052 011260 022767 000000 167720 CMP #0,$ERPSW ;ARE CC'S CORRECT?  
4053 011266 001513 BEQ 7$ ;BRANCH IF YES  
4054 011270 005067 167666 6$: CLR $TMP0 ;SAVE EXPECTED VALUE  
4055 011274 104077 ERROR 77 ;BAD CC'S DUE TO STATE DVC.70  
4056 011276 000507 BR 7$  
4057 011300 012767 000002 167654 5$: MOV #2,$TMP0 ;SAVE EXPECTED  
4058 011306 005067 167652 CLR $TMP1 ;VALUES  
4059 011312 020127 177776 CMP R1,#177776 ;DID BEN16*DR0(O) FAIL?  
4060 011316 001002 BNE 8$ ;BRANCH IF NO  
4061 011320 104100 ERROR 100 ;RACK E50(B0) STUCK HIGH  
4062 011322 000475 BR 7$  
4063 011324 020127 000004 8$: CMP R1,#4 ;DID BEN04 FAIL?  
4064 011330 001002 BNE 20$ ;BRANCH IF NO  
4065 011332 104101 ERROR 101 ;BEN04 FAILED  
4066 011334 000470 BR 7$  
4067 011336 010067 167612 20$: MOV R0,$REGU ;SAVE R0 &
```

```

4068 011342 010167 167610      MOV     R1,$REG1      ;R1 FOR TYPEOUT
4069 011346 104102      ERROR   102          ;QUOTIENT OK BUT REMAINDER BAD
4070 011350 000462      BR      7$
4071 011352 012767 000002 167602 4$:  MOV     #2,$TMP0     ;SAVE EXPECTED
4072 011360 005067 167600      CLR     $TMP1        ;VALUES
4073 011364 022700 000020      CMP     #20,R0       ;DID BEN03 FAIL?
4074 011370 001002      BNE     9$           ;BRANCH IF NO
4075 011372 104103      ERROR   103          ;RACK E49(A1) STUCK LOW
4076 011374 000450      BR      7$
4077 011376 022700 077777      9$:  CMP     #77777,R0    ;DID BEN04 (-DIV SUB) FAIL?
4078 011402 001002      BNE     10$          ;BRANCH IF NO
4079 011404 104104      ERROR   104          ;BEN04 STUCK TO DIV SUB
4080 011406 000443      BR      7$
4081 011410 022700 177777      10$:  CMP     #-1,R0       ;DID R0 GET A -1?
4082 011414 001002      BNE     13$          ;BRANCH IF NO
4083 011416 104110      ERROR   110          ;BEN04 (-N) FAILED
4084 011420 000436      BR      7$
4085 011422 020027 000000      13$:  CMP     R0,#0        ;DID R0 STAY 0?
4086 011426 001406      BEQ     11$          ;BRANCH IF YES
4087 011430 010067 167520      MOV     R0,$REG0     ;SAVE R0 & R1
4088 011434 010167 167516      MOV     R1,$REG1     ;FOR TYPEOUT
4089 011440 104105      ERROR   105          ;INSTRUCTION FAILED
4090 011442 000425      BR      7$
4091 011444 020127 155776      11$:  CMP     R1,#155776   ;DID BEN16*SR15(0) FAIL?
4092 011450 001002      BNE     12$          ;BRANCH IF NO
4093 011452 104106      ERROR   106          ;BEN16 FAILED TO DIV.50
4094 011454 000420      BR      7$
4095 011456 020127 000004      12$:  CMP     R1,#4        ;DID R1 CHANGE?
4096 011462 001406      BEQ     14$          ;BRANCH IF NO
4097 011464 010067 167464      MOV     R0,$REG0     ;SAVE R0 & R1
4098 011470 010167 167462      MOV     R1,$REG1     ;FOR TYPEOUT
4099 011474 104107      ERROR   107          ;CAN'T DETERMINE FAILURE
4100 011476 000407      BR      7$
4101 011500 032767 000001 167500 14$:  BIT     #BIT0,$ERPSW ;DID BEN02 FAIL?
4102 011506 001402      BEQ     15$          ;BRANCH IF NO
4103 011510 104111      ERROR   111          ;BEN02 FAILED
4104 011512 000401      BR      7$
4105 011514 104112      15$:  ERROR   112          ;BEN05 FAILED
4106
4107
4108 011516 012767 011524 167364 7$:  MOV     #63$,$LPERR  ;SETUP ERROR LOOP
4109 011524 005000 63$:  CLR     R0           ;PUT DIVIDEND IN R0
4110 011526 012701 000006      MOV     #6,R1        ;PUT DIVIDEND IN R1 (DIVISOR=2 IN R2)
4111 011532 000240      NOP
4112 011534 071002      DIV     R2,R0        ;OSCILLOSCOPE SYNC POINT
4113 011536 020127 000002      CMP     R1,#2        ;EXECUTE INSTRUCTION UNDER TEST
4114 011542 001001      BNE     16$          ;DID BEN16*DRO(1) FAIL?
4115 011544 104113      ERROR   113          ;BRANCH IF NO
4116
4117
4118 011546 012767 011554 167334 16$:  MOV     #62$,$LPERR  ;SETUP ERROR LOOP
4119 011554 005000 62$:  CLR     R0           ;
4120 011556 012701 000004      MOV     #4,R1        ;PUT DIVIDEND IN R0 & R1
4121 011562 012702 177776      MOV     #-2,R2       ;PUT DIVISOR IN R2
4122 011566 012767 177776 167366  MOV     #-2,$TMP0     ;SAVE EXPECTED
4123 011574 005067 167364      CLR     $TMP1        ;VALUES
    
```

\*\*\*\*\*

SECTION THREE

\*\*\*\*\*

SECTION FOUR (ALSO CHECK CC LOAD OF STATE DVC.90)



4124	011600	000240				NOP		:OSCILLOSCOPE SYNC POINT
4125	011602	071002				DIV	R2,R0	:EXECUTE INSTRUCTION UNDER TEST
4126	011604	013767	177776	167374		MOV	2#PSW,\$ERPSW	:SAVE CC'S
4127	011612	020027	177776			CMP	R0,#177776	:DID DIVIDE WORK?
4128	011616	001413				BEQ	17\$	:BRANCH IF YES
4129	011620	020027	027777			CMP	R0,#27777	:DID BEN16*SR15(1) FAIL?
4130	011624	001002				BNE	18\$	:BRANCH IF NO
4131	011626	104114				ERROR	114	:RACK E64(B0) STUCK LOW
4132	011630	000432				BR	21\$	
4133	011632	010067	167316		18\$:	MOV	R0,\$REG0	:SAVE R0 & R1 FOR
4134	011636	010167	167314			MOV	R1,\$REG1	:TYPEOUT
4135	011642	104115				ERROR	115	:EITHER STATE DVC.20 OR DVC.40 OR DVC.80
4136								:OR DVC.90 FAILED
4137	011644	000424				BR	21\$	
4138	011646	020127	000000		17\$:	CMP	R1,#0	:DID REMAINDER COME OUT CORRECT?
4139	011652	001406				BEQ	19\$	:BRANCH IF YES
4140	011654	010067	167274			MOV	R0,\$REG0	:SAVE R0 & R1
4141	011660	010167	167272			MOV	R1,\$REG1	:FOR TYPEOUT
4142	011664	104116				ERROR	116	:QUOTIENT OK, REMAINDER BAD
4143	011666	000413				BR	21\$	
4144	011670	042767	177760	167310	19\$:	BIC	#177760,\$ERPSW	:MASK OF CC'S
4145	011676	022767	000010	167302		CMP	#10,\$ERPSW	:DID CC'S COME OUT CORRECT?
4146	011704	001404				BEQ	21\$	:CONTINUE IF YES
4147	011706	012767	000010	167246		MOV	#10,\$TMPO	:SAVE EXPECTED VALUE
4148	011714	104117				ERROR	117	:BAD CC'S DUE TO STATE DVC.90
4149								
4150								:*****
4151	011716	012767	011724	167164	21\$:	MOV	#61\$,\$LPERR	:SECTION FIVE (DIV QUIT CAUSED BY N(0)*SR15(0))
4152	011724	012700	000002		61\$:	MOV	#2,R0	:SETUP ERROR LOOP
4153	011730	005001				CLR	R1	:SETUP R0 AND R1
4154	011732	012702	000001			MOV	#1,R2	:TO CAUSE DIV QUIT ABORT
4155	011736	000277				SCC		:PUT DIVISOR IN R2
4156	011740	000242				CLV		:SETUP CC'S TO COMPLIMENT
4157								:OF EXPECTED VALUE
4158	011742	071002				DIV	R2,R0	:AND OSCILLOSCOPE SYNC POINT
4159	011744	013767	177776	167234		MOV	2#PSW,\$ERPSW	:EXECUTE INSTRUCTION UNDER TEST
4160	011752	020027	000002			CMP	R0,#2	:SAVE CC'S
4161	011756	001402				BEQ	22\$	:DID DIVIDE ABORT?
4162	011760	104120				ERROR	120	:BRANCH IF YES
4163	011762	000413				BR	23\$	:DIV QUIT DID NOT GO LOW
4164	011764	042767	177760	167214	22\$:	BIC	#177760,\$ERPSW	:MASK OFF CC'S
4165	011772	022767	000002	167206		CMP	#2,\$ERPSW	:DID CC'S COME OUT OK?
4166	012000	001404				BEQ	23\$	:BRANCH IF YES
4167	012002	012767	000002	167152		MOV	#2,\$TMPO	:STORE EXPECTED CC'S FOR TYPEOUT
4168	012010	104121				ERROR	121	:CC'S BAD DUE TO EITHER DIV.30 OR DVE.20
4169								:*****
4170								:SECTION SIX (DIV QUIT CAUSED BY SHFTR=0)
4171	012012	012767	012020	167070	23\$:	MOV	#60\$,\$LPERR	:SETUP ERROR LOOP
4172	012020	012700	000002		60\$:	MOV	#2,R0	:SETUP R0 & R1 TO
4173	012024	005001				CLR	R1	:CAUSE DIV QUIT TO ABORT
4174	012026	012702	177776			MOV	#-2,R2	:SETUP DIVISOR (-2)
4175	012032	000240				NOP		:OSCILLOSCOPE SYNC POINT
4176	012034	071002				DIV	R2,R0	:EXECUTE INSTRUCTION UNDER TEST
4177	012036	022700	000002			CMP	#2,R0	:DID DIVIDE ABORT?
4178	012042	001401				BEQ	24\$	:BRANCH IF YES
4179	012044	104122				ERROR	122	:GRAJ E5 IS BAD (Z2(0)*LEFT SAVE (1))

```
4180  
4181  
4182 012046 012767 012054 167034  
4183 012054 012700 100000  
4184 012060 005001  
4185 012062 012702 000001  
4186 012066 000257  
4187 012070 000264  
4188 012072 071002  
4189 012074 013767 177776 167104  
4190 012102 020027 100000  
4191 012106 001402  
4192 012110 104123  
4193  
4194 012112 000413  
4195 012114 042767 177760 167064  
4196 012122 022767 000016 167056  
4197 012130 001404  
4198 012132 012767 000016 167022  
4199 012140 104124  
4200  
4201  
4202 012142 012767 012150 166740  
4203 012150 012700 177776  
4204 012154 012701 177777  
4205 012160 012702 177777  
4206 012164 000240  
4207 012166 071002  
4208 012170 020027 000001  
4209 012174 001401  
4210 012176 000403  
4211 012200 020127 000001  
4212 012204 001413  
4213 012206 012767 000001 166746  
4214 012214 010067 166734  
4215 012220 012767 000001 166736  
4216 012226 010167 166724  
4217 012232 104125  
4218  
4219  
4220 012234 012767 012242 166646  
4221 012242 012700 177777  
4222 012246 012701 177773  
4223 012252 012702 000002  
4224 012256 012767 177776 166676  
4225 012264 012767 177777 166672  
4226 012272 000240  
4227 012274 071002  
4228 012276 010067 166652  
4229 012302 010167 166650  
4230 012306 020127 177777  
4231 012312 001402  
4232 012314 104126  
4233 012316 000404  
4234 012320 022700 177776  
4235 012324 001401  
:*****  
:SECTION SEVEN  
24$: MOV #57$, $LPERR ;SETUP ERROR LOOP  
57$: MOV #BIT15, R0 ;MAKE DIVIDEND  
CLR R1 ;MOST NEGATIVE NUMBER  
MOV #1, R2 ;SETUP DIVISOR  
CCC ;SET CC'S TO COMPLIMENT OF EXPECTED  
SEZ ;OSCILLOSCOPE SYNC POINT  
DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST  
MOV @PSW, $ERPSW ;SAVE PSW  
CMP R0, #BIT15 ;DID DIVIDE ABORT?  
BEQ 25$ ;BRANCH IF YES  
ERROR 123 ;BEN04*N FAILED  
BR 26$  
25$: BIC #177760, $ERPSW ;MASK OFF CC'S  
CMP #16, $ERPSW ;DID CC'S LOAD PROPERLY?  
BEQ 26$ ;BRANCH IF YES  
MOV #16, $TMP0 ;STORE EXPECTED VALUE  
ERROR 124 ;CC'S DID NOT LOAD PROPERLY  
:*****  
:SECTION EIGHT (DIV QUIT CAUSED BY N(1)*SR15(1))  
26$: MOV #56$, $LPERR ;SETUP ERROR LOOP  
56$: MOV #177776, R0 ;SETUP R0 & R1 TO  
MOV #177777, R1 ;CAUSE DIV QUIT TO ABORT  
MOV #177777, R2 ;SETUP DIVISOR (-1)  
NOP ;OSCILLOSCOPE SYNC POINT  
DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST  
CMP R0, #1 ;DID DIVIDE ABORT LEAVE R0-1  
BEQ 27$ ;BRANCH IF YES  
BR 35$  
27$: CMP R1, #1 ;DID DIVIDE ABORT?  
BEQ 28$ ;BRANCH IF YES  
35$: MOV #1, $TMP0 ;STORE EXPECTED VALUE  
MOV R0, $REG0 ;SAVE R0  
MOV #1, $TMP1  
MOV R1, $REG1 ;SAVE R1  
ERROR 125 ;EITHER GRAJ E5 BAD OR MICROSTATE BAD  
:*****  
:SECTION NINE  
28$: MOV #55$, $LPERR ;SETUP ERROR LOOP  
55$: MOV #-1, R0 ;PUT DIVIDEND IN  
MOV #-5, R1 ;R0 & R1  
MOV #2, R2 ;PUT DIVISOR IN R2  
MOV #-2, $TMP0 ;SAVE EXPECTED VALUE  
MOV #-1, $TMP1 ;SAVE EXPECTED VALUE  
NOP ;OSCILLOSCOPE SYNC POINT  
DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST  
MOV R0, $REG0 ;SAVE R0  
MOV R1, $REG1 ;SAVE R1  
CMP R1, #-1 ;IS REMAINDER CORRECT?  
BEQ 29$ ;BRANCH IF YES  
ERROR 126 ;REMAINDER BAD  
BR 30$  
29$: CMP #-2, R0 ;IS QUOTIENT CORRECT?  
BEQ 30$ ;BRANCH IF YES
```

```
4236 012326 104127 ERROR 127 ;QUOTIENT IS INCORRECT
4237 *****
4238 :SECTION TEN
4239 012330 012767 012336 166552 30$: MOV #54$, $LPERR ;SETUP ERROR LOOP
4240 012336 012700 177777 54$: MOV #-1, R0 ;SETUP
4241 012342 012701 177773 MOV #-5, R1 ;DIVIDEND
4242 012346 012702 177776 MOV #-2, R2 ;AND DIVISOR
4243 012352 012767 000002 166602 MOV #2, $TMP0 ;SAVE EXPECTED VALUES
4244 012360 012767 177777 166576 MOV #-1, $TMP1 ;OF R0 & R1
4245 012366 000240 NOP ;OSCILLOSCOPE SYNC POINT
4246 012370 071002 DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST
4247 012372 010067 166556 MOV R0, $REG0 ;SAVE R0
4248 012376 010167 166554 MOV R1, $REG1 ;SAVE R1
4249 012402 020027 000002 CMP R0, #2 ;IS QUOTIENT CORRECT?
4250 012406 001402 BEQ 31$ ;BRANCH IF YES
4251 012410 104130 ERROR 130 ;QUOTIENT BAD
4252 012412 000404 BR 32$
4253 012414 020127 177777 31$: CMP R1, #-1 ;IS REMAINDER CORRECT
4254 012420 001401 BEQ 32$ ;BRANCH IF YES
4255 012422 104131 ERROR 131 ;REMAINDER INCORRECT (QUOT. OK)
4256 *****
4257 :SECTION ELEVEN
4258 012424 012767 012432 166456 32$: MOV #53$, $LPERR ;SETUP ERROR LOOP
4259 012432 012700 177776 53$: MOV #177776, R0 ;SETUP DIVIDEND
4260 012436 005001 CLR R1 ;AND DIVISOR TO GET A
4261 012440 012702 040000 MOV #40000, R2 ;QUOT OF -10 & REMAINDER=0
4262 012444 012767 177770 166510 MOV #-10, $TMP0 ;SAVE EXPECTED VALUE
4263 012452 005067 166506 CLR $TMP1 ;SAVE EXPECTED VALUE
4264 012456 000240 NOP ;OSCILLOSCOPE SYNC POINT
4265 012460 071002 DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST
4266 012462 010067 166466 MOV R0, $REG0 ;SAVE R0
4267 012466 010167 166464 MOV R1, $REG1 ;SAVE R1
4268 012472 020027 177770 CMP R0, #-10 ;IS QUOTIENT CORRECT?
4269 012476 001402 BEQ 33$ ;BRANCH IF YES
4270 012500 104132 ERROR 132 ;QUOTIENT IS BAD
4271 012502 000404 BR 34$
4272 012504 020127 000000 33$: CMP R1, #0 ;IS REMAINDER CORRECT?
4273 012510 001401 BEQ 34$ ;BRANCH IF YES
4274 012512 104133 ERROR 133 ;REMAINDER BAD (QUOT. OK)
4275 *****
4276 :SECTION TWELVE
4277 012514 012767 012522 166366 34$: MOV #52$, $LPERR ;SETUP ERROR LOOP
4278 012522 012700 000100 52$: MOV #100, R0 ;SETUP DIVIDEND
4279 012526 012701 000200 MOV #200, R1 ;AND DIVISOR TO GENERATE
4280 012532 012702 177601 MOV #-177, R2 ;DIVISION OVERFLOW
4281 012536 000277 SCC ;
4282 012540 000242 CLV ;AND OSCILLOSCOPE SYNC POINT
4283 012542 071002 DIV R2, R0 ;EXECUTE INSTRUCTION UNDER TEST
4284 012544 013767 177776 166434 MOV @#PSW, $ERPSW ;SAVE PSW
4285 012552 042767 177760 166426 BIC #177760, $ERPSW ;MASK OFF CC'S
4286 012560 022767 000002 166420 CMP #2, $ERPSW ;ARE CC'S CORRECT?
4287 012566 001404 BEQ TST15 ;TEST OK, GO TO NEXT TEST
4288 012570 012767 000002 166364 MOV #2, $TMP0 ;SAVE EXPECTED
4289 012576 104134 ERROR 134 ;BAD CC'S ON DIVISION OVERFLOW
4290 *****
4291 ;*TEST 15 MTP*DMO
```

4292  
4293  
4294  
4295  
4296  
4297  
4298  
4299  
4300  
4301  
4302  
4303  
4304  
4305 012600 000004  
4306 012602 012767 012746 166364  
4307 012610 012767 012746 166452  
4308 012616 012737 012736 000010  
4309 012624 005037 177776  
4310 012630 012706 001074  
4311 012634 012716 100000  
4312 012640 005000  
4313 012642 000277  
4314 012644 000250  
4315  
4316 012646 006600  
4317 012650 013767 177776 166330  
4318 012656 010067 166272  
4319 012662 012767 100000 166272  
4320 012670 100403  
4321 012672 012700 100000  
4322 012676 104135  
4323 012700 022706 001076 2\$:  
4324 012704 001401  
4325 012706 104136  
4326 012710 042767 177760 166270 3\$:  
4327 012716 022767 000011 166262  
4328 012724 001410  
4329 012726 012767 000011 166226  
4330 012734 104137  
4331 012736 012737 000012 000010 1\$:  
4332 012744 104140

IF FORK A FAILS EXECUTION WILL GO TO RSD.00.  
THE ONLY OTHER POSSIBLE FAILURES WOULD BE IN ROM STATES  
MTP.00 OR MTP.10.  
NOTE: THIS TEST ONLY TESTS THE CPU FUNCTIONS OF THIS INSTRUCTION.  
THE MEMORY MANAGEMENT TEST VERIFIES THE INTERMODE TRANSFER.  
AS FAR AS THE CPU IS CONCERNED THERE IS NO DIFFERENCE  
BETWEEN MTP1 AND MTPD.

ROM FLOW-45,151,146,205  
\*\*\*\*\*  
TST15: SCOPE  
MOV #TST16,\$ESCAPE ;SAVE START ADDRESS OF NEXT TEST  
MOV #TST16,NEXTTST ;SAVE START ADDRESS OF NEXT TEST  
MOV #1\$,@#RESVEC ;SETUP RESVEC  
CLR @#PSW ;ENSURE PREVIOUS MODE KERNAL  
MOV #1074,SP ;SETUP THE SP  
MOV #BIT15,(SP) ;SET SIGN BIT ON STACK  
CLR R0 ;ENSURE R0 CLEAR  
SCC ;SET CC'S TO COMPLIMENT  
CLN ;OF EXPECTED VALUE (EXCEPT FOR C)  
;AND OSCILLOSCOPE SYNC POINT  
MTP1 R0 ;EXECUTE INSTRUCTION UNDER TEST  
MOV @#PSW,\$ERPSW ;SAVE PSW  
MOV R0,\$REGO ;SAVE R0 & SEC CC'S  
MOV #BIT15,\$TMPO ;SAVE EXPECTED VALUE  
BMI 2\$ ;BRANCH IF R0 LOADED  
MOV #BIT15,R0 ;SAVE EXPECTED VALUE  
ERROR 135 ;R0 DID NOT LOAD  
2\$: CMP #1076,SP ;DID SP INCREMENT?  
BEQ 3\$ ;BRANCH IF YES  
ERROR 136 ;SP DID NOT INCREMENT  
3\$: BIC #177760,\$ERPSW ;MASK OFF CC'S  
CMP #11,\$ERPSW ;DID CC'S COME OUT CORRECT?  
BEQ TST16 ;BRANCH IF YES  
MOV #11,\$TMPO ;SAVE EXPECTED VALUE  
ERROR 137 ;CC'S BAD (CC ROM)  
1\$: MOV #12,@#RESVEC ;RESTORE RESVEC  
ERROR 140 ;FORK A FAILED

\*\*\*\*\*  
\*TEST 16 MTP\*DM1  
IF FORK A FAILS EXECUTION WILL GO TO RSD.00.  
THIS WILL ONLY HAPPEN IF RACF E20(4) IS STUCK HIGH.  
THIS TEST ENSURES STATE MTP.10 RELOADS THE  
DR IF THE DESTINATION IS R6 AND THAT IT PUTS THE PC IN THE  
DR IF THE DESTINATION FIELD IS R7.

4343  
4344  
4345 012746 000004  
4346 012750 012767 013120 166312  
4347 012756 012737 013110 000010

ROM FLOW-45,151,146,111,155,312  
\*\*\*\*\*  
TST16: SCOPE  
MOV #TST17,NEXTTST ;SAVE ADDRESS OF NEXT TEST  
MOV #1\$,@#RESVEC ;SETUP RESVEC

```
4348 012764 012706 001072      MOV      #1072,SP      ;SETUP THE SP
4349 012770 012716 100000      MOV      #BIT15,(SP)  ;SET THE SIGN BIT ON THE STACK
4350 012774 005066 000002      CLR      2(SP)        ;ENSURE 1074 IS CLEAR
4351 013000 000240                NOP                    ;OSCILLOSCOPE SYNC POINT
4352 013002 006616                MTP1    (SP)          ;EXECUTE INSTRUCTION UNDER TEST
4353 013004 005737 001074      TST      @#1074       ;DID 1074 GET SIGN BIT SET?
4354 013010 100413                BMI     2$            ;BRANCH IF YES
4355 013012 022706 001074      CMP      #1074,SP     ;DID MTP.10 FAIL TO RELOAD THE DR?
4356 013016 001002                BNE     3$            ;BRANCH IF NO
4357 013020 104141                ERROR   141          ;STATE MTP.10 FAILED
4358 013022 000406                BR      2$
4359 013024 010667 166124      3$:  MOV     SP,$REG0    ;SAVE SP
4360 013030 012767 001074 166120      MOV     #1074,$REG1   ;SAVE EXPECTED VALUE
4361 013036 104142                ERROR   142          ;DON'T KNOW WHAT HAPPENED
4362 013040 012767 013046 166042      2$:  MOV     #64$,$LPERR ;SETUP ERROR LOOP
4363 013046 012706 001074      64$:  MOV     #1074,SP   ;SETUP THE SP
4364 013052 000240                NOP                    ;OSCILLOSCOPE SYNC POINT
4365 013054 006617                MTP1    (PC)          ;EXECUTE INSTRUCTION UNDER TEST
4366 013056 000000                4$:  .WORD   0
4367 013060 005767 177772      TST      4$           ;DID 4$ GET BIT15 SET?
4368 013064 100403                BMI     5$            ;BRANCH IF YES
4369 013066 005067 177764      CLR      4$           ;RESTORE 4$
4370 013072 104143                ERROR   143          ;STATE MTP.10 DID NOT PUT PCB IN DR
4371 013074 005067 177756      5$:  CLR      4$           ;RESTORE 4$
4372 013100 012737 000012 000010      MOV     #12,@#RESVEC  ;RESTORE RESVEC
4373 013106 000404                BR      TST17         ;GO TO NEXT TEST
4374 013110 012737 000012 000010      1$:  MOV     #12,@#RESVEC  ;RESTORE RESVEC
4375 013116 104144                ERROR   144          ;FORK A FAILED
4376                                     :*****
4377                                     :*TEST 17      MFP*DMO
4378                                     :*
4379                                     :* IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
4380                                     :* IF ANYTHING ELSE FAILS, THEN A ROM STATE IS BAD.
4381                                     :*
4382                                     :* ROM FLOW-46,304,250,222,300
4383                                     :*****
4384 013120 000004      TST17: SCOPE
4385 013122 012767 013306 166044      MOV     #TST20,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
4386 013130 012767 013306 166132      MOV     #TST20,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
4387 013136 012767 013172 165742      MOV     #4$,$LPADR     ;SET UP LOOP
4388 013144 012767 013172 165736      MOV     #4$,$LPERR     ;SETUP ERROR LOOP
4389 013152 013767 177776 166010      MOV     @#PSW,$TMP3    ;SAVE PSW
4390 013160 012746 000340      MOV     #PR7,-(SP)     ;PUT NEW PSW ON STACK
4391 013164 012746 013172      MOV     #4$,-(SP)     ;PUT RETURN ADDR ON STACK
4392 013170 000006                RTT                    ;TURN T BIT OFF
4393 013172 012737 013276 000010 4$:  MOV     #1$,@#RESVEC   ;SETUP RESVEC
4394 013200 012706 001076      MOV     #1076,SP      ;SETUP THE SP
4395 013204 005016                CLR     (SP)          ;ENSURE 1076 CLEAR
4396 013206 005066 177776      CLR     -2(SP)        ;ENSURE 1074 CLEAR
4397 013212 012700 100000      MOV     #BIT15,RO     ;SET THE SIGN BIT IN RO
4398 013216 000277                SCC                    ;SETUP CC'S TO COMPLIMENT
4399 013220 000250                CLN                    ;OF EXPECTED VALUE (EXCEPT FOR C)
4400                                     ;AND OSCILLOSCOPE SYNC POINT
4401 013222 006500                MFPI   RO             ;EXECUTE INSTRUCTION UNDER TEST
4402 013224 013767 177776 165754      MOV     @#PSW,$ERPSW  ;SAVE THE PSW
4403 013232 022706 001074      CMP     #1074,SP      ;DID THE SP DECREMENT?
```

```
4404 013236 001401 BEQ 2$ :BRANCH IF YES
4405 013240 104145 ERROR 145 :STATE MFP.10 DID NOT DEC. THE SP
4406 013242 005716 2$: TST (SP) :DID RO GET PUT ON THE STACK?
4407 013244 100401 BMI 3$ :BRANCH IF YES
4408 013246 104146 ERROR 146 :RO DID NOT GET PUT ON THE STACK
4409 013250 042767 177760 165730 3$: BIC #177760,$ERPSW :MASK OFF THE CC'S
4410 013256 022767 000011 165722 CMP #11,$ERPSW :DID THE CC'S SET CORRECTLY?
4411 013264 001410 BEQ TST20 :BRANCH IF YES
4412 013266 012767 000011 165666 MOV #11,$TMP0 :SAVE EXPECTED VALUE
4413 013274 104147 ERROR 147 :INCORRECT CC'S
4414 013276 012737 000012 000010 1$: MOV #12,@#RESVEC :RESTORE RESVEC
4415 013304 104150 ERROR 150 :FORK A FAILED
4416 :*****
4417 :*TEST 20 MFP*DM2
4418 :*
4419 :* IF FORK A FAILS EXECUTION WILL GO TO RSD.00.
4420 :* IF FORK B FAILS EXECUTION WILL ALSO GO TO RSD.00 BUT THE DR
4421 :* WILL HAVE BEEN INCREMENTED. IF ANYTHING ELSE FAILS THEN
4422 :* STATE MFP.00 IS BAD.
4423 :*
4424 :* ROM FLOW-2,175,66,250,222,300
4425 :*****
4426 013306 000004 TST20: SCOPE
4427 013310 012767 013446 165656 MOV #TST21,$ESCAPE :SAVE START ADDRESS OF NEXT TEST
4428 013316 012767 013446 165744 MOV #TST21,NEXTTST :SAVE START ADDRESS OF NEXT TEST
4429 013324 012737 013426 000010 MOV #1$,@#RESVEC :SETUP RESVEC
4430 013332 012706 001076 MOV #1076,SP :SETUP THE SP
4431 013336 005016 CLR (SP) :ENSURE WORDS ON
4432 013340 005066 177776 CLR -2(SP) :STACK CLEAR
4433 013344 012700 001164 MOV #$TMP1,RO :PUT ADDRESS OF $TMP1 IN RO
4434 013350 012710 100000 MOV #BIT15,(RO) :SET THE SIGN BIT IN $TMP1
4435 013354 000277 SCC :SETUP CC'S TO COMPLIMENT OF
4436 013356 000250 CLN :EXPECTED VALUE (EXCEPT FOR C)
4437 :AND OSCILLOSCOPE SYNC POINT
4438 013360 006520 MFPI (RO)+ :EXECUTE INSTRUCTION UNDER TEST
4439 013362 013767 177776 165616 MOV @#PSW,$ERPSW :SAVE PSW
4440 013370 022706 001074 CMP #1074,SP :DID THE SP DECREMENT?
4441 013374 001401 BEQ 2$ :BRANCH IF YES
4442 013376 104151 ERROR 151 :STATE MFP.00 IS BAD
4443 013400 042767 177760 165600 2$: BIC #177760,$ERPSW :MASK OFF THE CC'S
4444 013406 022767 000011 165572 CMP #11,$ERPSW :DID THE CC'S SET CORRECTLY?
4445 013414 001414 BEQ TST21 :BRANCH IF YES
4446 013416 012767 000011 165536 MOV #11,$TMP0 :SAVE EXPECTED VALUE
4447 013424 104152 ERROR 152 :INCORRECT CC'S DUE TO STATE MFP.00
4448 013426 012737 000012 000010 1$: MOV #12,@#RESVEC :RESTORE RESVEC
4449 013434 022700 001166 CMP #$TMP2,RO :DID RO INCREMENT?
4450 013440 001401 BEQ 3$ :BRANCH IF YES
4451 013442 104153 ERROR 153 :FORK A FAILED
4452 013444 104154 3$: ERROR 154 :FORK B FAILED
4453 :*****
4454 :*TEST 21 BPT
4455 :*
4456 :* FORK A SHOULD NOT FAIL.
4457 :* IF THE TRAP VECTOR LOGIC FAILS THE TRAP VECTOR WOULD
4458 :* COME OUT TO BE 4.
4459 :* THE ONLY OTHER FAILURE WOULD BE TRP.00.
```

```
4460      :*      IF THIS STATE FAILS TO LOAD THE DR THE TRAP VECTOR WILL
4461      :*      BE WHATEVER IS IN R3.  IF IT FAILS TO LOAD THE BR THE OLD
4462      :*      PS WILL FAIL TO BE STACKED.
4463      :*****
4464 013446 000004      TST21: SCOPE
4465 013450 012767 013562 165516      MOV      #TST22,$ESCAPE      :SAVE START ADDRESS OF NEXT TEST
4466 013456 012767 013562 165604      MOV      #TST22,NEXTTST     :SAVE START ADDRESS OF NEXT TEST
4467 013464 012737 013522 000014 4$:  MOV      #1$,@#BPTVEC       :SETUP BPT VECTOR
4468 013472 012706 001100      MOV      #STACK,SP         :SETUP THE SP
4469 013476 012737 013542 000004      MOV      #2$,@#ERRVEC      :SETUP LOCATION 4
4470 013504 012737 013544 000024      MOV      #3$,@#24          :SETUP LOCATION 24
4471 013512 012703 000024      MOV      #24,R3            :SETUP R3
4472 013516 000237      SPL      7                  :SETUP PSW
```

```

4473 013520 000003          BPT          :EXECUTE INSTRUCTION UNDER TEST
4474 013522 042737 177437 001076 1$: BIC          #177437,@#1076 :MASK OFF PRIORITIES OF OLD PS
4475 013530 022737 000340 001076    CMP          #340,@#1076 :DID OLD PS GET STACKED?
4476 013536 001403          BEQ          5$          :BRANCH IF YES
4477 013540 104160          ERROR        160        :STATE TRP.00 FAILED
4478 013542 104161          2$: ERROR        161        :TRAP VECTOR CAME UP BAD
4479 013544 104162          3$: ERROR        162        :STATE TRP.00 FAILED
4480 013546 012737 032666 000004 5$: MOV          #CPUSPUR,@#ERRVEC :RESTORE ERR VEC
4481 013554 012737 032652 000014    MOV          #SRTN,@#TBITVEC :RESTORE T BIT VECTOR
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497 013562 000004          TST22: SCOPE
4498 013564 012767 013766 165402    MOV          #TST23,$ESCAPE :SAVE START ADDRESS OF NEXT TEST
4499 013572 012767 013766 165470    MOV          #TST23,NEXTTST :SAVE START ADDRESS OF NEXT TEST
4500 013600 012706 001100          MOV          #STACK,SP      :INITIALIZE THE SP
4501 013604 012767 013642 165274    MOV          #4,$,$LPADR    :SETUP LOOP ADR
4502 013612 012767 013642 165270    MOV          #4,$,$LPERR    :SETUP ERROR LOOP
4503 013620 032767 000020 165342    BIT          #BIT4,$TMP3    :WAS T BIT ON?
4504 013626 001405          BEQ          4$          :BRANCH IF NO
4505 013630 012746 000360          MOV          #360,-(SP)     :PUT NEW PSW ON STACK
4506 013634 0127'6 013642          MOV          #4,$,-(SP)    :PUT RETURN ADDR ON STACK
4507 013640 000006          RTT          :TURN T BIT ON
4508 013642 000237          4$: SPL          7          :SET THE CPU PRIORITY AT 7.
4509 013644 005067 165312          CLR          $TMP0         :SETUP COMPARISON LOCATION
4510 013650 005037 177772          CLR          @#PIRQ        :CLEAR PIRQ REGISTER
4511 013654 026737 165302 177772    CMP          $TMP0,@#PIRQ   :DID PIRQ CLEAR?
4512 013662 001035          BNE          1$          :BRANCH IF NO
4513 013664 012700 000177          MOV          #177,R0        :SETUP ITERATION COUNT
4514 013670 012767 001042 165264    MOV          #1042,$TMP0    :SETUP COMPARISON LOCATION
4515 013676 012701 000002          MOV          #2,R1         :SETUP R1
4516 013702 062737 001000 177772 2$: ADD          #1000,@#PIRQ   :START COUNT PATTERN
4517 013710 026737 165246 177772    CMP          $TMP0,@#PIRQ   :DID REGISTER SET CORRECT?
4518 013716 001017          BNE          1$          :BRANCH IF NO
4519 013720 120137 177773          CMPB        R1,@#PIRQ+1    :IS PIRQ READY TO GO TO NEXT LEVEL?
4520 013724 001005          BNE          3$          :BRANCH IF NO
4521 013726 062767 000042 165226    ADD          #42,$TMP0     :INCREMENT ENCODED VALUES IN TEST LOC.
4522 013734 005201          INC          R1           :SETUP R1 FOR ROTATE
4523 013736 006101          ROL          R1           :SET R1 TO NEXT CHECK LEVEL
4524 013740 062767 001000 165214 3$: ADD          #1000,$TMP0    :INC. PIRQ LEVEL IN TEST LOCATION
4525 013746 077023          SOB          R0,2$        :CONTINUE COUNT
4526 013750 005037 177772          CLR          @#PIRQ        :ENSURE PIRQ CLEAR
4527 013754 000404          BR          TST23         :GO TO NEXT TEST
4528 013756 013767 177772 165232 1$: MOV          @#PIRQ,$EPIRQ :SAVE PIRQ FOR TYPEOUT
    
```

\*\*\*\*\*  
 \*ALL THE LOGIC FOR (JMP+JSR)\*DMO HAS BEEN TESTED.  
 \*\*\*\*\*

\*\*\*\*\*  
 \*TEST 22 BIT TEST OF PIRQ REGISTER  
 \*  
 \* IF ONE OF THE BLOCK LEVELS IS STUCK HIGH OR TMCB  
 \* PS07(0)'S STUCK HIGH OR PDRD PRIORITY=0 IS STUCK HIGH,  
 \* A PIRQ TRAP LOOP WILL OCCUR WHEN THAT PIR LEVEL IS ENABLED.  
 \*  
 \* A COUNT PATTERN IS THEN RUN THRU THE REGISTER TO ENSURE THAT  
 \* THE ENCODER FUNCTIONS PROPERLY.  
 \*\*\*\*\*

TST22: SCOPE  
 MOV #TST23,\$ESCAPE :SAVE START ADDRESS OF NEXT TEST  
 MOV #TST23,NEXTTST :SAVE START ADDRESS OF NEXT TEST  
 MOV #STACK,SP :INITIALIZE THE SP  
 MOV #4,\$,\$LPADR :SETUP LOOP ADR  
 MOV #4,\$,\$LPERR :SETUP ERROR LOOP  
 BIT #BIT4,\$TMP3 :WAS T BIT ON?  
 BEQ 4\$ :BRANCH IF NO  
 MOV #360,-(SP) :PUT NEW PSW ON STACK  
 MOV #4,\$,-(SP) :PUT RETURN ADDR ON STACK  
 RTT :TURN T BIT ON  
 4\$: SPL 7 :SET THE CPU PRIORITY AT 7.  
 CLR \$TMP0 :SETUP COMPARISON LOCATION  
 CLR @#PIRQ :CLEAR PIRQ REGISTER  
 CMP \$TMP0,@#PIRQ :DID PIRQ CLEAR?  
 BNE 1\$ :BRANCH IF NO  
 MOV #177,R0 :SETUP ITERATION COUNT  
 MOV #1042,\$TMP0 :SETUP COMPARISON LOCATION  
 MOV #2,R1 :SETUP R1  
 2\$: ADD #1000,@#PIRQ :START COUNT PATTERN  
 CMP \$TMP0,@#PIRQ :DID REGISTER SET CORRECT?  
 BNE 1\$ :BRANCH IF NO  
 CMPB R1,@#PIRQ+1 :IS PIRQ READY TO GO TO NEXT LEVEL?  
 BNE 3\$ :BRANCH IF NO  
 ADD #42,\$TMP0 :INCREMENT ENCODED VALUES IN TEST LOC.  
 INC R1 :SETUP R1 FOR ROTATE  
 ROL R1 :SET R1 TO NEXT CHECK LEVEL  
 3\$: ADD #1000,\$TMP0 :INC. PIRQ LEVEL IN TEST LOCATION  
 SOB R0,2\$ :CONTINUE COUNT  
 CLR @#PIRQ :ENSURE PIRQ CLEAR  
 BR TST23 :GO TO NEXT TEST  
 1\$: MOV @#PIRQ,\$EPIRQ :SAVE PIRQ FOR TYPEOUT



4529 013764 104163  
4530  
4531  
4532  
4533  
4534  
4535  
4536  
4537  
4538  
4539  
4540  
4541  
4542  
4543 013766 000004  
4544 013770 012767 014336 165176  
4545 013776 012767 014336 165264  
4546 014004 012737 014274 000024  
4547 014012 012737 014326 000000  
4548 014020 012737 000340 000002  
4549 014026 012737 014302 000004  
4550 014034 012737 014252 000240  
4551 014042 032737 000020 177776  
4552 014050 001404  
4553 014052 012737 000360 000242  
4554 014060 000403  
4555 014062 012737 000340 000242  
4556 014070 012706 001100  
4557 014074 000230  
4558 014076 052737 001000 177772  
4559 014104 005037 177772  
4560  
4561 014110 012737 014132 000240  
4562 014116 012737 001000 177772  
4563 014124 005037 177772  
4564 014130 000403  
4565 014132 005037 177772  
4566 014136 104164  
4567  
4568 014140 012737 014162 000240  
4569 014146 052737 010000 177772  
4570 014154 005037 177772  
4571 014160 000403  
4572 014162 005037 177772  
4573 014166 104165  
4574  
4575  
4576 014170 012737 014212 000240  
4577 014176 052737 040000 177772  
4578 014204 005037 177772  
4579 014210 000403  
4580 014212 005037 177772  
4581 014216 104166  
4582  
4583  
4584

```
ERROR 163 ;PIRQ REG. FAILED
:*****
*TEST 23 PIR LEVEL 1 INTERRUPT
*
* IF BEN13 FAILS EXECUTION WOULD GO TO ONE OF THE FOLLOWING:
* PUP.00, BRK.20, OR SER.00.
* PUP.00 WOULD START THE POWER UP ROUTINE.
* BRK.20 WOULD CAUSE A TRAP TO ZERO.
* SER.00 WOULD PUT 4 IN THE SP AND PERFORM A RED ZONE TRAP.
* IF TMCB PIRQ DOES NOT GET TO DAPE OR IF DAPE TV05*07 DOES NOT
* GO HIGH OR DOES NOT GET THRU TO THE ALU A TRAP TO 4 WILL OCCUR.
* IF THE INTERRUPT DOESN'T OCCUR AN ATTEMPT IS MADE TO ISOLATE THE
* FAILURE.
:*****
TST23: SCOPE
MOV #TST24,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
MOV #TST24,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
MOV #10$,@#PWRVEC ;SETUP
MOV #12$,@#0 ;SETUP LOCATION 0
MOV #PR7,@#2 ;PUT PRIORITY 7 IN 2
MOV #11$,@#4 ;FAILURE TRAP VECTORS
MOV #1$,@#PIRQVEC ;SETUP PIRQ VECTOR
BIT #BIT4,@#PSW ;IS T BIT ON?
BEQ 13$ ;BRANCH IF NO
MOV #360,@#PIRQVEC+2 ;SET T BIT IN PIRQ VECTOR
BR 14$
13$: MOV #PR7,@#PIRQVEC+2 ;SETUP PIRQ VECTOR PSW
14$: MOV #STACK,SP ;SETUP THE SP
SPL 0 ;SET PRIORITY LEVEL AT ZERO
BIS #BIT9,@#PIRQ ;SET LEVEL ONE
CLR @#PIRQ ;CLEAR LEVEL ONE
;TRY TO GET INTERRUPT AT FET.01 INSTEAD OF TST.10.
MOV #2$,@#PIRQVEC ;SETUP PIRQ VEC
MOV #BIT9,@#PIRQ ;SET LEVEL ONE
CLR @#PIRQ
BR 3$ ;BRANCH IF NO INTERRUPT
2$: CLR @#PIRQ ;CLEAR LEVEL 1
ERROR 164 ;STATE TST.10 HAS BAD BEN FIELD
;TRY PIR LEVEL 4
3$: MOV #4$,@#PIRQVEC ;SETUP PIRQ VECTOR
BIS #BIT12,@#PIRQ ;SET LEVEL 4
CLR @#PIRQ
BR 5$ ;BRANCH IF NO INTERRUPT
4$: CLR @#PIRQ ;CLEAR LEVEL 4
ERROR 165 ;EITHER TMCB E62(1) IS BAD OR SOMETHING
;IN THE HONOR PIR 1 LOGIC IS BAD.
;TRY PIR 6
5$: MOV #6$,@#PIRQVEC ;SETUP PIRQ VECTOR
BIS #BIT14,@#PIRQ ;SET LEVEL 6
CLR @#PIRQ
BR 7$ ;BRAHNCN IF NO INTERRUPT
6$: CLR @#PIRQ
ERROR 166 ;EITHER TMCB E51(9) OR E55(10-11) OR
;E62 IS BAD OR TMCA INH BELOW BR0 IS
;STUCK LOW
;TRY PIR 7
```

```

4585 014220 012737 014242 000240 7$: MOV #8$,@PIRQVEC ;SETUP PIRQ VECTOR
4586 014226 052737 100000 177772 BIS #BIT15,@PIRQ ;SET LEVEL 7
4587 014234 005037 177772 CLR @PIRQ ;CLEAR LEVEL 7
4588 014240 000403 BR 9$ ;BRANCH IF NO INTERRUPT
4589 014242 005037 177772 8$: CLR @PIRQ
4590 014246 104167 ERROR 167 ;TMCA ABOVE BR7 MIGHT BE STUCK LOW
4591 014250 104170 9$: ERROR 170 ;TMCE BRQ CLOCK MIGHT BE STUCK LOW
4592 014252 005037 177772 1$: CLR @PIRQ ;CLEAR LEVEL 1
4593 014256 012737 032666 000004 MOV #CPUSPUR,@ERRVEC ;RESTORE ERR VEC
4594 014264 012737 036300 000024 MOV #SPWRDN,@PWRVEC ;RESTORE THE POWER VECTOR
4595 014272 000421 BR TST24 ;;GO TO NEXT TEST
4596 014274 005037 177772 10$: CLR @PIRQ ;CLEAR LEVEL 1
4597 014300 104171 ERROR 171 ;BEN 13 FAILED
4598 014302 005037 177772 11$: CLR @PIRQ
4599 014306 012737 032666 000004 MOV #CPUSPUR,@ERRVEC ;RESTORE LOCATION 4
4600 014314 012706 001100 MOV #STACK,SP ;RESTORE THE SP
4601 014320 005037 177766 CLR @CPUERR ;CLEAR ERROR REG
4602 014324 104172 ERROR 172 ;BEN 13 FAILED OR TRAP VECTOR FAILED
4603 014326 005037 177772 12$: CLR @PIRQ ;CLEAR LEVEL 1
4604 014332 104377 ERROR 377 ;EITHER TMCB E53 IS NOT GOING HIGH
4605 014334 000454 454 ;OR TMCB PIRQ DOES NOT GO LOW. (BEN 13 FAILURE)
4606 *****
4607 ;*TEST 24 PIR LEVEL 2 INTERRUPT
4608 ;*
4609 ;* IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.
4610 ;* THIS WILL ONLY HAPPEN IF TMCB E63(2) IS BAD.
4611 ;*
4612 ;* IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(2)
4613 ;* IS BAD OR TMCB HONOR PIR2 IS BEING HELD HIGH.
4614 ;* *****
4615 014336 000004 TST24: SCOPE
4616 014340 012767 014474 164722 MOV #TST25,NEXTTST ;SAVE ADDRESS OF NEXT TEST
4617 014346 012706 001100 MOV #STACK,SP ;SETUP THE SP
4618 014352 012737 014406 000000 MOV #1$,@#0 ;SETUP LOC. 0 TO CATCH BEN 13 FAILURE
4619 014360 012737 014422 000240 MOV #2$,@PIRQVEC ;SETUP PIRQ VECTOR
4620 014366 000231 SPL 1 ;SET CPU PRIORITY TO 1
4621 014370 052737 002000 177772 BIS #BIT10,@PIRQ ;SET LEVEL 2
4622 014376 005037 177772 CLR @PIRQ ;CLEAR LEVEL 2
4623 014402 104174 ERROR 174 ;PIR 2 DID NOT INTERRUPT
4624 014404 000406 BR 2$
4625 014406 005037 177772 1$: CLR @PIRQ ;CLEAR LEVEL 2
4626 014412 012737 032666 000004 MOV #CPUSPUR,@ERRVEC ;RESTORE LOCATION 4
4627 014420 104175 ERROR 175 ;BEN 13 FAILED
4628 ;ENSURE PIR09 & 10 NOT SHORTED
4629 014422 012767 014430 164460 2$: MOV #64$, $LPERR ;SETUP ERROR LOOP
4630 014430 005037 177772 64$: CLR @PIRQ ;CLEAR LEVEL 2
4631 014434 012737 014460 000240 MOV #3$,@PIRQVEC ;SETUP VECTOR
4632 014442 000231 SPL 1 ;SET LEVEL 1
4633 014444 052737 001000 177772 BIS #BIT9,@PIRQ ;SET PIR 1
4634 014452 005037 177772 CLR @PIRQ ;CLEAR PIR 1
4635 014456 000406 BR TST25 ;;GO TO NEXT TEST
4636 014460 013767 177772 164530 3$: MOV @PIRQ,$EPIRQ ;SAVE PIRQ FOR TYPEOUT
4637 014466 005037 177772 CLR @PIRQ ;CLEAR LEVEL 1
4638 014472 104176 ERROR 176 ;PIR09 & 10 SHORTED
4639 *****
4640 ;*TEST 25 PIR LEVEL 3 INTERRUPT

```

```

4641
4642
4643
4644
4645
4646
4647
4648 014474 000004
4649 014476 012767 014632 164564
4650 014504 012706 001076
4651 014510 012737 014560 000240
4652 014516 012737 014544 000000
4653 014524 000232
4654 014526 052737 004000 177772
4655 014534 005037 177772
4656 014540 104177
4657 014542 000406
4658 014544 005037 177772
4659 014550 012737 032666 000004
4660 014556 104200
4661 014560 012767 014566 164322
4662 014566 005037 177772
4663 014572 012737 014616 000240
4664 014600 000232
4665 014602 052737 002000 177772
4666 014610 005037 177772
4667 014614 000406
4668 014616 013767 177772 164372
4669 014624 005037 177772
4670 014630 104201
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681 014632 000004
4682 014634 012767 015002 164426
4683 014642 012706 001100
4684 014646 012737 014720 000240
4685 014654 012737 014704 000000
4686 014662 000233
4687 014664 052737 010000 177772
4688 014672 012737 032666 000004
4689 014700 104202
4690 014702 000406
4691 014704 005037 177772
4692 014710 012737 032666 000004
4693 014716 104203
4694 014720 012767 014726 164162
4695 014726 005037 177772
4696 014732 012737 032666 000004

:
:
: IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.
: THIS WILL ONLY HAPPEN IF TMCB E63(3) IS BAD.
:
:
: IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(3)
: IS BAD OR TMCB HONOR PIR3 IS BEING HELD HIGH.
:
:*****
TST25: SCOPE
MOV #TST26,NEXTTST ;SAVE ADDRESS OF NEXT TEST
MOV #1076,SP ;INITIALIZE THE SP
MOV #1$,@#PIRQVEC ;SETUP PIRQ VECTOR
MOV #2$,@#0 ;SETUP LOCATION 0
SPL 2 ;SET CPU AT LEVEL 2
BIS #BIT11,@#PIRQ ;SET LEVEL 3 PIR
CLR @#PIRQ ;CLEAR LEVEL 3
ERROR 177 ;INTERRUPT FAILED
BR 1$
2$: CLR @#PIRQ ;CLEAR LEVEL 3
MOV #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
ERROR 200 ;BEN 13 FAILED
1$: MOV #64$,SLPERR ;SETUP ERROR LOOP
64$: CLR @#PIRQ ;CLEAR LEVEL 3
MOV #3$,@#PIRQVEC ;SETUP PIRQ VECTOR
SPL 2 ;SET CPU PRIORITY AT 2
BIS #BIT10,@#PIRQ ;ENABLE PIR2
CLR @#PIRQ ;CLEAR LEVEL 2
BR TST26 ;GO TO NEXT TEST
3$: MOV @#PIRQ,$EPIRQ ;SAVE PIRQ
CLR @#PIRQ ;CLEAR IT
ERROR 201 ;LEVEL 2 INTERRUPT WHEN CPU LEVEL
;2 ENABLED.
:*****
: *TEST 26 PIR LEVEL 4 INTERRUPT
:
:
: IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.
: THIS WILL ONLY HAPPEN IF TMCB E63(5) IS BAD.
:
:
: IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(5)
: IS BAD OR TMCA HONOR PIR 4 IS BEING HELD HIGH.
:
:*****
TST26: SCOPE
MOV #TST27,NEXTTST ;SAVE ADDRESS OF NEXT TEST
MOV #STACK,SP ;INITIALIZE THE SP
MOV #1$,@#PIRQVEC ;SETUP PIRQ VEC
MOV #2$,@#0 ;SETUP LOCATION 0
SPL 3 ;SET CPU AT 3
BIS #BIT12,@#PIRQ ;ENABLE PIR 4
MOV #CPUSPUR,@#ERRVEC
ERROR 202 ;INTERRUPT DID NOT OCCUR
BR 1$
2$: CLR @#PIRQ ;CLEAR LEVEL 4
MOV #CPUSPUR,@#ERRVEC
ERROR 203 ;BEN 13 FAILED
1$: MOV #64$,SLPERR ;SETUP ERROR LOOP
64$: CLR @#PIRQ ;CLEAR LEVEL 4
MOV #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4

```

```

4697 014740 012737 014766 000240      MOV    #3$,@#PIRQVEC    ;SETUP PIRQ VECTOR
4698 014746 000233              SPL    3                ;SET CPU AT LEVEL 3
4699 014750 052737 004000 177772      BIS    #BIT11,@#PIRQ    ;SET LEVEL 3
4700 014756 005037 177772      CLR    @#PIRQ           ;CLEAR LEVEL 3
4701 014762 000237              SPL    7
4702 014764 000406              BR     TST27            ;;GO TO NEXT TEST
4703 014766 013767 177772 164222 3$:  MOV    @#PIRQ,$EPIRQ    ;SAVE PIRQ
4704 014774 005037 177772      CLR    @#PIRQ
4705 015000 104204      ERROR  204              ;LEVEL 3 INTERRUPT WHEN CPU LEVEL 3 ENABLED
4706
4707      ;*****
4708      ;*TEST 27      PIR LEVEL 5 INTERRUPT
4709
4710      ;*
4711      ;*      IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.
4712      ;*      THIS WILL ONLY HAPPEN IF TMCB E63(11) IS BAD.
4713
4714      ;*
4715      ;*      IF THE INTERRUPT DOESN'T OCCUR THEN EITHER TMCB E62(11)
4716      ;*      IS BAD OR TMCA HONOR PIR 5 IS BEING HELD HIGH.
4717      ;*      ;*****
4718      ;*
4719      ;*      TST27: SCOPE
4720      ;*      MOV    #TST30,NEXTTST ;SAVE ADDRESS OF NEXT TEST
4721      ;*      MOV    #STACK,SP      ;INITIALIZE THE SP
4722      ;*      MOV    #1$,@#PIRQVEC ;SETUP THE PIRQ VECTOR
4723      ;*      MOV    #2$,@#0        ;SETUP LOCATION 0
4724      ;*      SPL    4              ;SET CPU ST LEVEL 4
4725      ;*      BIS    #BIT13,@#PIRQ ;SET LEVEL 5
4726      ;*      CLR    @#PIRQ       ;CLEAR LEVEL 5
4727      ;*      MOV    #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4728      ;*      ERROR  205          ;INTERRUPT DID NOT OCCUR
4729      ;*      BR     1$
4730      ;*      CLR    @#PIRQ       ;CLEAR LEVEL 5
4731      ;*      MOV    #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4732      ;*      ERROR  206          ;BEN 13 FAILED
4733      ;*      MOV    #64$, $LPERR   ;SETUP ERROR LOOP
4734      ;*      MOV    #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4735      ;*      CLR    @#PIRQ       ;CLEAR LEVEL 5
4736      ;*      MOV    #3$,@#PIRQVEC ;SETUP PIRQ VECTOR
4737      ;*      SPL    4              ;SET CPU AT LEVEL 4
4738      ;*      MOV    #BIT12,@#PIRQ ;SET LEVEL 4
4739      ;*      CLR    @#PIRQ       ;CLEAR LEVEL 4
4740      ;*      SPL    7
4741      ;*      BR     TST30          ;;GO TO NEXT TEST
4742      ;*      MOV    @#PIRQ,$EPIRQ ;SAVE PIRQ
4743      ;*      CLR    @#PIRQ       ;CLEAR LEVELS
4744      ;*      ERROR  207          ;LEVEL 4 INT. WHEN CPU LEVEL 4 ENABLED
4745
4746      ;*****
4747      ;*TEST 30      PIR LEVEL 6 INTERRUPT
4748
4749      ;*
4750      ;*      IF BEN13 FAILS EXECUTION WILL GO TO BRK.20.
4751      ;*
4752      ;*      THIS WILL ONLY HAPPEN IF EITHER TMCB E63(12)
4753      ;*      IS BAD, OR E61(1) IS BAD.
4754
4755      ;*
4756      ;*      IF THE INTERRUPT DOES NOT OCCUR LEVEL 7 IS TRYED, TO TRY
4757      ;*      AND ISOLATE THE FAILURE BEFORE TMCB E55(9-8).
4758      ;*      ;*****
4759      ;*
4760      ;*      TST30: SCOPE
    
```

4752 015156 000004

```

4753 015160 012767 015356 164102      MOV      #TST31,NEXTTST  ;SAVE ADDRESS OF NEXT TEST
4754 015166 012706 001100                MOV      #STACK,SP      ;INITIALIZE THE SP
4755 015172 012737 015274 000240      MOV      #1$,@#PIRQVEC  ;SETUP THE PIRQ VECTOR
4756 015200 012737 015260 000000      MOV      #2$,@#0        ;SETUP LOCATION 0
4757 015206 000235                    SPL      5                ;SET THE CPU AT LEVEL 5
4758 015210 052737 040000 177772      BIS      #BIT14,@#PIRQ   ;SET LEVEL 6
4759 015216 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4760 015224 012737 015250 000240      MOV      #3$,@#PIRQVEC  ;SETUP THE PIRQ VECTOR
4761 015232 012737 100000 177772      MOV      #BIT15,@#PIRQ   ;SET LEVEL 7
4762 015240 005037 177772                CLR      @#PIRQ          ;CLEAR LEVEL 7
4763 015244 104210                    ERROR    210              ;FAILURE IS AFTER TMCB E70
4764 015246 000412                    BR       1$
4765 015250 005037 177772      3$:    CLR      @#PIRQ          ;CLEAR LEVEL 7
4766 015254 104211                    ERROR    211              ;FAILURE IS IN TMCB E70 OR BEFORE
4767 015256 000406                    BR       1$
4768 015260 012737 032666 000004      2$:    MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4769 015266 005037 177772                CLR      @#PIRQ          ;CLEAR LEVEL 6
4770 015272 104212                    ERROR    212              ;BEN 13 FAILED
4771 015274 012767 015302 163606      1$:    MOV      #64$, $LPERR    ;SETUP ERROR LOOP
4772 015302 005037 177772      64$:   CLR      @#PIRQ          ;CLEAR LEVEL 6
4773 015306 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4774 015314 012737 015342 000240      MOV      #4$,@#PIRQVEC  ;SETUP PIRQ VECTOR
4775 015322 000235                    SPL      5                ;SET PRIORITY AT 5
4776 015324 012737 020000 177772      MOV      #BIT13,@#PIRQ   ;SET LEVEL 5
4777 015332 005037 177772                CLR      @#PIRQ          ;CLEAR LEVEL 5
4778 015336 000237                    SPL      7
4779 015340 000406                    BR       TST31           ;:GO TO NEXT TEST
4780 015342 013767 177772 163646      4$:    MOV      @#PIRQ,$EPIRQ    ;:SAVE PIRQ
4781 015350 005037 177772                CLR      @#PIRQ          ;:CLEAR LEVEL 5
4782 015354 104213                    ERROR    213              ;:LEVEL 5 INTERRUPT WHEN CPU 5 ENABLED

```

\*\*\*\*\*  
: \*TEST 31 PIR LEVEL 7 INTERRUPT

: \*  
: \* IF BEN 13 FAILS EXECUTION WILL GO TO BRK.20.  
: \* THIS WILL ONLY HAPPEN IF TMCB E63(6) IS BAD.

: \*  
: \* IF THE INTERRUPT DOES NOT OCCUR THEN EITHER TMCB E70(6)  
: \* IS BAD OR TMCA HONOR PIR7 IS BEING HELD HIGH.

\*\*\*\*\*

```

4792 015356 000004                    TST31: SCOPE
4793 015360 012767 015530 163702      MOV      #TST32,NEXTTST  ;SAVE ADDRESS OF NEXT TEST
4794 015366 012706 001100                MOV      #STACK,SP      ;INITIALIZE THE SP
4795 015372 012737 015450 000240      MOV      #1$,@#PIRQVEC  ;SETUP THE PIRQ VECTOR
4796 015400 012737 015434 000000      MOV      #2$,@#0        ;SETUP LOCATION 0
4797 015406 000236                    SPL      6                ;SET PRIORITY AT LEVEL 6
4798 015410 052737 100000 177772      BIS      #BIT15,@#PIRQ   ;SET LEVEL 7
4799 015416 005037 177772                CLR      @#PIRQ          ;CLEAR LEVEL 7
4800 015422 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4801 015430 104214                    ERROR    214              ;LEVEL 7 DID NOT INTERRUPT
4802 015432 000406                    BR       1$
4803 015434 005037 177772      2$:    CLR      @#PIRQ          ;CLEAR LEVEL 7
4804 015440 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4805 015446 104215                    ERROR    215              ;BEN 13 FAILED
4806 015450 012767 015456 163432      1$:    MOV      #64$, $LPERR    ;SETUP ERROR LOOP
4807 015456 012737 032666 000004      64$:   MOV      #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4808 015464 012737 015514 000240      MOV      #3$,@#PIRQVEC  ;SETUP PIRQ VECTOR

```

```
4809 015472 005037 177772          CLR    @#PIRQ          ;CLEAR LEVEL 7
4810 015476 000236                   SPL    6              ;SET LEVEL 6 IN CPU
4811 015500 012737 040000 177772    MOV    #BIT14,@#PIRQ  ;SET PIR 6
4812 015506 005037 177772          CLR    @#PIRQ          ;CLEAR PIR 6
4813 015512 000406                   BR     TST32          ;;GO TO NEXT TEST
4814 015514 013767 177772 163474 3$: MOV    @#PIRQ,$EPIRQ
4815 015522 005037 177772          CLR    @#PIRQ          ;CLEAR LEVEL 6
4816 015526 104216                   ERROR  216           ;LEVEL 6 INT. WHEN CPU AT 6
4817
4818
4819      ;*****
4820      ;*TEST 32          UNIBUS TIMEOUT
4821      ;*
4822      ;*      IF TMCC ABORT DOES NOT GO HIGH OR DOES NOT GET TO RACA
4823      ;*      OR IF RACA ZAP DOES NOT GO LOW THE PROCESSOR WILL NOT TRAP TO 4.
4824      ;*
4825      ;*      IF BEN06 FAILS THE STACKED PC WILL BE 160000 INSTEAD OF 1$-2.
4826      ;*
4827      ;*      IF BEN 13 FAILS EITHER TMCC AERF(1) L IS NOT GOING LOW
4828      ;*      OR TMCB E53(11) IS BAD.
4829      ;*      A TEST IS THEN MADE TO ENSURE THAT TMCC PRIORITY CLEAR GOES LOW.
4830      ;*****
4830 015530 000004                   TST32: SCOPE
4831 015532 012767 016650 163530    MOV    #TST33,NEXTTST ;SAVE ADDRESS OF NEXT TEST
4832 015540 012737 015632 000004    MOV    #2$,@#ERRVEC   ;SETUP TIMEOUT VECTOR
4833 015546 032737 000020 177776    BIT    #BIT4,@#PSW    ;IS T BIT ON?
4834 015554 001404                   BEQ    19$            ;BRANCH IF NO
4835 015556 012737 000360 000006    MOV    #360,@#ERRVEC+2 ;SETUP ERROR VEC FOR T BIT
4836 015564 000403                   BR     17$            ;
4837 015566 012737 000340 000006 19$: MOV    #PR7,@#ERRVEC+2 ;SETUP ERROR VEC WITHOUT T BIT
4838 015574 012767 015602 163306 17$: MOV    #12$, $LPERR  ;SETUP ERROR LOOP
4839 015602 012706 001100 12$: MOV    #STACK,SP    ;INITIALIZE THE SP
4840
4841      ;EXECUTE A TIMEOUT ON A DATI
4842 015606 013700 160000          MOV    @#160000,RO    ;EXECUTE TIMEOUT ON DATI
4843      ;FAILURE-DID NOT TIMEOUT OR AERF DID NOT GO LOW
4844 015612 032737 000020 177766    BIT    #BIT4,@#CPUERR ;DID TIMEOUT FLAG SET?
4845 015620 001002                   BNE    1$            ;BRANCH IF YES
4846 015622 104217                   ERROR  217           ;DID NOT TIMEOUT
4847 015624 000423                   BR     7$            ;
4848 015626 104220          1$: ERROR  220           ;BEN 13 FAILED
4849 015630 000421                   BR     7$            ;
4850 015632 022716 160000          2$: CMP    #160000,(SP) ;DID 160000 GET STACKED?
4851 015636 001003                   BNE    16$           ;BRANCH IF NO
4852 015640 104377                   ERROR  377           ;EITHER PS RESTORE STUCK HIGH
4853      ;OR RACK E63(B0) BAD
4854 015642 000455                   455
4855 015644 000413                   BR     7$            ;
4856 015646 012737 015674 000004 16$: MOV    #7$,@#ERRVEC  ;SETUP TIMEOUT VECTOR
4857 015654 012767 015662 163226    MOV    #13$, $LPERR  ;SETUP ERROR LOOP
4858 015662 012706 001100 13$: MOV    #STACK,SP    ;INITIALIZE THE SP
4859
4860      ;EXECUTE A TIMEOUT ON A DATO
4861 015666 010037 160000          MOV    RO,@#160000    ;EXECUTE TIMEOUT ON DATO
4862 015672 104221                   ERROR  221           ;DID NOT TIMEOUT
4863      ;PRIORITY CLEAR ON ABORT
4864      ;ENSURES PIR VECTOR DOES NOT COME IN ON ABORT
```

```

CEKBBE0 11/70-74MP CPU #2          MACY11 30A(1052) 10-MAR-80 09:32 C 8 PAGE 94
CEKBBE.P11 10-MAR-80 09:31         T32      UNIBUS TIMEOUT                               SEQ 0093

4865 015674 000237          7$:   SPL      7          ;ENSURE CPU AT LEVEL 7
4866 015676 012767 015746 163204   MOV      #15$, $LPERR ;SETUP THE ERROR LOOP
4867 015704 012737 015766 000004   MOV      #11$, @#ERRVEC ;SETUP ERRVEC
4868 015712 012737 015774 000240   MOV      #9$, @#PIRQVEC ;SETUP PIRQ VECTOR
4869 015720 032737 000020 177776   BIT      #BIT4, @#PSW   ;IS T BIT ON?
4870 015726 001404          BEQ      18$          ;BRANCH IF NO
4871 015730 012737 000360 000242   MOV      #360, @#PIRQVEC+2 ;SETUP PIRQ PSW
4872 015736 000403          BR       15$          ;
4873 015740 012737 000340 000006 18$:   MOV      #PR7, @#ERRVEC+2 ;SETUP NEW PSW
4874 015746 012706 001100 15$:   MOV      #STACK, SP    ;INITIALIZE THE SP
4875 015752 052737 100000 177772   BIS      #BIT15, @#PIRQ ;SET PIR LEVEL 7
4876 015760 000236          SPL      6
4877 015762 005237 160000          INC      @#160000      ;EXECUTE REFERENCE TO BAD ADDRESS
4878 015766 005037 177772          11$:   CLR      @#PIRQ        ;CLEAR PIRQ REGISTER
4879 015772 000404          BR       10$          ;SKIP OVER ERROR CALL
4880 015774 005037 177772          9$:   CLR      @#PIRQ        ;CLEAR PIRQ REG
4881 016000 104377          ERROR   377          ;TMCC PRIORITY CLEAR DID NOT GO LOW
4882 016002 000435          435
4883 016004 012737 032666 000004 10$:   MOV      #CPUSPUR, @#ERRVEC ;RESTORE ERRVEC
4884          ;CPU ERROR REGISTER BIT 4 TEST
4885 016012 022737 000020 177766   CMP      #BIT4, @#CPUERR ;DID CPU ERROR REG TIMEOUT BIT SET?
4886 016020 001407          BEQ      5$          ;BRANCH IF YES
4887 016022 013767 177766 163124   MOV      @#CPUERR, $REGO ;SAVE REG FOR TYPEOUT
4888 016030 012767 000020 163124   MOV      #BIT4, $TMP0   ;SAVE EXPECTED VALUE
4889 016036 104255          ERROR   255          ;CPU ERROR REG FAILED TO SET
4890 016040 005037 177766          5$:   CLR      @#CPUERR      ;CLEAR OUT BIT 4
4891 016044 022737 000000 177766   CMP      #0, @#CPUERR  ;DID REGISTER CLEAR?
4892 016052 001401          BEQ      PDINTE      ;;BRANCH IF YES
4893 016054 104256          ERROR   256          ;CPU ERROR REG DOES NOT CLEAR
4894          ;*****
4895          ;SBTTL PERIPHERAL DETERMINATOR & INTERRUPT ENABLE ROUTINES
4896          ;* THIS SECTION OF CODE TRYS TO FIND A DEVICE ON BR5 AND BR6.
4897          ;* WHEN IT FINDS A DEVICE IT PUTS THE ADDRESS OF THAT DEVICES
4898          ;* SUBROUTINE IN A LOCATION.
4899          ;*
4900          ;* THE CODE TO INITIATE AN INTERRUPT SEQUENCE ON CERTAIN
4901          ;* DEVICES IS ALSO HERE. WHEN A TEST REQUIRES AN INTERRUPT ON
4902          ;* A CERTAIN LEVEL IT LOOKS AT INTERX TO DETERMINE IF A
4903          ;* DEVICE IS AVAILABLE AND IF SO DOES A JSR TO THAT DEVICES
4904          ;* INTERRUPT ENABLE ROUTINE.
4905          ;*****
4906 016056          PDINTE:
4907 016056 012767 016650 163110   MOV      #TST33, $ESCAPE ;SAVE START ADDRESS OF NEXT TEST
4908 016064 012767 016650 163176   MOV      #TST33, NEXTTST ;SAVE START ADDRESS OF NEXT TEST
4909 016072 005767 163002          TST      $PASS        ;IS THIS FIRST PASS?
4910 016076 001004          BNE      1$          ;BRANCH IF NO
4911 016100 032737 040000 177570   BIT      #SW14, @#SWR   ;IS SWITCH 14 ON?
4912 016106 001402          BEQ      2$          ;BRANCH IF NO
4913 016110 000177 163154          1$:   JMP      @NEXTTST      ;GO TO NEXT TEST
4914 016114 012706 001100          2$:   MOV      #STACK, SP    ;INITIALIZE THE SP
4915 016120 012737 016134 000004   MOV      #3$, @#ERRVEC ;SETUP ERROR VECTOR
4916 016126 005777 163106          TST      @RSCS1       ;IS RS AVAILABLE?
4917 016132 000430          BR       6$          ;YES
4918 016134 012737 016150 000004 3$:   MOV      #4$, @#ERRVEC ;SETUP ERROR VECTOR
4919 016142 005777 163076          TST      @RPCS1       ;IS RP AVAILABLE?
4920 016146 000431          BR       7$          ;YES

```

D 8

CEKB8E0 11/70-74MP CPU #2      MACY11 30A(1052) 10-MAR-80 09:32 PAGE 95  
CEKB8E.P11      10-MAR-80 09:31      PERIPHERAL DETERMINATOR & INTERRUPT ENABLE ROUTINES      SEQ 0094

```

4921 016150 012737 016164 000004 4$:  MOV    #5$,@#ERRVEC    ;SETUP ERROR VECTOR
4922 016156 005777 163072                TST    @TMCS1          ;IS TM AVAILABLE?
4923 016162 000432                BR     8$              ;YES
4924 016164 012737 016274 000004 5$:  MOV    #10$,@#ERRVEC   ;SETUP ERROR VECTOR
4925 016172 005777 163052                TST    @RKCS1         ;IS RK AVAILABLE?
4926 016176 016767 163046 163024    MOV    RKCS1,INT5ST    ;YES,SAVE RK STATUS
4927 016204 016767 163042 163014    MOV    RKVEC,INT5VEC   ;SAVE RK VECTOR
4928 016212 000424                BR     9$              ;EXIT
4929 016214 016767 163020 163006    MOV    RSCS1,INT5ST    ;SAVE RS STATUS
4930 016222 016767 163014 162776    MOV    RSVEC,INT5VEC   ;SAVE RS VECTOR
4931 016230 000415                BR     9$              ;EXIT
4932 016232 016767 163006 162770    MOV    RPCS1,INT5ST    ;SAVE RP STATUS
4933 016240 016767 163002 162760    MOV    RPVEC,INT5VEC   ;SAVE RP VECTOR
4934 016246 000406                BR     9$              ;EXIT
4935 016250 016767 163000 162752    MOV    TMCS1,INT5ST    ;SAVE TM STATUS
4936 016256 016767 162774 162742    MOV    TMVEC,INT5VEC   ;SAVE TM VECTOR
4937 016264 012767 016464 162732    MOV    #INT5SU,INTERS  ;SAVE ADDRESS OF INTER 5 SUBROUTINE
4938 016272 000406                BR     LEVE6           ;GO CHECK LEVEL 6
4939 016274 032737 000440 177570 10$:  BIT    #440,@#SWR      ;SWITCH 8 OR 5 ON?
4940 016302 001002                BNE   LEVE6           ;BRANCH IF YES
4941 016304 104400 072432                TYPE   ,EM710
4942
4943                ;:FIND OUT WHAT IS AVAILABLE ON LEVEL 6
4944 016310 012737 016360 000004  LEVE6: MOV    #1$,@#ERRVEC   ;SETUP LOCATION 4
4945 016316 005777 162736                TST    @LKSTAT        ;IS LINE CLOCK AVAILABLE?
4946 016322 012767 016530 162702    MOV    #SKW11L,INTER6 ;PUT ADDRESS OF KW11-L SUBROUTINE IN STORAGE
4947 016330 016767 162726 162676    MOV    LKVEC,INT6VEC   ;SAVE BR 6 INTERR VEC
4948 016336 016767 162716 162672    MOV    LKSTAT,INT6ST   ;SAVE ADDRESS OF BR 6 STATUS
4949 016344 012737 032666 000004    MOV    #CPUSPUR,@#ERRVEC
4950 016352 005037 177766                CLR    @#CPUERR        ;ENSURE TIMEOUT BIT CLEAR
4951 016356 000534                BR     TST33           ;:PERIPHERAL DETERMINATOR FINISHED
4952 016360 104400 073121 1$:    TYPE   ,EM725
4953
4954                ;:LINE CLOCK NOT AVAILABLE, SEE IF PROGRAMMABLE CLOCK AVAILABLE
4955 016364 012737 016434 000004    MOV    #2$,@#ERRVEC   ;SETUP LOCATION 4
4956 016372 005777 162666                TST    @PLKSTAT       ;IS PROGRAMMABLE AVAILABLE?
4957 016376 012767 016574 162626    MOV    #SKW11P,INTER6 ;YES, PUT ADDRESS OF KW11-P SUBROUTINE IN STORAGE
4958 016404 016767 162656 162622    MOV    PLKVEC,INT6VEC  ;SAVE BR 6 INTERR VEC
4959 016412 016767 162646 162616    MOV    PLKSTAT,INT6ST  ;SAVE ADDRESS OF BR 6 STATUS
4960 016420 005037 177766                CLR    @#CPUERR        ;ENSURE TIMEOUT BIT CLEAR
4961 016424 012737 032666 000004    MOV    #CPUSPUR,@#ERRVEC ;RESTORE ERROR VECTOR
4962 016432 000506                BR     TST33           ;:PERIPHERAL DETERMINATOR FINISHED
4963 016434 012737 032666 000004 2$:  MOV    #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
4964 016442 005037 177766                CLR    @#CPUERR        ;ENSURE TIMEOUT BIT CLEAR
4965 016446 032737 000500 177570    BIT    #500,@#SWR      ;SWITCH 6 OR 8 ON?
4966 016454 001002                BNE   3$              ;BRANCH IF YES
4967 016456 104400 072452                TYPE   ,EM711         ;TYPE MESSAGE(NO DEVICE AVAILABLE)
4968 016462
4969 016462 000472 3$:    BR     TST33           ;:PERIPHERAL DETERMINATOR FINISHED
4970
4971                ;:THIS CODE SETS UP THE DEVICE ON LEVEL 5 TO INTERRUPT
4972                ;:AND IS CALLED BY A JSR PC,@INTERS
4972 016464 011600  INT5SU: MOV    (SP),R0        ;SAVE RETURN PC
4973 016466 012777 016520 162532    MOV    #ENDBR5,@INT5VEC ;SETUP LEVEL 5 VECTOR
4974 016474 005001                CLR    R1              ;SETUP WAIT COUNTER
4975 016476 012777 000311 162524    MOV    #311,@INT5ST    ;SET LEVEL 5 INTERRUPT
4976 016504 005201 2$:    INC    R1              ;WAIT FOR

```



```
4977 016506 001376          BNE 2$          : INTERRUPT
4978 016510 005077 162514    CLR @INT5ST     : CLEAR INTERRUPT FLAG
4979 016514 062700 000002    ADD #2,R0      : ADJUST RETURN PC
4980 016520 005077 162504    ENDBR5: CLR @INT5ST : CLEAR LEVEL 5 STATUS
4981 016524 010046          MOV R0,-(SP)   : PUT RETURN PC ON STACK
4982 016526 000207          RTS PC        : RETURN
4983
4984          : THIS CODE SETS UP THE KW11-L TO INTERRUPT AND IS CALLED BY A JSR PC,@INTER6
4985 016530 011600          $KW11L: MOV (SP),R0   : SAVE THE RETURN PC
4986 016532 012777 016566 162474  MOV #1$,@INT6VEC : SETUP INTERRUPT VECTOR
4987 016540 005001          CLR R1        : SETUP COUNTER
4988 016542 012777 000100 162466  MOV #BIT6,@INT6ST : SET INTERRUPT ENABLE BIT & CLR MONITOR BIT
4989 016550 005201 2$:      INC R1        : WAIT FOR
4990 016552 001376          BNE 2$        : INTERRUPT
4991 016554 005077 162456    CLR @INT6ST    : CLEAR INTERRUPT BIT
4992 016560 062700 000002    ADD #2,R0      : ADJUST RETURN PC
4993 016564 000427          BR $ENDBR6    : RETURN
4994 016566 005077 162444    1$: CLR @INT6ST : CLEAR INTERRUPT FLAG
4995 016572 000424          BR $ENDBR6    : RETURN
4996
4997          : THIS CODE SETS UP THE KW11-P TO INTERRUPT AND IS CALLED BY A JSR PC,@INTER6
4998 016574 011600          $KW11P: MOV (SP),R0   : SAVE THE RETURN PC
4999 016576 012777 016640 162430  MOV #1$,@INT6VEC : SETUP THE INTERRUPT VECTOR
5000 016604 012737 000001 172544  MOV #1,@#PLKC   : SET THE COUNTER FOR ONE COUNT
5001 016612 005001          CLR R1        : SETUP THE WAIT COUNTER
5002 016614 012777 000105 162414  MOV #105,@INT6ST : START COUNTER
5003 016622 005201 2$:      INC R1        : WAIT FOR
5004 016624 001376          BNE 2$        : INTERRUPT
5005 016626 005077 162404    CLR @INT6ST    : CLEAR INTERRUPT BIT
5006 016632 062700 000002    ADD #2,R0      : ADJUST R0 FOR RETURN
5007 016636 000402          BR $ENDBR6    : RETURN
5008 016640 005077 162372    1$: CLR @INT6ST : CLEAR INTERRUPT FLAG
5009 016644 010046          $ENDBR6: MOV R0,-(SP) : PUT RETURN PC ON STACK.
5010 016646 000207          RTS PC        : RETURN
5011
5012          : *****
5013          : *TEST 33      BR LEVEL 4 INTERRUPT
5014          : *
5015          : *      BEN 13 SHOULD NOT FAIL.
5016          : *      IF THE INTERRUPT DOESN'T OCCUR AN ATTEMPT IS MADE TO
5017          : *      ISOLATE THE FAILURE.
5018          : *****
5019 016650 000004          TST33: SCOPE
5020 016652 012767 017144 162410  MOV #TST34,NEXTTST : SAVE ADDRESS OF NEXT TEST
5021 016660 012767 003706 162216  MOV #^D1990,$ICNT  : ADJUST ITERATION COUNT
5022 016666 012767 016674 162212  MOV #14$,$LPADR    : SETUP LOOP ADR
5023 016674 032737 000400 177570  14$: BIT #SW8,@#SWR  : SWITCH 8 ON?
5024 016702 001012          BNE 8$        : BRANCH IF YES
5025 016704 032737 000020 177570  BIT #SW4,@#SWR    : IS SWITCH 4 ON?
5026 016712 001406          BEQ 8$        : BRANCH IF NO
5027 016714 005767 162160    TST $PASS      : IS THIS FIRST PASS?
5028 016720 001002          BNE 9$        : BRANCH IF NO
5029 016722 104400 072472    TYPE ,EM712    : TYPE MESSAGE (SKIPPING TEST)
5030 016726          9$:
5031 016726 000506          BR TST34      : GO TO NEXT TEST
5032 016730 012706 001100    8$: MOV #STACK,SP  : INITIALIZE THE SP
```

```

5033 016734 012737 017010 000064      MOV      #1$,@#TPVEC      ;SETUP THE TERMINAL INTERRUPT VECTOR
5034 016742 032737 000020 177776      BIT      #BIT4,@#PSW      ;IS T BIT ON?
5035 016750 001404                BEQ      10$              ;BRANCH IF NO
5036 016752 012737 000360 000066      MOV      #360,@#TPVEC+2  ;SETUP TPVEC PSW
5037 016760 000403                BR       11$
5038 016762 012737 000340 000066 10$:      MOV      #PR7,@#TPVEC+2  ;PUT PRIORITY 7 IN NEW PSW
5039 016770 000237 11$:      SPL      7                ;SET CPU AT LEVEL 7
5040 016772 005000                CLR      R0                ;SETUP R0
5041 016774 052777 000100 162140      BIS      #100,@#STPS      ;GET INTERRUPT ON BR 4
5042 017002 005200 2$:      INC      R0                ;WAIT AND SEE
5043 017004 001376                BNE      2$              ;IF INTERRUPT OCCURS
5044 017006 000403                BR       3$              ;NO INTERRUPT
5045 017010 005077 162126 1$:      CLR      @#STPS          ;CLEAR INTERRUPT ENABLE
5046 017014 104222                ERROR    222              ;EITHER TMCB PS07(0) IS NOT GETTING TO
5047                                ;TMCB E77 OR E77 IS BAD
5048 017016 3$:
5049 017016 012767 017144 162150      MOV      #TST34,$ESCAPE  ;SAVE START ADDRESS OF NEXT TEST
5050 017024 012767 017144 162236      MOV      #TST34,NEXTTST  ;SAVE START ADDRESS OF NEXT TEST
5051 017032 012737 017140 000064      MOV      #4$,@#TPVEC      ;SETUP THE INTERRUPT VECTOR
5052 017040 005000                CLR      R0                ;SETUP R0
5053 017042 000233                SPL      3                ;SET CPU AT LEVEL 3
5054 017044 052777 000100 162070      BIS      #100,@#STPS      ;GET A BR 4
5055 017052 005200 5$:      INC      R0                ;WAIT FOR
5056 017054 001376                BNE      5$              ;INTERRUPT
5057 017056 005077 162060      CLR      @#STPS          ;CLEAR TP INTERRUPT BIT
5058                                ;BR 4 INTERRUPT FAILED. IS THERE A BR 6 DEVICE AVAILABLE?
5059 017062 005767 162144      TST      INTER6          ;TEST LEVEL 6
5060 017066 001422                BEQ      7$
5061 017070 016700 162140      MOV      INT6VEC,R0      ;GET ADDR OF INTER 6 VECTOR
5062 017074 062700 000002      ADD      #2,R0           ;ADJUST TO PSW VEC
5063 017100 032737 000020 177776      BIT      #BIT4,@#PSW      ;IS T BIT ON?
5064 017106 001403                BEQ      12$             ;BRANCH IF NO
5065 017110 012710 000360      MOV      #360,(R0)      ;SETUP PSW
5066 017114 000402                BR       13$
5067 017116 012710 000340 12$:      MOV      #PR7,(R0)      ;SETUP PSW NO T BIT
5068 017122 000235 13$:      SPL      5                ;SET CPU AT 5
5069 017124 004777 162102      JSR      PC,@INTER6     ;EXECUTE INTERRUPT
5070 017130 000402      BR       6$              ;RETURN HERE IF 6 INTERRUPTS
5071 017132 104223                ERROR    223              ;BOTH BR 4 AND BR 6 FAILED
5072 017134 104224                ERROR    224              ;BR 4 FAILED
5073 017136 104225                ERROR    225              ;BR 4 FAILED BUT BR 6 OK
5074 017140 005077 161776 4$:      CLR      @#STPS          ;CLEAR PRINTER INTERRUPT FLAG
5075
5076                                ;:*****
5077                                ;*TEST 34          BR LEVEL 5 INTERRUPT
5078                                ;*
5079                                ;*
5080                                ;*   THE ONLY POSSIBLE FAILURE IS THAT TMCA HONOR BR 5
5081                                ;*   DOES NOT GO LOW OR TMCB E62(6) IS BAD.
5082                                ;:*****
5082 017144 000004      TST34:  SCOPE
5083 017146 012767 017274 162020      MOV      #TST35,$ESCAPE  ;SAVE START ADDRESS OF NEXT TEST
5084 017154 012767 017274 162106      MOV      #TST35,NEXTTST  ;SAVE START ADDRESS OF NEXT TEST
5085 017162 032737 000400 177570      BIT      #SW8,@#SWR      ;SWITCH 8 ON?
5086 017170 001012                BNE      1$              ;BRANCH IF YES
5087 017172 032737 000040 177570      BIT      #SW5,@#SWR      ;IS SWITCH 5 UP?
5088 017200 001406                BEQ      1$              ;BRANCH IF NO
    
```

```

5089 017202 005767 161672      TST    $PASS           ;IS THIS FIRST PASS?
5090 017206 001002              BNE    3$             ;BRANCH IF NO
5091 017210 104400 072517      TYPE   ,EM713        ;TYPE MESSAGE (TEST BEING SKIPPED)
5092 017214              3$:
5093 017214 000427              BR     TST35         ;;GO TO NEXT TEST
5094 017216 005767 162002      1$:  TST    INTER5     ;IS THERE A DEVICE AVAILABLE?
5095 017222 001424              BEQ    TST35         ;BRANCH IF NO
5096 017224 012706 001100      2$:  MOV    #STACK,SP   ;INITIALIZE THE SP
5097 017230 016700 161772      MOV    INT5VEC,R0   ;GET ADDR OF BR 5 VECTOR
5098 017234 062700 000002      ADD    #2,R0        ;ADJUST TO PSW ADDR
5099 017240 032737 000020 177776 BIT    #BIT4,@#PSW  ;IS T BIT ON?
5100 017246 001403              BEQ    5$             ;BRANCH IF NO
5101 017250 012710 000360      MOV    #360,(R0)    ;PUT NEW PSW IN VECTOR
5102 017254 000402              BR     6$             ;
5103 017256 012710 000340      5$:  MOV    #PR7,(R0)    ;PUT NEW PSW IN VEC NO T BIT
5104 017262 000234      6$:  SPL    4             ;SET CPU AT LEVEL 4
5105 017264 004777 161734      JSR    PC,@INTER5   ;EXECUTE LEVEL 5 INTERRUPT
5106 017270 000401              BR     TST35         ;;GO TO NEXT TEST
5107 017272 104226      ERROR  226          ;BR 5 DID NOT INTERRUPT

```

```

*****
*TEST 35      BR LEVEL 6 INTERRUPT
*
*      THE ONLY POSSIBLE FAILURE IS THAT TMCA HONOR BR 6 DOES NOT GO LOW
*      OR TMCB E62(12) IS BAD.
*****

```

```

5115 017274 000004      TST35: SCOPE
5116 017276 012767 017424 161670 MOV    #TST36,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
5117 017304 012767 017424 161756 MOV    #TST36,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
5118 017312 032737 000400 177570 BIT    #SW8,@#SWR     ;SWITCH 8 ON?
5119 017320 001012              BNE    1$             ;BRANCH IF YES
5120 017322 032737 000100 177570 BIT    #SW6,@#SWR     ;IS SWITCH 6 UP?
5121 017330 001406              BEQ    1$             ;BRANCH IF NO
5122 017332 005767 161542      TST    $PASS         ;IS THIS FIRST PASS?
5123 017336 001002              BNE    3$             ;BRANCH IF NO
5124 017340 104400 072544      TYPE   ,EM715        ;TYPE MESSAGE (TEST BEING SKIPPED)
5125 017344              3$:
5126 017344 000427              BR     TST36         ;;GO TO NEXT TEST
5127 017346 005767 161660      1$:  TST    INTER6     ;IS THERE A DEVICE AVAILABLE?
5128 017352 001424              BEQ    TST36         ;BRANCH IF NO
5129 017354 012706 001100      2$:  MOV    #STACK,SP   ;INITIALIZE THE SP
5130 017360 016700 161650      MOV    INT6VEC,R0   ;GET ADDR OF BR 6 VECTOR
5131 017364 062700 000002      ADD    #2,R0        ;ADJUST TO PSW ADDR
5132 017370 032737 000020 177776 BIT    #BIT4,@#PSW  ;IS T BIT ON?
5133 017376 001403              BEQ    5$             ;BRANCH IF NO
5134 017400 012710 000360      MOV    #360,(R0)    ;SETUP NEW PSW
5135 017404 000402              BR     6$             ;
5136 017406 012710 000340      5$:  MOV    #PR7,(R0)    ;SETUP NEW PSW
5137 017412 000235      6$:  SPL    5             ;SET CPU AT LEVEL 5
5138 017414 004777 161612      JSR    PC,@INTER6   ;EXECUTE LEVEL 6 INTERRUPT
5139 017420 000401              BR     TST36         ;;GO TO NEXT TEST
5140 017422 104227      ERROR  227          ;BR 6 DID NOT INTERRUPT

```

```

*****
*TEST 36      YELLOW ZONE TRAP
*

```

5141  
5142  
5143  
5144

```

5145 : * A YELLOW ZONE IS FIRST ATTEMPTED WITH THE SP AT 376.
5146 : * IF BEN 13 FAILS THE TRAP WILL NOT OCCUR.
5147 : *
5148 : * IF THE PROCESSOR FAILS TO TRAP EITHER TMCD SL YEL IS
5149 : * NOT GOING HIGH OR TMCA HONOR SLY IS NOT GOING LOW
5150 : * OR TMCB E70(3) IS BAD OR BEN13 FAILED.
5151 : *
5152 : * IF TMCC PRIORITY CLEAR DOES NOT GO LOW THE PROCESSOR WILL HANG
5153 : * UP IN A RED ZONE TRAP LOOP.
5154 : *
5155 : * A JSR WITH A BAD SP IS THEN EXECUTED TO ENSURE TMCC
5156 : * KERNAL R6 GOES HIGH WHEN ENABLED BY 'STACK REFERENCE * KERNAL MODE'.
5157 : *
5158 : * IF THE TRAP WORKS TESTS WILL BE PERFORMED TO ENSURE ALL THE
5159 : * APPROPRIATE CONDITIONS DISABLE THE TRAP EXCEPT
5160 : * THE PRIORITY ARBITRATOR.
5161 : *
    
```

```

*****
TST36: SCOPE
5162 017424 000004      MOV      #11$, $LPADR      ;SETUP LP ADR
5163 017426 012767 017472 161452      MOV      #11$, $LPERR     ;SETUP ERROR LOOP
5164 017434 012767 017472 161446      MOV      @#PSW, $TMP3     ;SAVE PSW
5165 017442 013767 177776 161520      BIT      #BIT4, @#PSW     ;IS T BIT ON?
5166 017450 032737 000020 177776      BEQ      11$              ;BRANCH IF NO
5167 017456 001405      MOV      #PR7, -(SP)      ;PUT NEW PSW ON STACK
5168 017460 012746 000340      MOV      #11$, -(SP)      ;PUT RETURN ADR ON STACK
5169 017464 012746 017472      RTT                          ;TURN T BIT OFF
5170 017470 000006      MOV      #PR7, @#ERRVEC+2 ;RESTORE ERR VEC PSW
5171 017472 012737 000340 000006 11$:  MOV      #1$, @#ERRVEC     ;SETUP ERRVEC
5172 017500 012737 017536 000004      MOV      #2$, $LPERR     ;SETUP ERROR LOOP
5173 017506 012767 017514 161374      MOV      #376, SP        ;SETUP THE SP
5174 017514 012706 000376 2$:      MOV      #-1, R0          ;SETUP R0
5175 017520 012700 177777      MOV      R0, (SP)        ;EXECUTE INSTRUCTION UNDER TEST
5176 017524 010016      MOV      #STACK, SP      ;REINITIALIZE THE SP
5177 017526 012706 001100      ERROR   230              ;YELLOW ZONE DID NOT OCCUR
5178 017532 104230      BR      8$
5179 017534 000407      CMP      #-1, @#376      ;DID RED ZONE OCCUR?
5180 017536 022737 177777 000376 1$:  BEQ      8$              ;BRANCH IF NO
5181 017544 001403      MOV      #STACK, SP      ;RESTORE SP
5182 017546 012706 001100      ERROR   254              ;RED ZONE IN YELLOW REGION
5183 017552 104254      :
5184 :
5185 : JSR WITH A BAD SP
5186 017554 012737 017614 000004 8$:  MOV      #6$, @#ERRVEC     ;SETUP ERRVEC
5187 017562 012767 017570 161320      MOV      #63$, $LPERR    ;SETUP THE ERROR LOOP
5188 017570 012706 000376 63$:  MOV      #376, SP        ;SETUP THE SP
5189 017574 004767 000000      JSR      PC, 7$          ;EXECUTE INSTRUCTION UNDER TEST
5190 017600 012706 001100 7$:  MOV      #STACK, SP      ;RESET THE SP
5191 017604 012737 032666 000004      MOV      #CPUSPUR, @#ERRVEC ;RESTORE ERRVEC
5192 017612 104232      ERROR   232              ;TMCC KERNAL R6 DID NOT GO HIGH ON JSR
5193 :
5194 : DISABLE WITH STACK LIMIT REGISTER
5195 017614 012737 017642 000004 6$:  MOV      #3$, @#ERRVEC     ;SETUP ERRVEC
5196 017622 012767 017630 161260      MOV      #62$, $LPERR    ;SETUP ERROR LOOP
5197 017630 012706 000740 62$:  MOV      #740, SP        ;SETUP THE SP
5198 017634 013716 000742      MOV      @#742, (SP)     ;EXECUTE INSTRUCTION UNDER TEST
5199 017640 000406      BR      4$              ;GO TO NEXT SECTION
5200 017642 012737 032666 000004 3$:  MOV      #CPUSPUR, @#ERRVEC ;RESTORE ERRVEC
    
```

```
5201 017650 012706 001100      MOV    #STACK,SP      ;RESTORE THE SP
5202 017654 104233              ERROR  233            ;PDRG STACK LIMIT DID NOT DISABLE TRAP
5203                          ;
5204                          ;DISABLE WITH TMCC KERNAL R6
5205                          ;DISABLE KERNAL R6 WITH DATI
5206 017656 012737 017702 000004 4$:  MOV    #5$,@#ERRVEC  ;SETUP ERRVEC
5207 017664 012767 017672 161216  MOV    #61$, $LPERR   ;SETUP THE ERROR LOOP
5208 017672 012706 000376      61$:  MOV    #376,SP       ;SETUP THE SP
5209 017676 011606              MOV    (SP),SP       ;EXECUTE INSTRUCTION UNDER TEST
5210 017700 000415              BR     9$            ;GO TO NEXT SECTION
5211 017702 012706 001100      5$:  MOV    #STACK,SP     ;RESTORE THE SP
5212 017706 012737 032666 000004  MOV    #CPUPUR,@#ERRVEC ;RESTORE ERRVEC
5213 017714 104234              ERROR  234            ;TMCC KERNAL R6 DID NOT DISABLE ON DATI
5214 017716 000406              BR     9$
5215                          ;
5216                          ;USED FOR ERROR LOOP IF BIT 3 IN ERROR REG
5217                          ;DOES NOT SET
5218 017720 012706 000376      60$:  MOV    #376,SP       ;SETUP THE SP
5219 017724 012737 017734 000004  MOV    #9$,@#ERRVEC   ;SETUP ERROR VECTOR
5220 017732 005016              CLR    (SP)          ;CAUSE TRAP
5221 017734 012767 017720 161146  9$:  MOV    #60$, $LPERR   ;SETUP ERROR LOOP
5222 017742 012706 001100      MOV    #STACK,SP     ;RESTORE THE SP
5223 017746 022737 000010 177766  CMP    #BIT3,@#CPUERR ;DID YEL ZONE BIT SET?
5224 017754 001407              BEQ    10$           ;BRANCH IF YES
5225 017756 013767 177766 161170  MOV    @#CPUERR,$REGO ;SAVE FOR TYPEOUT
5226 017764 012767 000010 161170  MOV    #BIT3,$TMP0    ;SAVE EXPECTED VALUE
5227                          ;
5228 017772 104257              ERROR  257            ;YEL ZONE BIT DID NOT SET IN CPUERR
5229 017774 012767 020002 161106 10$:  MOV    #57$, $LPERR   ;SETUP ERROR LOOP
5230 020002 005037 177766      57$:  CLR    @#CPUERR      ;CLEAR YEL ZONE BIT
5231 020006 005737 177766      TST    @#CPUERR      ;DID IT CLEAR?
5232 020012 001401              BEQ    TST37         ;BRANCH IF YES
5233 020014 104260              ERROR  260            ;BIT3 DID NOT CLEAR IN CPUERR
5234                          ;
5235                          ;*****
5236                          ;*TEST 37 ROM FIELD CHECK OF PC MANIPULATOR STATES
5237                          ;*
5238                          ;* THIS TEST EXECUTES THE MACHINE STATES THAT MANIPULATE
5239                          ;* THE PC TO ENSURE THAT THE PCB ROM FIELD OR DRX ROM FIELD
5240                          ;* OR SRX ROM FIELD OF THESE STATES IS FUNCTIONAL.
5241                          ;* THESE STATES ARE S13.00, S45.00, MTP.10, D45.80, D45.90,
5242                          ;* D45.00, AND D45.01.
5243                          ;*****
5243 020016 000004      TST37: SCOPE
5244 020020 012767 020050 161060  MOV    #18$, $LPADR   ;SETUP LOOP ADR
5245 020026 032767 000020 161134  BIT    #BIT4,$TMP3    ;WAS T BIT ON?
5246 020034 001405              BEQ    18$           ;BRANCH IF NO
5247 020036 012746 000360      MOV    #360,-(SP)    ;PUT NEW PSW ON STACK
5248 020042 012746 020050      MOV    #18$,-(SP)    ;PUT RETURN ADDR ON STACK
5249 020046 000006              RTT                  ;TURN T BIT ON
5250                          ;BIN*SM1*SF7*DM0(S13.00)
5251 020050 012767 020056 161032 18$:  MOV    #64$, $LPERR   ;SETUP ERROR LOOP
5252 020056 011700      64$:  MOV    (PC),R0       ;EXECUTE INSTRUCTION UNDER TEST
5253 020060 020027 020027      CMP    R0,#20027     ;DID TEST WORK
5254 020064 001402              BEQ    1$            ;BRANCH IF YES
5255 020066 104377              ERROR  377            ;STATE S13.00 FAILED
5256 020070 000443              443
```

```

5257      :BIN*SM4*SF7*DM2*DF7 (S45.00)
5258 020072 012767 020100 161010 1$:  MOV      #63$, $LPERR      ; SETUP ERROR LOOP
5259 020100 012767 024727 000000 63$:  MOV      #24727, 2$      ; PUT INSTRUCTION IN 2$
5260 020106 024727                2$:  CMP      -(PC), (PC)+      ; EXECUTE INSTRUCTION UNDER TEST
5261 020110 000240                NOP                                ; USED TO BE SAFE
5262 020112 001402                BEQ      3$                    ; BRANCH IF TEST OK
5263 020114 104377                ERROR   377                  ; STATE S45.00 FAILED
5264 020116 000444                444
5265      :MTP*DM2*DF7 (MTP.10)
5266 020120 012767 020126 160762 3$:  MOV      #62$, $LPERR      ; SETUP ERROR LOOP
5267 020126 012706 001100 62$:  MOV      #STACK, SP      ; INITIALIZE THE SP
5268 020132 012746 177777        MOV      #-1, -(SP)        ; SET 1076 TO ALL ONES
5269 020136 005067 000002        CLR      4$                    ; ENSURE 4$ CLEAR
5270 020142 006627                MTP1    (PC)+              ; EXECUTE INSTRUCTION UNDER TEST
5271 020144 000000                4$:  .WORD   0                    ; DID INSTRUCTION WORK?
5272 020146 022767 177777 177770  CMP      #-1, 4$          ; BRANCH IF YES
5273 020154 001402                BEQ      7$                    ; STATE MTP.10 FAILED
5274 020156 104377                ERROR   377
5275 020160 000445                445
5276      :BIN*SM2*SF7*DM4*DF7 (D45.80)
5277 020162 012767 020170 160720 7$:  MOV      #61$, $LPERR      ; SETUP ERROR LOOP
5278 020170 012747                61$:  MOV      (PC)+, -(PC)      ; EXECUTE INSTRUCTION UNDER TEST
5279 020172 000402                BR      10$                   ; WILL EXECUTE THIS IF INSTR. WORKS
5280 020174 104377                ERROR   377                  ; STATE D45.80 FAILED
5281 020176 000447                447
5282      :BIN*SM1*SFO*DM4*DF7 (D45.90)
5283 020200 012767 020206 160702 10$:  MOV      #60$, $LPERR      ; SETUP ERROR LOOP
5284 020206 012767 141047 000024 60$:  MOV      #141047, 11$      ; SETUP INSTRUCTION TO EXECUTE
5285 020214 012700 001162        MOV      #STMP0, R0        ; PUT ADDRESS OF STMP0 IN R0
5286 020220 005200                INC      R0
5287 020222 112710 000002        MOV     #BIT1, (R0)        ; SET BIT1 IN STMP1 HIGH BYTE
5288 020226 012705 001166        MOV      #STMP2, R5        ; SETUP R5
5289 020232 012767 001000 160724 11$:  MOV      #BIT9, STMP1      ; SETUP STMP1
5290 020240 141047                BIC     (R0), -(PC)        ; EXECUTE INSTRUCTION UNDER TEST
5291 020242 005767 160716        TST     STMP1              ; DID STMP1 GET CLEARED?
5292 020246 001402                BEQ     12$                   ; BRANCH IF YES
5293 020250 104377                ERROR   377                  ; STATE D45.00 FAILED
5294 020252 000450                450
5295      :DAC*DM4*DF7 (D45.00)
5296 020254 012767 020262 160626 12$:  MOV      #57$, $LPERR      ; SETUP ERROR LOOP
5297 020262 012767 140047 000016 57$:  MOV      #140047, 13$      ; SETUP INSTRUCTION TO EXECUTE
5298 020270 012700 000002        MOV      #BIT1, R0        ; SETUP R0 TO CHANGE DR FROM 7 TO 5
5299 020274 012705 001165        MOV      #STMP1+1, R5     ; PUT ADDRESS OF STMP1 HIGH BYTE IN R5
5300 020300 012767 000002 160656 13$:  MOV      #BIT1, STMP1      ; SET UP STMP1 SO INSTRUCTION CLEARS IT
5301 020306 140047                BIC     R0, -(PC)          ; EXECUTE INSTRUCTION UNDER TEST
5302 020310 005767 160650        TST     STMP1              ; DID STMP1 CLEAR?
5303 020314 001402                BEQ     14$                   ; BRANCH IF YES
5304 020316 104377                ERROR   377                  ; STATE D45.90 FAILED
5305 020320 000451                451
5306      :DAC*DM5*DF7 (D45.01)
5307 020322 012767 020330 160560 14$:  MOV      #56$, $LPERR      ; SETUP ERROR LOOP
5308 020330 012767 130057 000052 56$:  MOV      #130057, 15$      ; SETUP INSTRUCTION TO EXECUTE
5309 020336 032737 000020 177776  BIT      #BIT4, @#PSW      ; IS T BIT ON?
5310 020344 001404                BEQ     20$                   ; BRANCH IF NO
5311 020346 012737 000360 000066  MOV      #360, @#TPVEC+2   ; SETUP TP VEC PSW
5312 020354 000403                BR      17$
    
```

```

5313 020356 012737 000340 000066 20$: MOV #PR7,@#TPVEC+2
5314 020364 012737 020424 000064 17$: MOV #16$,@#TPVEC ;SETUP BR4 INTERRUPT VECTOR
5315 020372 000233 SPL 3 ;SET PROCESSOR PRIORITY BELOW BR4
5316 020374 152777 000100 160540 BISB #BIT6,@$TPS ;SET INTERRUPT BIT
5317 020402 012777 000015 160534 MOV #15,@$TPB ;SEND CHARACTER TO PRINTER
5318 020410 130057 000100 160522 15$: BITB R0,@-(PC) ;EXECUTE INSTRUCTION UNDER TEST
5319 020412 142777 000100 160522 BICB #BIT6,@$TPS ;WILL EXECUTE IF STATE FAILS
5320 020420 104377 ERROR 377 ;STATE D45.01 FAILED
5321 020422 000452 452
5322 020424 142777 000100 160510 16$: BICB #BIT6,@$TPS ;TEST OK, CLEAR INTERRUPT BIT
5323 ;CONTINUE
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368

```

```

*****
*TEST 40 RED ZONE TRAP
*****

```

```

*
* A RED ZONE TRAP IS FIRST ATTEMPTED WITH THE SP AT 336.
* IF BEN13 FAILS EXECUTION WILL GO TO EITHER BRK.80 OR BRK.20
* OR PUP.00.
* BRK.80 WILL CAUSE THE OLD PSW AND PC TO BE STACKED ON THE
* OLD STACK INSTEAD OF LOCATIONS 2 AND 0.
* BRK.20 WILL MAKE IT LOOK LIKE THE RED ZONE FAILED.
* PUP.00 WILL CAUSE A TRAP TO LOCATION 24.
*
* IF THE PROCESSOR FAILS TO TRAP EITHER TMCD SL RED IS
* NOT GOING LOW OR TMCC ABORT IS NOT GOING LOW.
*
* IF UBCB ABORT RESTART FAILS TO GO LOW OR E10(13)
* IS BAD THE PROCESSOR WILL HANG IN THE PAUSE STATE.

```

```

*****
TST40: SCOPE
*****

```

```

5342 020432 000004
5343 020434 012767 021166 160626 MOV #TST41,NEXTTST ;SAVE ADDRESS OF NEXT TEST
5344 020442 012767 020500 160436 MOV #14$, $LPADR ;SETUP LOOP ADR
5345 020450 013767 177776 160512 MOV @#PSW,$TMP3 ;SAVE PSW
5346 020456 032737 000020 177776 BIT #BIT4,@#PSW ;IS T BIT ON?
5347 020464 001405 BEQ 14$ ;BRANCH IF NO
5348 020466 012746 000340 MOV #PR7,-(SP) ;PUT NEW PSW ON STACK
5349 020472 012746 020500 MOV #14$,-(SP) ;PUT RETURN ADR ON STACK
5350 020476 000006 RTT ;TURN T BIT OFF
5351 020500 012737 000340 000006 14$: MOV #PR7,@#ERRVEC+2 ;RESTORE ERRVEC PSW
5352 020506 012767 020522 160374 MOV #64$, $LPERR ;SETUP ERROR LOOP
5353 020514 012737 020560 000024 MOV #3$,@#PWRVEC ;SETUP LOCATION 24
5354 020522 012737 020576 000004 64$: MOV #1$,@#ERRVEC ;SETUP ERRVEC
5355 020530 012706 000336 MOV #336,SP ;SET THE SP TO RED ZONE
5356 020534 012700 177777 MOV #-1,R0 ;SETUP R0
5357 020540 010016 MOV R0,(SP) ;EXECUTE THE TRAP INSTRUCTION
5358 020542 012706 001100 000004 6$: MOV #STACK,SP ;RESET THE SP
5359 020546 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5360 020554 104235 ERROR 235 ;RED ZONE REFERENCE FAILED TO TRAP
5361 020556 000431 BR 8$
5362 020560 012706 001100 000004 3$: MOV #STACK,SP ;RESET THE SP
5363 020564 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4
5364 020572 104237 ERROR 237 ;BEN 13 FAILED TO PUP.00
5365 020574 000422 BR 8$
5366 020576 023727 000000 020542 1$: CMP @#0,#6$ ;DID BEN 13 FAIL?
5367 020604 001407 BEQ 7$ ;BRANCH IF NO
5368 020606 012706 001100 MOV #STACK,SP ;RESET THE SP

```

```

CEKBEO 11/70-74MP CPU #2          MACY11 30A(1052) 10-MAR-80 09:32  L 8  PAGE 103
CEKBBE .P11 10-MAR-80 09:31      T40     RED ZONE TRAP                                SEQ 0102

5369 020612 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5370 020620 104240                ERROR    240                ;BEN 13 FAILED TO BRK.80
5371 020622 000407                BR       8$
5372 020624 022737 177777 000336 7$:    CMP      #-1,@#336        ;;DID YEL ZONE OCCUR?
5373 020632 001003                BNE     8$                ;BRANCH IF NO
5374 020634 005037 000336        CLR     @#336             ;SETUP FOR LOOPING
5375 020640 104251                ERROR    251             ;YEL ZONE IN RED REGION
5376
5377                ;TEST TO ENSURE PSW REFERENCE VIA THE SP CAUSES A RED ZONE TRAP
5378 020642 012767 020650 160240 8$:    MOV      #63$, $LPERR    ;SETUP ERROR LOOP
5379 020650 012737 020700 000004 63$:   MOV      #4$, @#ERRVEC   ;SETUP ERRVEC
5380 020656 012706 177776        MOV     #PSW, SP        ;PUT ADDRESS OF PSW IN SP
5381 020662 005016                CLR     (SP)            ;EXECUTE THE TRAP CAUSING INSTRUCTION
5382 020664 012706 001076        MOV     #1076, SP       ;RESET THE SP
5383 020670 012737 032666 000004        MOV     #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5384 020676 104241                ERROR    241             ;NO RED ZONE ON STACK OVERFLOW
5385
5386                ;TEST TO ENSURE SL REG GREATER THAN BADRR CAUSES A RED ZONE
5387 020700 012767 020706 160202 4$:    MOV      #62$, $LPERR    ;SETUP ERROR LOOP
5388 020706 012737 020750 000004 62$:   MOV      #5$, @#ERRVEC   ;SETUP RESVEC
5389 020714 012737 000400 177774        MOV     #400,@#STKLMT   ;SET STACK LIMIT REGISTER TO 400
5390 020722 012706 000336        MOV     #336, SP        ;SET THE SP
5391 020726 005016                CLR     (SP)            ;EXECUTE THE TRAP CAUSING INSTRUCTION
5392 020730 012706 001100        MOV     #STACK, SP     ;RESET THE SP
5393 020734 005037 177774        CLR     @#STKLMT       ;ENSURE SL REG. CLEAR
5394 020740 012737 032666 000004        MOV     #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5395 020746 104242                ERROR    242             ;NO RED ZONE WHEN SL REG>BUS ADDR
5396
5397                ;TEST OF TMCD YEL ZONE (E15)
5398 020750 012767 020762 160132 5$:    MOV      #61$, $LPERR    ;SETUP ERROR LOOP
5399 020756 005037 177774        CLR     @#STKLMT       ;ENSURE SL CLEAR
5400 020762 012737 021002 000004 61$:   MOV      #9$, @#ERRVEC   ;SETUP ERRVEC
5401 020770 012706 000240        MOV     #240, SP       ;SETUP THE SP
5402 020774 012700 177777        MOV     #-1, RO        ;SETUP RO
5403 021000 010016                MOV     RO, (SP)       ;EXECUTE THE TRAP CAUSING INSTRUCTION
5404 021002 022737 177777 000240 9$:    CMP      #-1,@#240      ;DID YEL ZONE OCCUR?
5405 021010 001010                BNE     10$             ;BRANCH IF NO
5406 021012 012706 001100        MOV     #STACK, SP     ;RESTORE THE SP
5407 021016 012737 032666 000004        MOV     #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5408 021024 005037 000240        CLR     @#240          ;FOR LOOPING
5409 021030 104252                ERROR    252             ;TMCD YEL ZONE DID NOT GO LOW
5410 021032 012767 021040 160050 10$:   MOV      #60$, $LPERR    ;SETUP ERROR LOOP
5411 021040 012737 021054 000004 60$:   MOV      #11$, @#ERRVEC  ;SETUP ERRVEC
5412 021046 012706 000140        MOV     #140, SP       ;SETUP THE SP
5413 021052 010016                MOV     RO, (SP)       ;EXECUTE THE TRAP CAUSING INSTR.
5414 021054 022737 177777 000140 11$:   CMP      #-1,@#140      ;DID YEL ZONE OCCUR?
5415 021062 001010                BNE     12$             ;BRANCH IF NO
5416 021064 012706 001100        MOV     #STACK, SP     ;RESTORE SP
5417 021070 012737 032666 000004        MOV     #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5418 021076 005037 000140        CLR     @#140          ;FOR LOOPING
5419 021102 104253                ERROR    253             ;TMCD YEL ZONE DID NOT GO LOW
5420 021104 012706 001100 12$:   MOV      #STACK, SP     ;RESTORE SP
5421 021110 012737 032666 000004        MOV     #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5422 021116 022737 000004 177766        CMP     #BIT2,@#CPUERR  ;DID RED ZONE BIT IN CPU ERROR SET?
5423 021124 001407                BEQ     13$             ;BRANCH IF YES
5424 021126 013767 177766 160020        MOV     @#CPUERR,$REGO ;SAVE FOR TYPEOUT

```



```

5425 021134 012767 000004 160020      MOV    #BIT2,$TMP0      ;SAVE EXPECTED VALUE
5426 021142 104261                ERROR  261             ;RED ZONE BIT IN CPU ERROR DID NOT SET
5427 021144 012767 021152 157736 13$:  MOV    #57$,$LPERR     ;SETUP ERROR LOOP
5428 021152 005037 177766 57$:  CLR    @#CPUERR        ;CLEAR RED ZONE BIT
5429 021156 005737 177766      TST    @#CPUERR        ;DID REG CLEAR?
5430 021162 001401                BEQ    TST41           ;:BRANCH IF YES
5431 021164 104262                ERROR  262             ;RED ZONE BIT DID NOT CLEAR
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445 021166 000004                TST41: SCOPE
5446 021170 012767 021432 157776      MOV    #TST42,$ESCAPE  ;SAVE START ADDRESS OF NEXT TEST
5447 021176 012767 021432 160064      MOV    #TST42,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
5448 021204 012737 036300 000024      MOV    #SPWRDN,@#PWVREC ;RESTORE POWER VECTOR
5449 021212 005005                CLR    R5              ;CLEAR R5
5450 021214 012737 125252 177774      MOV    #125252,@#STKLMT ;PUT PATTERN IN STACK LIMIT REG
5451 021222 012737 000200 177770      MOV    #200,@#177770   ;PUT 200 IN PB REGISTER
5452 021230 012700 125000                MOV    #125000,R0      ;SETUP R0 TO LOOK LIKE STK LIMIT
5453 021234 000237                SPL    7                ;PUT PRIORITY BITS IN KNOWN CONFIGURATION
5454 021236 020037 177774      CMP    R0,@#STKLMT     ;EXECUTE TEST ON STKLMT REG.
5455 021242 001404                BEQ    1$              ;BRANCH IF TEST OK
5456 021244 005205                INC    R5              ;INCREMENT TEST FAIL INDICATOR
5457 021246 013767 177774 157744      MOV    @#STKLMT,E1STKLM ;SAVE ERROR VALUE
5458 021254 012737 052400 177774 1$:  MOV    #52400,@#STKLMT ;PUT COMPLEMENT PATTERN REG.
5459 021262 012700 052400                MOV    #52400,R0      ;PUT IN R0
5460 021266 020037 177774      CMP    R0,@#STKLMT     ;EXECUTE TEST ON REG.
5461 021272 001415                BEQ    2$              ;BRANCH IF TEST OK
5462 021274 005705                TST    R5              ;DID FIRST TEST FAIL?
5463 021276 001023                BNE    3$              ;BRANCH IF YES
5464 021300 013767 177774 157714      MOV    @#STKLMT,E2STKLM ;SAVE ERROR VALUE
5465 021306 012767 052400 157646      MOV    #52400,$TMP0    ;SAVE EXPECTED VALUE
5466 021314 005037 177774      CLR    @#STKLMT        ;CLEAR THE REG
5467 021320 012706 001100                MOV    #STACK,SP      ;RESTORE THE SP
5468 021324 104243                ERROR  243             ;52400 PATTERN FAILED
5469 021326 005705 2$:  TST    R5              ;DID FIRST TEST FAIL?
5470 021330 001436                BEQ    4$              ;BRANCH IF NO
5471 021332 012767 125000 157622      MOV    #125000,$TMP0   ;SAVE EXPECTED VALUE
5472 021340 005037 177774      CLR    @#STKLMT        ;CLEAR REG.
5473 021344 104244                ERROR  244             ;125252 PATTERN FAILED
5474 021346 005037 177774 3$:  CLR    @#STKLMT        ;CLEAR STACK LIMIT REG
5475 021352 026727 157642 013767      CMP    E1STKLM,#13767  ;DID BR GET SELECTED ON STACK LIMIT REF.?
5476 021360 001001                BNE    5$              ;BRANCH IF NO
5477 021362 104245                ERROR  245             ;BR SELECTED BY DMUX
5478 021364 026727 157630 125200 5$:  CMP    E1STKLM,#125200 ;DID PB GET GATED ALSO?
5479 021372 001001                BNE    6$              ;BRANCH IF NO
5480 021374 104246                ERROR  246             ;TMCD LO BYTE EN DOES NOT GO LOW
    
```



5489  
5490  
5491  
5492  
5493  
5494  
5495  
5496  
5497  
5498  
5499  
5500  
5501  
5502  
5503  
5504  
5505  
5506  
5507  
5508  
5509  
5510  
5511  
5512  
5513  
5514  
5515  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5525  
5526  
5527  
5528  
5529  
5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5539  
5540  
5541  
5542  
5543  
5544

021432 000004

```
*****  
*TEST 42 SL REGISTER COMPARATOR TEST 1  
*  
* THIS TEST RUNS A HIGH BYTE COUNT PATTERN THRU THE BUS  
* ADDRESS MUX FOR EACH PATTERN OF THE STACK LIMIT REGISTER.  
* FOR EACH PATTERN OF ADDRESSES THERE WILL BE ONE YEL TRAP  
* AT THE ADDRESS CORRESPONDING TO THE SL REG+340 AND A RED  
* TRAP AT EVERY ADDRESS BELOW THIS.  
* THIS TEST ONLY TESTS ADDRESSES UP TO THE I/O PAGE.  
* THE I/O PAGE ADDRESSES WILL BE TESTED SEPARATELY WITH  
* MEMORY MANAGEMENT ENABLED AND THE I/O PAGE MAPPED INTO RESIDENT  
* MEMORY  
*  
* THE FOLLOWING ARE THE TYPES OF ERRORS THAT CAN OCCUR IN THIS TEST:  
* TYPE DESCRIPTION  
* 0 RED ZONE TRAP ON YELLOW ZONE ADDRESS  
* 2 RED ZONE TRAP ON LEGAL ADDRESS  
* 4 YELLOW ZONE TRAP ON RED ZONE ADDRESS  
* 6 YELLOW ZONE TRAP ON LEGAL ADDRESS  
* 10 NO TRAP ON RED ZONE ADDRESS  
* 12 NO TRAP ON YELLOW ZONE ADDRESS  
*  
* THE LOW BYTE ADDRESS IN THE STACK POINTER IS ALWAYS  
* 340 AND WILL NOT BE TYPED ON AN ERROR.  
*****  
TST42: SCOPE  
*****  
*NOTE: IF THE LOOP ON ERROR SWITCH IS UP (SWITCH 9) THE TEST WILL  
* LOOP ON THE FIRST ERROR WITH NO ERROR TYPEOUT. OTHERWISE ALL  
* ERRORS WILL BE RECORDED IN A TABLE AND TYPED OUT AT THE  
* END OF THE TEST.  
* IF SWITCH 3 (DISABLE MEMORY MANAGEMENT TESTS) IS NOT ON,  
* THIS TEST IS SKIPPED AND TEST 70 WILL EXECUTE.  
*****  
MOV #TST43,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST  
MOV #TST43,NEXTTST ;SAVE START ADDRESS OF NEXT TEST  
MOV #^D1990,$ICNT ;SETUP ITERATION COUNT  
MOV #13$, $LPADR ;SETUP LOOP ADDRESS  
13$: BIT #BIT3,@#SWR ;IS SWITCH 3 ON?  
BNE 11$ ;BRANCH IF YES  
JMP @ $ESCAPE ;GO TO NEXT TEST  
11$: MOV #1$,@#ERRVEC ;SETUP ERRVEC  
CLR $TMP2 ;INITIALIZE ERROR OVERFLOW FLAG  
MOV #340,R3 ;SET SOB COUNT FOR SL REGISTER  
MOV #120000,R0 ;INITIALIZE ERROR DATA POINTER  
CLR 2(R0) ;INITIALIZE ERROR DATA BUFFER  
MOV #340,R1 ;INITIALIZE YELLOW ZONE ADDRESS(TRAP CASE)  
MOV #344,R4 ;SETUP YELLOW ZONE ADDR(NO TRAP)  
2$: MOV #340,R2 ;SET SOB COUNT FOR SP  
MOV #344,R5 ;INITIALIZE SECONDARY STORAGE FOR SP  
3$: MOV -2(R5),$TMP0 ;SAVE WORDS AT  
MOV -4(R5),$TMP1 ;STACK UNDER TEST  
4$: MOV R5,R6 ;SET THE SP  
MOV (R6),(R6) ;EXECUTE TEST INSTRUCTION  
;NO TRAP. DETERMINE IF THIS IS CORRECT.
```

```
5545 021566 020604          CMP      R6,R4          ;IS ADDRESS > YELL ZONE BOUNDRY?
5546 021570 101066          BHI      5$             ;BRANCH IF YES
5547 021572 001403          BEQ      6$             ;BRANCH IF ADDRESS = YELL ZONE BOUNDRY
5548                               ;NO TRAP ADDRESS IS LESS THAN YELLOW ZONE BOUNDRY
5549 021574 052710 000010      BIS      #BIT3,(R0)     ;SET ERROR TYPE IN DATA BUFFER
5550 021600 000442          BR       10$           ;GO RECORD DATA
5551                               ;NO TRAP EQUALS YELLOW ZONE BOUNDRY
5552 021602 052710 000012      6$:    BIS      #12,(R0) ;SET ERROR TYPE IN DATA BUFFER
5553 021606 000437          BR       10$           ;GO RECORD DATA
5554                               ;
5555                               ;GOT A TRAP. RESTORE AND DETERMINE IF IT IS CORRECT.
5556 021610 016765 157346 177776 1$:    MOV      $TMP0,-2(R5) ;RESTORE WORDS AT
5557 021616 016765 157342 177774      MOV      $TMP1,-4(R5) ;OLD STACK UNDER TEST
5558 021624 032737 000004 177766      BIT      #BIT2,@#CPUERR ;WAS IT A RED ZONE?
5559 021632 001406          BEQ      8$             ;BRANCH IF NO
5560 021634 020106          CMP      R1,R6         ;IS ADDRESS < YELL ZONE BOUNDRY?
5561 021636 101043          BHI      5$             ;BRANCH IF YES
5562 021640 001422          BEQ      10$           ;BRANCH IF ADDRESS = YELL ZONE BOUNDRY
5563                               ;RED ZONE TRAP ON LEGAL ADDRESS
5564 021642 052710 000002      BIS      #BIT1,(R0)     ;SET ERROR TYPE IN DATA BUFFER
5565 021646 000417          BR       10$           ;GO RECORD DATA
5566                               ;NOT A RED ZONE. IS IT A YELLOW ZONE?
5567 021650 032737 000010 177766 8$:    BIT      #BIT3,@#CPUERR ;IS THIS A YELLOW ZONE TRAP?
5568 021656 001002          BNE      12$           ;BRANCH IF NO
5569 021660 000167 011002      JMP      CPUSPUR        ;GO TO SPURIOUS ROUTINE
5570 021664 020106          12$:    CMP      R1,R6         ;IS ADDRESS = YELL ZONE BOUNDRY?
5571 021666 001427          BEQ      5$             ;BRANCH IF YES
5572 021670 101003          BHI      9$             ;BRANCH IF ADDRESS IS < YELL ZONE BOUNDRY
5573                               ;YELLOW ZONE TRAP ON LEGAL ADDRESS
5574 021672 052710 000006      BIS      #6,(R0)        ;SET ERROR TYPE IN DATA BUFFER
5575 021676 000403          BR       10$           ;GO RECORD DATA
5576                               ;YELLOW ZONE TRAP ON RED ZONE ADDRESS
5577 021700 052710 000004      9$:    BIS      #BIT2,(R0) ;SET ERROR TYPE IN DATA BUFFER
5578 021704 000400          BR       10$           ;GO RECORD DATA
5579                               ;
5580                               ;RECORD ERROR DATA
5581 021706 032737 001000 177570 10$:   BIT      #BIT9,@#SWR    ;IS LOOP ON ERROR ENABLED?
5582 021714 001322          BNE      4$             ;BRANCH IF YES
5583 021716 005767 157244      TST      $TMP2          ;HAS ERROR BUFFER OVERFLOWED?
5584 021722 001011          BNE      5$             ;BRANCH IF YES
5585 021724 005200          INC      R0             ;SET POINTER TO HIGH BYTE
5586 021726 113720 177775      MOVB    @#STKLM+1,(R0)+ ;SAVE ERROR STACK LIMIT
5587 021732 010620          MOV      R6,(R0)+      ;SAVE ERROR SP
5588 021734 020027 157774      CMP      R0,#157774    ;IS BUFFER AT PAGE 7?
5589 021740 001002          BNE      5$             ;BRANCH IF NO
5590 021742 005267 157220      INC      $TMP2          ;SET BUFFER OVERFLOW FLAG
5591                               ;
5592 021746 062705 000400      5$:    ADD      #400,R5        ;GO TO NEXT STACK ADDRESS
5593 021752 005037 177766      CLR      @#CPUERR       ;CLEAR ERROR REG
5594 021756 000240          NOP                     ;
5595 021760 005302          DEC      R2             ;REPLACES A
5596 021762 001271          BNE      3$             ;SOB INSTRUCTION
5597 021764 062737 000400 177774      ADD      #400,@#STKLMT ;GO TO NEXT SL ADDRESS
5598 021772 062701 000400      ADD      #400,R1        ;SET NEXT YELLOW ZONE ADDRESS
5599 021776 000240          NOP                     ;
5600 022000 062704 000400      ADD      #400,R4        ;SET NEXT YELLOW ZONE ADDR(NO TRAP)
```

```
5601 022004 005303          DEC    R3          ;THIS REPLACES
5602 022006 001253          BNE    2$          ;A SOB
5603
5604          ;DONE WITH TEST.  WAS THERE AN ERROR?
5605 022010 005037 177774    CLR    @#STKLMT    ;RESET THE SL REG
5606 022014 012706 001100    MOV    #STACK,SP  ;AND SP
5607 022020 012737 032666 000004  MOV    #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5608 022026 005737 120002    TST    @#120002   ;WAS THERE AN ERROR?
5609 022032 001403          BEQ    TST43       ;:BRANCH IF NO
5610 022034 010067 157114    MOV    R0,$REGO   ;SAVE ERROR DATA POINTER
5611 022040 104263          ERROR  26$        ;STACK LIMIT COMPARATORS FAILED
5612
5613          ;*****
5614          ;*TEST 43          ODD ADDRESS ERROR
5615          ;*
5616          ;*      BEN 13 SHOULD NOT FAIL.
5617          ;*      IF THE PROCESSOR FAILS TO TRAP IN ALL SECTIONS EITHER TMCC ODD
5618          ;*      ADRS ERR IS NOT GOING LOW OR TMCC BUS ERROR IS NOT GOING LOW.
5619          ;*
5620          ;*      EACH TYPE OF ODD ADDRESS ERROR IS TESTED INDIVIDUALLY TO
5621          ;*      ALLOW MAXIMUM ISOLATION.
5622          ;*      NOTE:AN ODD ADDRESS ON 'KERNEL DATI' CANNOT BE TESTED.
5623          ;*      THIS SIGNAL COMES UP WHEN A TRAP VECTOR IS READ IN FROM THE BUS.
5624          ;*      *****
5624 022042 000004          TST43: SCOPE
5625 022044 012767 022476 157216  MOV    #TST44,NEXTTST ;SAVE ADDRESS OF NEXT TEST
5626 022052 005005          CLR    R5          ;INITIALIZE ERROR COUNT
5627          ;NON BYTE REFERENCE ON DATIP
5628 022054 012706 001100    MOV    #STACK,SP  ;INITIALIZE THE SP
5629 022060 012737 022140 000004  MOV    #1$,@#ERRVEC ;SETUP ERRVEC
5630 022066 012700 001163    MOV    #STMP0+1,R0 ;PUT ODD ADDRESS IN R0
5631 022072 012767 177777 157062  MOV    #-1,$TMP0   ;PUT -1 IN $TMP0 TO SEE IF TEST CHANGES IT
5632 022100 005210          INC    (R0)        ;EXECUTE ODD ADDRESS INSTRUCTION
5633          ;TRAP DID NOT OCCUR. TRY A DATI.
5634 022102 012737 022126 000004  MOV    #2$,@#ERRVEC ;SETUP ERRVEC
5635 022110 110030          MOVB  R0,@(R0)+   ;EXECUTE DATI TO CAUSE ODD ADDR
5636 022112 012737 032666 000004  MOV    #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5637 022120 005205          INC    R5          ;SET ERROR COUNT
5638 022122 104265          ERROR  26$        ;NEITHER - BYIN OR DATI CAUSE ODD ADDR
5639 022124 000423          BR    3$
5640 022126 012737 032666 000004 2$:  MOV    #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5641 022134 104266          ERROR  26$        ;DATI TRAPPED BUT -BYIN DIDN'T
5642 022136 000416          BR    3$
5643
5644          ;NON BYTE REFERENCE WORKED. NOW TRY DATI.
5645 022140 005037 177766    1$:  CLR    @#CPUERR    ;CLEAR ERROR REG
5646 022144 012706 001100    MOV    #STACK,SP  ;INITIALIZE THE SP
5647 022150 012767 022140 156732  MOV    #1$, $LPERR ;CHANGE LPERR ADDRESS TO THIS SECTION
5648 022156 012737 022174 000004  MOV    #3$,@#ERRVEC ;SETUP ERRVEC
5649 022164 012700 001163    MOV    #STMP0+1,R0 ;PUT ODD ADDR IN R0
5650 022170 110030          MOVB  R0,@(R0)+   ;EXECUTE ODD ADDRESS INSTRUCTION
5651 022172 000404          BR    13$
5652 022174 032737 000100 177766 3$:  BIT    #BIT6,@#CPUERR ;DID ODD ADDR BIT SET?
5653 022202 001037          BNE    12$        ;BRANCH IF YES
5654          ;TRAP FAILED OR ERROR REG FAILED. TRY A DATO. (REVERSES INPUTS TO TMCC E5(12,13))
5655 022204 005037 177766    13$: CLR    @#CPUERR    ;CLEAR ODD ADDR BIT
5656 022210 012706 001077    MOV    #STACK-1,SP ;MAKE SP AN ODD ADDRESS
```

```

CEKBEO 11/70-74MP CPU #2          MACY11 30A(1052) 10-MAR-80 09:32 E 9 PAGE 109
CEKBBE.P11 10-MAR-80 09:31        T43      ODD ADDRESS ERROR                               SEQ 0108

5657 022214 012737 022250 000004      MOV      #4$,@#ERRVEC      ;SETUP ERRVEC
5658 022222 012737 022232 000030      MOV      #5$,@#EMTVEC     ;SETUP EMT VECTOR
5659 022230 104000                    EMT      0                  ;EXECUTE DATO TO SP
5660 022232 012706 001100                    5$:     MOV      #STACK,SP      ;RESTORE SP
5661 022236 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5662 022244 104267                    ERROR    267                ;BOTH DATI AND DATO FAILED
5663 022246 000446                    BR       6$
5664 022250 012706 001100                    4$:     MOV      #STACK,SP      ;RESTORE THE SP
5665 022254 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5666 022262 032737 000100 177766      BIT      #BIT6,@#CPUERR   ;DID ODD ADDR BIT SET?
5667 022270 001401                    BEQ     14$                ;BRANCH IF NO
5668 022272 104270                    ERROR    270                ;DATO WORKS BUT DATI FAILED
5669 022274 104377                    14$:    ERROR    377
5670 022276 000456                    456
5671 022300 000431                    BR       6$
5672
5673      ;EITHER DATI WORKED OR -BYIN AND DATI FAILED. NOW TRY DATO.
5674 022302 012767 022174 156600                    12$:    MOV      #3$, $LPERR      ;CHANGE LPERR ADDRESS TO THIS SECTION
5675 022310 012706 001077                    MOV      #STACK-1,SP      ;MAKE SP AN ODD ADDRESS
5676 022314 012737 022364 000004      MOV      #6$,@#ERRVEC     ;SETUP ERRVEC
5677 022322 012737 022332 000030      MOV      #7$,@#EMTVEC     ;SETUP EMTVEC TO CATCH A FAILURE
5678 022330 104000                    EMT      0                  ;EXECUTE DATO TO ODD ADDRESS
5679      ;TRAP FAILED
5680 022332 012706 001076                    7$:     MOV      #1076,SP          ;RESET SP
5681 022336 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5682 022344 012737 033550 000030      MOV      # $ERROR,@#EMTVEC ;RESTORE EMTVEC
5683 022352 005705                    TST     R5
5684 022354 001002                    BNE     11$
5685 022356 104271                    ERROR    271                ;NO TRAP ON DATO BUT DATI & -BYIN OK
5686 022360 000401                    BR       6$
5687 022362 104274                    11$:    ERROR    274                ;NO TRAPS
5688
5689      ;EITHER DATO WORKED OR -BYIN AND DATI AND DATO FAILED OR DATO
5690      ;FAILED BUT -BYIN AND DATI OK. TRY SM357 AND SRC1 DATI.
5691 022364 012737 033550 000030                    6$:     MOV      # $ERROR,@#EMTVEC ;RESTORE EMT VEC
5692 022372 012767 022400 156510                    MOV      #8$, $LPERR      ;CHANGE LOOP ERROR ADDR TO THIS SECT.
5693 022400 012706 001076                    8$:     MOV      #1076,SP          ;RESET SP
5694 022404 012737 022452 000004      MOV      #9$,@#ERRVEC     ;SETUP ERRVEC
5695 022412 012700 001163                    MOV      #STMP0+1,R0      ;PUT ODD ADDRESS IN R0
5696 022416 012767 001164 156536      MOV      #STMP1,STMP0     ;PUT EVEN ADDRESS IN STMP0
5697 022424 113001                    MOV     @ (R0)+,R1        ;EXECUTE INSTRUCTION TO CAUSE TRAP
5698 022426 012737 032666 000004      MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5699 022434 012767 022476 156532      MOV      #TST44,$ESCAPE   ;SAVE START ADDRESS OF NEXT TEST
5700 022442 012767 022476 156620      MOV      #TST44,NEXTTST   ;SAVE START ADDRESS OF NEXT TEST
5701 022450 104272                    ERROR    272                ;SM357*SRC1 DATI FAILED TO TRAP
5702
5703      ;SM357*SRC1 DATI WORKS CHECK CPU ERROR REG.
5704 022452 012737 032666 000004                    9$:     MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
5705 022460 032737 000100 177766      BIT      #BIT6,@#CPUERR   ;IS ODD ADDRESS BIT SET?
5706 022466 001001                    BNE     10$                ;BRANCH IF YES
5707 022470 104273                    ERROR    273                ;CPU ERROR REG BIT DOES NOT SET
5708 022472 005037 177766                    10$:    CLR      @#CPUERR         ;CLEAR ERROR REG.
5709
5710      ;*****
5711      ;*TEST 44      ILLEGAL INSTRUCTIONS
5712      ;*      THIS TEST ENSURES THAT ILLEGAL OP CODES TRAP TO LOCATION 10.

```

```
5713 :* ONLY THOSE OP CODES THAT HAVE A SINGLE BIT THAT
5714 :* DISTINGUISHES THEM FROM A LEGAL INSTRUCTION WILL BE TESTED.
5715 :*****
5716 022476 000004 TST44: SCOPE
5717 022500 012767 023142 156562 MOV #KBTST,NEXTTST ;SAVE ADDRESS OF NEXT TEST
5718 022506 012767 022542 156372 MOV #31$, $LPADR ;SETUP LOOP ADR
5719 022514 032737 000020 177776 BIT #BIT4,@#PSW ;IS T BIT ON?
5720 022522 001404 BEQ 30$ ;BRANCH IF NO
5721 022524 012737 000360 000012 MOV #360,@#RESVEC+2 ;SETUP RESVEC PSW
5722 022532 000403 BR 31$ ;GO DO TEST
5723 :SECTION 1-10 THRU 77
5724 022534 012737 000340 000012 30$: MOV #PR7,@#RESVEC+2 ;RESTORE RESVEC PSW
5725 022542 012767 022556 156340 31$: MOV #1$, $LPERR ;SETUP ERROR LOOP
5726 022550 012737 022574 000010 MOV #2$,@#RESVEC ;SETUP RESVEC
5727 022556 012706 001100 1$: MOV #STACK,SP ;INITIALIZE THE SP
5728 022562 012746 022570 MOV #3$, -(SP) ;PUT ADDRESS OF 3$ ON STACK
5729 022566 000010 10 ;EXECUTE OP CODE 10
5730 :FAILURE-RTT OCCURED
5731 022570 104377 3$: ERROR 377 ;OP CODE 10 FAILED TO TRAP
5732 022572 000453 453
5733 022574 012767 022610 156306 2$: MOV #4$, $LPERR ;SETUP ERROR LOOP
5734 022602 012737 022616 000010 MOV #5$,@#RESVEC ;SETUP RESVEC
5735 022610 000015 4$: 15 ;EXECUTE OP CODE 15
5736 :FAILURE
5737 022612 104377 ERROR 377 ;OP CODE 15 FAILED TO TRAP
5738 022614 000453 453
5739 022616 012767 022632 156264 5$: MOV #6$, $LPERR ;SETUP ERROR LOOP
5740 022624 012737 022640 000010 MOV #7$,@#RESVEC ;SETUP RESVEC
5741 022632 000025 6$: 25 ;EXECUTE OP CODE 25
5742 :FAILURE
5743 022634 104377 ERROR 377 ;OP CODE 25 FAILED TO TRAP
5744 022636 000453 453
5745 022640 012767 022654 156242 7$: MOV #8$, $LPERR ;SETUP ERROR LOOP
5746 022646 012737 022662 000010 MOV #9$,@#RESVEC ;SETUP RESVEC
5747 022654 000045 8$: 45 ;EXECUTE OP CODE 45
5748 :FAILURE
5749 022656 104377 ERROR 377 ;OP CODE 45 FAILED TO TRAP
5750 022660 000453 453
5751 :
5752 :*****
5753 :SECTION 2-210 THRU 227
5754 022662 012767 022676 156220 9$: MOV #10$, $LPERR ;SETUP ERROR LOOP
5755 022670 012737 022714 000010 MOV #11$,@#RESVEC ;SETUP RESVEC
5756 022676 012706 001100 10$: MOV #STACK,SP ;INITIALIZE THE SP
5757 022702 012700 022710 MOV #12$,R0 ;SETUP R0 INCASE RTS
5758 022706 000210 210 ;EXECUTE OP CODE 210
5759 :FAILURE-RTS EXECUTED
5760 022710 104377 12$: ERROR 377 ;OP CODE 210 FAILED TO TRAP
5761 022712 000453 453
5762 022714 012767 022730 156166 11$: MOV #13$, $LPERR ;SETUP ERROR LOOP
5763 022722 012737 022746 000010 MOV #14$,@#RESVEC ;SETUP RESVEC
5764 022730 012706 001100 13$: MCV #STACK,SP ;INITIALIZE THE SP
5765 022734 012700 022742 MOV #15$,R0 ;SETUP R0 TO CATCH RTS
5766 022740 000220 220 ;EXECUTE OP CODE 220
5767 :FAILURE-RTS EXECUTED
5768 022742 104377 15$: ERROR 377 ;OP CODE 220 FAILED TO TRAP
```

```
5769 022744 000453          453
5770
5771
5772
5773 022746 012767 022762 156134
5774 022754 012737 022770 000010
5775 022762 007000
5776
5777 022764 104377
5778 022766 000453
5779
5780
5781
5782 022770 012767 023004 156112
5783 022776 012737 023012 00001C
5784 023004 075000
5785
5786 023006 104377
5787 023010 000453
5788 023012 012767 023026 156070
5789 023020 012737 023034 000010
5790 023026 076000
5791
5792 023030 104377
5793 023032 000453
5794
5795
5796
5797 023034 012767 023050 156046
5798 023042 012737 023056 000010
5799 023050 106400
5800
5801 023052 104377
5802 023054 000453
5803
5804
5805
5806 023056 012767 023072 156024
5807 023064 012737 023104 000010
5808 023072 012706 001100
5809 023076 106700
5810
5811 023100 104377
5812 023102 000453
5813 023104 012767 023120 155776
5814 023112 012737 023126 000010
5815 023120 107000
5816
5817 023122 104377
5818 023124 000453
5819 023126 012737 000012 000010
5820 023134 012737 000340 000012
5821
5822
5823
5824
```

\*\*\*\*\*  
SECTION 3-7000 THRU 7777  
14\$: MOV #27\$, \$LPERR ; SETUP ERROR LOOP  
MOV #16\$, @#RESVEC ; SETUP RESVEC  
27\$: 7000 ; EXECUTE OP CODE 7000  
:FAILURE  
ERROR 377 ; OP CODE 7000 FAILED TO TRAP  
453  
\*\*\*\*\*  
SECTION 4-75000 THRU 76000  
16\$: MOV #17\$, \$LPERR ; SETUP ERROR LOOP  
MOV #18\$, @#RESVEC ; SETUP RESVEC  
17\$: 75000 ; EXECUTE OP CODE 75000  
:FAILURE  
ERROR 377 ; OP CODE 75000 FAILED TO TRAP  
453  
18\$: MOV #19\$, \$LPERR ; SETUP ERROR LOOP  
MOV #20\$, @#RESVEC ; SETUP RESVEC  
19\$: 76000 ; EXECUTE OP CODE 76000  
:FAILURE  
ERROR 377 ; OP CODE 76000 FAILED TO TRAP  
453  
\*\*\*\*\*  
SECTION 5-106400 THRU 106477  
20\$: MOV #21\$, \$LPERR ; SETUP ERROR LOOP  
MOV #22\$, @#RESVEC ; SETUP RESVEC  
21\$: 106400 ; EXECUTE OP CODE 106400  
:FAILURE  
ERROR 377 ; OP CODE 106400 FAILED TO TRAP  
453  
\*\*\*\*\*  
SECTION 6-106700 THRU 107777  
22\$: MOV #23\$, \$LPERR ; SETUP ERROR LOOP  
MOV #24\$, @#RESVEC ; SETUP RESVEC  
23\$: MOV #STACK, SP ; INITIALIZE THE SP  
106700 ; EXECUTE OP CODE 106700  
:FAILURE  
ERROR 377 ; OP CODE 106700 FAILED TO TRAP  
453  
24\$: MOV #25\$, \$LPERR ; SETUP ERROR LOOP  
MOV #26\$, @#RESVEC ; SETUP RESVEC  
25\$: 107000 ; EXECUTE OP CODE 107000  
:FAILURE  
ERROR 377 ; OP CODE 107000 FAILED TO TRAP  
453  
26\$: MOV #12, @#RESVEC ; RESTORE RESVEC  
MOV #PR7, @#RESVEC+2 ; RESTORE RESVEC PSW  
; CONTINUE  
\*\*\*\*\*



```
5825 ; THIS ROUTINE IS USED TO DETERMINE WHICH CPU WE ARE RUNNING ON FOR THE NEXT
5826 ; TEST. THE ROUTINE EXECUTES AN MFPT INSTRUCTION WHICH ONLY EXISTS ON A KB11-E
5827 ; OR KB11-EM. ON A KB11-B/C OR KB11-CM THE MFPT WILL TRAP. AFTER DETERMINING
5828 ; WHICH CPU IS BEING RUN ON A MESSAGE WILL BE PRINTED. THIS TEST COULD NOT
5829 ; BE RUN BEFORE THE PREVIOUS RESERVED INSTRUCTION TEST.
5830
5831 023142 005227 177777 KBTST: INC #-1 ;FIRST TIME?
5832 023146 001032 BNE KBDONE ;BRANCH IF NOT
5833 023150 005037 001272 CLR @#KB11E ;CLEAR KB11E AND KB11EM FLAGS
5834 023154 012737 023174 000010 MOV #MFPTTR,@#RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
5835 023162 000007 MFPT ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
5836 ;KB11-CM
5837 023164 012737 000001 001272 MOV #1,@#KB11E ;HERE IF KB11E OR KB11EM. SET FLAG
5838 023172 000403 BR ENDKTB ;DONE DETERMINING WHICH CPU
5839
5840 023174 MFPTTR: ;HERE IF MFPT TRAPPED.
5841 023174 012716 023202 MOV #ENDKTB,(SP) ;SET UP RETURN ADDRESS FOR RTI
5842 023200 000002 RTI ;RETURN
5843 023202 012737 000012 000010 ENDKTB: MOV #12,@#RESVEC ;RESTORE RESERVE VECTOR
5844
5845 023210 104400 036440 TYPE ,MSG1 ;<15><12>CPU UNDER TEST FOUND TO BE A
5846 023214 005737 001272 TST @#KB11E ;IS THIS A KB11-E OR KB11-EM?
5847 023220 001003 BNE 101$ ;BR IF EITHER ONE
5848 023222 104400 036477 TYPE ,MSG3 ;KB11-B/C OR KB11-CM<15><12>
5849 023226 000402 BR KBDONE ;SKIP OTHER MESSAGE
5850 023230 104400 036544 101$: TYPE ,MSG5 ;KB11-E OR KB11EM<15><12>
5851 023234 KBDONE:
5852 ;*****
5853 ;*****
5854 ;*TEST 45 T BIT TRAP
5855 ;*
5856 ;* IF BEN 13 FAILS EXECUTION WOULD GO TO BRK.20.
5857 ;* THIS WOULD LOOK LIKE THE TRAP DIDN'T OCCUR.
5858 ;*
5859 ;* IF THE TRAP DOESN'T OCCUR THEN EITHER PDRD PS04(1) DOES NOT GET
5860 ;* TO TMCB AS A HIGH OR IT DOES NOT GET TO TMCB E51(10)
5861 ;* AS A LOW OR E51 IS BAD OR IRCD RTT DOES NOT GET TO TMCB AS A HIGH.
5862 ;*
5863 ;* THIS TEST ALSO CHECKS THAT PS<08> SET WILL INHIBIT A T BIT TRAP IF
5864 ;* THIS IS A KB11-E OR KB11-EM.
5865 ;*****
5866 023234 000004 TST45: SCOPE
5867 023236 012767 023504 155730 MOV #TST46,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
5868 023244 012767 023504 156016 MOV #TST46,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
5869 ;*****
5870 ;THIS CODE TURNS THE T BIT OFF IF IT IS ON.
5871 023252 012746 00340 MOV #PR7,-(SP) ;PUT PRIORITY LEVEL 7 ON SP
5872 023256 012746 023272 MOV #1$,-(SP) ;PUT RETURN ADDRESS ON SP
5873 023262 013767 177776 155700 MOV @#PSW,$TMP3 ;SAVE PSW
5874 023270 000006 RTT ;TURN OFF T BIT
5875 ;*****
5876 023272 012767 023300 155610 1$: MOV #2$,$LPERR ;SETUP LOOP ADDRESS
5877 023300 012706 001100 2$: MOV #1100,SP ;INITIALIZE THE SP
5878 023304 012737 023370 000014 MOV #3$,@#TBITVEC ;SET UP T BIT VECTOR
5879 023312 012737 023366 000010 MOV #8$,@#RESVEC ;SETUP RESVEC
5880 023320 012746 000360 MOV #360,-(SP) ;PUT NEW PSW ON STACK (ENABLE T BIT)
```

```
5881 023324 012746 023332      MOV      #5$,-(SP)      ;PUT RETURN LOCATION ON STACK
5882 023330 000002      RTI                    ;TURN ON T BIT
5883 023332 012746 000340      5$:  MOV      #PR7,-(SP) ;SETUP STACK TO
5884 023336 012746 023352      MOV      #6$,-(SP)      ;TURN OFF T BIT
5885 023342 013767 177776 155636  MOV      @#PSW,$ERPSW   ;SAVE PSW
5886 023350 000006      RTT                    ;TURN T BIT OFF
5887 023352 032767 000020 155626 6$:  BIT      #BIT4,$ERPSW  ;DID T BIT SET?
5888 023360 001401      BEQ      7$            ;BRANCH IF NO
5889 023362 104275      ERROR    275          ;T BIT TRAP FAILED
5890 023364 104276      7$:  ERROR    276          ;T BIT NEVER SET
5891 023366 104300      8$:  ERROR    300          ;TRAP VECTOR DECODE FAILED
5892 023370 005767 155676 3$:  TST      KB11E        ;KB11-E OR KB11-EM?
5893 023374 001443      BEQ      11$          ;DONE IF NOT
5894 023376 012746 000340      MOV      #PR7,-(SP)    ;PUT PRIORITY LEVEL 7 ON SP
5895 023402 012746 023416      MOV      #15$,-(SP)    ;PUT RETURN ADDRESS ON SP
5896 023406 013767 177776 155554  MOV      @#PSW,$TMP3    ;SAVE PSW
5897 023414 000006      RTT                    ;TURN OFF T BIT
5898 023416 012767 023424 155464 15$:  MOV      #12$,$LPERR   ;SET UP LOOP ADDRESS
5899 023424 012706 001100 12$:  MOV      #STACK,SP     ;INIT STACK POINTER
5900 023430 012737 023450 000014  MOV      #10$,@#TBITVEC ;SET UP T BIT VECTOR
5901 023436 012746 000760      MOV      #760,-(SP)    ;NEW PSW ON STACK, ENABLE T BIT AND
5902                                ;SET PS<08> (SUSPEND BIT)
5903 023442 012746 023504      MOV      #11$,-(SP)    ;RETURN LOCATION ON STACK. SHOULD GO TO 11$
5904                                ;INSTEAD OF 10$ ON RTI
5905 023446 000002      RTI                    ;TURN ON T BIT AND PS<08>
5906 023450 012746 000340 10$:  MOV      #PR7,-(SP)    ;SETUP STACK TO
5907 023454 012746 023470      MOV      #13$,-(SP)    ;TURN OFF T BIT
5908 023460 013767 177776 155520  MOV      @#PSW,$ERPSW   ;SAVE PSW
5909 023466 000006      RTT                    ;TURN T BIT OFF
5910 023470 032767 000400 155510 13$:  BIT      #BIT8,$ERPSW  ;DID PS<08> SET?
5911 023476 001401      BEQ      14$          ;BR IF NOT
5912 023500 104277      ERROR    277          ;SETTING PS<08> FAILED TO DISABLE T-TRAP
5913 023502 104376      14$:  ERROR    376          ;PS<08> FAILED TO SET
5914 023504      11$:  ;CONTINUE
5915      ;*****
5916      ;*TEST 46      T BIT TRAP AND RTT
5917      ;*
5918      ;* IF THE INSTRUCTION AFTER THE RTT DOES NOT GET EXECUTED THEN
5919      ;* EITHER IRCD RTT DOES NOT GO LOW OR IT DOES NOT GET TO
5920      ;* TMCB E74(1T) OR TMCB E74 IS BAD.
5921      ;*****
5922 023504 000004      TST46: SCOPE
5923 023506 012767 023556 155554  MOV      #TST47,NEXTTST ;SAVE ADDRESS OF NEXT TEST
5924 023514 005000      CLR      R0            ;ENSURE R0 CLEAR
5925 023516 012706 001100      MOV      #STACK,SP     ;INITIALIZE THE SP
5926 023522 012746 000360      MOV      #360,-(SP)    ;PUT NEW PSW ON STACK (ENABLE T BIT)
5927 023526 012746 023542      MOV      #1$,-(SP)     ;PUT PC ON STACK
5928 023532 012737 023546 000014  MOV      #2$,@#TBITVEC ;SETUP T BIT VECTOR
5929 023540 000006      RTT                    ;TURN T BIT ON WITH RTT
5930 023542 012700 000001 1$:  MOV      #1,R0         ;THIS SHOULD EXECUTE
5931 023546 022700 000001 2$:  CMP      #1,R0         ;DID MOV INSTRUCTION EXECUTE?
5932 023552 001401      BEQ      TST47        ;BRANCH IF YES
5933 023554 104301      ERROR    301          ;RTT DID NOT DISABLE T BIT
5934      ;*****
5935      ;*TEST 47      PRIORITY ARBITRATION
5936      ;*
```

5937  
 5938  
 5939  
 5940  
 5941  
 5942  
 5943  
 5944  
 5945  
 5946  
 5947  
 5948  
 5949  
 5950  
 5951  
 5952  
 5953  
 5954  
 5955  
 5956  
 5957  
 5958  
 5959  
 5960  
 5961  
 5962  
 5963  
 5964  
 5965  
 5966  
 5967 023556 000004  
 5968 023560 012767 026400 155502  
 5969 023566 012767 003706 155310  
 5970 023574 012767 023610 155304  
 5971 023602 012737 032652 000014  
 5972 023610 005005  
 5973 023612 005004  
 5974 023614 005003  
 5975 023616 032737 000020 001170  
 5976 023624 001417  
 5977 023626 012737 000360 000066  
 5978 023634 012737 000360 000242  
 5979 023642 012737 000360 000006  
 5980 023650 012746 000360  
 5981 023654 012746 023706  
 5982 023660 000006  
 5983 023662 000411  
 5984 023664 012737 000340 000066 71\$:  
 5985 023672 012737 000340 000006  
 5986 023700 012737 000340 000242  
 5987 023706 032737 000400 177570 72\$:  
 5988 023714 001006  
 5989 023716 032737 000040 177570  
 5990 023724 001402  
 5991 023726 005205 102\$:  
 5992 023730 000403

```

    : * THIS TEST ASSURES THAT EACH NECESSARY INPUT TO AN HONOR FLAG
    : * CAN DISABLE THAT FLAG.
    : *
    : * EACH SECTION WILL PERFORM A SETUP SO THAT A TIGHT ERROR LOOP
    : * CAN BE OBTAINED.
    : *
    : * THE FOLLOWING IS A TABLE OF CONTENTS OF THIS TEST:
    : *
    : * SECTION NUMBER LEVEL UNDER TEST DISABLING FUNCTION
    : *
    : * 1 PIR 1 BR 4
    : * 2 PIR 1 SL YELLOW
    : * 3 PIR 2 SL YELLOW
    : * 4 PIR 3 SL YELLOW
    : * 5 BR 4 PIR 4
    : * 6 BR 4 PIR 5
    : * 7 BR 4 BR 5
    : * 8 BR 4 PIR 6
    : * 9 BR 4 PIR 7
    : * 10 PIR 4 BR 5
    : * 11 PIR 4 BR 6
    : * 12 PIR 4 SL YELLOW
    : * 13 BR 5 PIR 5
    : * 14 BR 5 PIR 6
    : * 15 BR 5 PIR 7
    : * 16 PIR 5 BR 6
    : * 17 PIR 5 SL YELLOW
    : * 18 BR 6 PIR 6
    : * 19 BR 6 PIR 7
    : * 20 PIR 6 SL YELLOW
    : * 21 PIR 7 SL YELLOW
    : *
    : * *****
    
```

```

TST47: SCOPE
      MOV #TST50,NEXTTST :SAVE ADDRESS OF NEXT TEST
CHGE1: MOV #^D1990,$ICNT :SETUP ITERATION COUNT
CHGE2: MOV #103$,$LPADR :SETUP LOOP ADDRESS
      MOV #SRTN,@TBITVEC :RESTORE T BIT VEC
103$: CLR R5 :CLEAR BR5 DISABLE FLAG
      CLR R4 :CLEAR BR6 DISABLE FLAG
      CLR R3 :CLEAR BR4 DISABLE FLAG
      BIT #BIT4,@$STMP3 :WAS T BIT ON?
      BEQ 71$ :BRANCH IF NO
      MOV #360,@TPVEC+2
      MOV #360,@PIRQVEC+2
      MOV #360,@ERRVEC+2
      MOV #360,-(SP) :SET UP STACK PSW
      MOV #72$,-(SP) :PUT RETURN ADDRESS ON STACK
      RTT :TURN T BIT ON
      BR 72$
71$: MOV #PR7,@TPVEC+2 :PUT PRIORITY 7 IN PRINTER VECTOR
      MOV #PR7,@ERRVEC+2 :RESTORE ERRVEC PSW
      MOV #PR7,@PIRQVEC+2 :PUT PRIORITY 7 IN PIRQ VECTOR
72$: BIT #SW8,@$SWR :SWITCH 8 ON?
      BNE 100$ :BRANCH IF YES
      BIT #SW5,@$SWR :IS BR 5 TESTING DISABLED?
      BEQ 100$ :BRANCH IF NO
102$: INC R5 :SET BR 5 DISABLE FLAG
      BR 101$ :CONTINUE
    
```

```
5993 023732 005767 155266 100$: TST INTERS ;IS THERE A BR 5 DEVICE?  
5994 023736 001773 BEQ 102$ ;BRANCH IF NO  
5995 :*****  
5996 :SECTION 1 - BR4 AND PIR1  
5997 023740 032737 000400 177570 101$: BIT #SW8,@#SWR ;SWITCH 8 ON?  
5998 023746 001006 BNE 68$ ;BRANCH IF YES  
5999 023750 032737 000020 177570 BIT #SW4,@#SWR ;IS BR4 TESTING DISABLED?  
6000 023756 001402 BEQ 68$ ;BRANCH IF NO  
6001 023760 005203 INC R3 ;SET BR4 FLAG  
6002 023762 000432 BR 2$ ;GO TO NEXT SECTION  
6003 023764 012767 024012 155116 68$: MOV #1$,SLPERR ;SETUP ERROR LOOP  
6004 023772 012737 024050 000064 MOV #2$,@#TPVEC ;SETUP PRINTER VECTOR  
6005 024000 012737 024036 000240 MOV #3$,@#PIRQVEC ;SETUP PIRQ VECTOR  
6006 :***** CHGE1 *****  
6007 024006 005077 155124 CLR @#STKS ;CON'T ALLOW KEYBD INTERRUPTS  
6008 :*****  
6009 024012 012706 001076 1$: MOV #1076,SP ;INITIALIZE THE SP  
6010 024016 000237 SPL 7 ;ENSURE CPU AT LEVEL 7  
6011 024020 052737 001000 177772 BIS #BIT9,@#PIRQ ;SET PIR LEVEL 1  
6012 024026 004767 002212 JSR PC,LEVEL4 ;GO GET BR4  
6013 024032 000230 SPL 0 ;SET CPU AT ZERO  
6014 024034 000240 NOP ;USED WITH SPL  
6015 :FAILURE PIR CAME THRU  
6016 024036 005037 177772 3$: CLR @#PIRQ ;CLEAR THE PIRQ  
6017 024042 005077 155074 CLR @#STPS ;CLEAR THE PRINTER FLAG  
6018 024046 104302 ERROR 302 ;PIR 1 DID NOT DISABLE ON BR  
6019 :*****  
6020 :SECTION 2 - PIR 1 AND SL YELLOW  
6021 024050 005077 155066 2$: CLR @#STPS ;CLEAR PRINTER INT FLAG  
6022 024054 012767 024102 155026 MOV #4$,SLPERR ;SETUP ERROR LOOP  
6023 024062 012737 024140 000004 MOV #5$,@#ERRVEC ;SETUP LOCATION 4  
6024 024070 012737 024120 000240 MOV #6$,@#PIRQVEC ;SETUP PIRQ VECTOR  
6025 024076 005037 177772 CLR @#PIRQ ;CLEAR LEVEL 1  
6026 024102 012706 000376 4$: MOV #376,SP ;SETUP THE SP  
6027 024106 052737 001000 177772 BIS #BIT9,@#PIRQ ;SET LEVEL 1  
6028 024114 000230 SPL 0 ;SET CPU AT ZERO  
6029 024116 011616 MOV (SP),(SP) ;EXECUTE YELL ZONE INSTR  
6030 :FAILURE, PIR CAME THRU  
6031 024120 012706 001100 6$: MOV #STACK,SP ;RESET SP  
6032 024124 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESTORE LOCATION 4  
6033 024132 005037 177772 CLR @#PIRQ ;CLEAR LEVEL 1  
6034 024136 104303 ERROR 303 ;PIR CAME THRU ON YELLOW ZONE  
6035 :  
6036 :*****  
6037 :SECTION 3 - PIR 2 AND SL YELLOW  
6038 024140 005037 177772 5$: CLR @#PIRQ ;CLEAR LEVEL 1  
6039 024144 012767 024166 154736 MOV #7$,SLPERR ;SETUP ERROR LOOP  
6040 024152 012737 024224 000004 MOV #8$,@#ERRVEC ;SETUP LOCATION 4  
6041 024160 012737 024204 000240 MOV #9$,@#PIRQVEC ;SETUP PIRQ VECTOR  
6042 024166 012706 000376 7$: MOV #376,SP ;SETUP THE SP  
6043 024172 052737 002000 177772 BIS #BIT10,@#PIRQ ;SET LEVEL 2  
6044 024200 000231 SPL 1 ;SET CPU AT LEVEL 1  
6045 024202 011616 MOV (SP),(SP) ;EXECUTE TRAP CAUSING INSTR  
6046 :FAILURE, PIR CAME THRU  
6047 024204 012706 001100 9$: MOV #STACK,SP ;RESET SP  
6048 024210 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESET ERRVEC
```

```
6049 024216 005037 177772 CLR @PIRQ ;CLEAR LEVEL 2
6050 024222 104304 ERROR 304 ;PIR 2 CAME THRU ON YELLOW ZONE
6051
6052
6053 :*****
:SECTION 4 - PIR 3 AND YEL ZONE
8$: CLR @PIRQ ;CLEAR LEVEL 2
MOV #10$, $LPERR ;SETUP ERROR LOOP
MOV #11$, @ERRVEC ;SETUP ERR VECTOR
MOV #12$, @PIRQVEC ;SETUP PIRQ VECTOR
10$: MOV #376, SP ;SETUP THE SP
BIS #BIT11, @PIRQ ;SET LEVEL 3
SPL 2 ;SET CPU AT LEVEL 2
MOV (SP), (SP) ;EXECUTE TRAP CAUSING INSTR
:FAILURE, PIR CAME THRU
12$: MOV #STACK, SP ;RESET SP
MOV #CPUSPUR, @ERRVEC ;RESET LOCATION 4
CLR @PIRQ ;CLEAR LEVEL 3
ERROR 305 ;PIR 3 CAME THRU ON YELLOW ZONE
:*****
:SECTION 5- PIR4 AND BR4
11$: MOV #STACK, SP ;RESTORE THE SP
TST R3 ;IS BR4 TESTING DISABLED?
BEQ 70$ ;BRANCH IF NO
JMP 29$ ;SKIP BR4 SECTIONS
70$: MOV #13$, $LPERR ;SETUP ERROR LOOP
MOV #15$, @PIRQVEC ;SETUP PIRQ VEC
MOV #14$, @TPVEC ;SETUP TPVEC
13$: MOV #STACK, SP ;INITIALIZE THE SP
MOV #BIT12, @PIRQ ;SET PIR LEVEL 4
JSR PC, LEVEL4 ;GO GET BR 4
SPL 3 ;LOWER CPU
NOP ;REQUIRED BECAUSE OF SPL
:FAILURE, BR4 CAME THRU
14$: CLR @STPS ;CLEAR PRINTER INT FLAG
CLR @PIRQ ;CLEAR LEVEL 4
ERROR 306 ;BR4 CAME THRU ON PIR4
:*****
:SECTION 6- PIR 5 AND BR4
15$: MOV #16$, $LPERR ;SETUP ERROR LOOP
MOV #17$, @TPVEC ;SETUP BR4 VEC
MOV #18$, @PIRQVEC ;SETUP PIRQ VEC
16$: MOV #STACK, SP ;INITIALIZE THE SP
MOV #BIT13, @PIRQ ;SET PIR LEVEL 5
JSR PC, LEVEL4 ;GO GET BR4
SPL 3 ;SET CPU AT 3
NOP ;REQUIRED BECAUSE OF SPL
:FAILURE, BR4 CAME THRU
17$: CLR @STPS ;CLEAR BR4 INT FLAG
CLR @PIRQ ;CLEAR PIR5
ERROR 307 ;BR4 CAME THRU ON PIR5
:*****
:SECTION 7- BR5 AND BR4
18$: CLR @PIRQ ;CLEAR PIR LEVEL 5
```

```
6105 024464 005077 154452 CLR @STPS ;CLEAR PRINTER INT FLAG
6106 024470 005705 TST R5 ;IS BR 5 TESTING DISABLED?
6107 024472 001040 BNE 23$ ;BRANCH IF YES
6108 024474 012767 024542 154406 20$: MOV #21$, $LPERR ;SETUP ERROR LOOP
6109 024502 012737 024562 000064 MOV #22$, @#TPVEC ;SETUP BR4 VECTOR
6110 024510 016700 154512 MOV INT5VEC, R0 ;GET BR5 VECTOR
6111 024514 012720 024574 MOV #23$, (R0)+ ;PUT ADDRESS OF 23$ IN VECTOR
6112 024520 032737 000020 177776 BIT #BIT4, @#PSW ;IS T BIT ON?
6113 024526 001403 BEQ 73$ ;BRANCH IF NO
6114 024530 012710 000360 MOV #360, (R0) ;SETUP VECTOR PSW
6115 024534 000402 BR 21$
6116 024536 012710 000340 73$: MOV #PR7, (R0) ;PUT PR7 IN VECTOR+2
6117 024542 012706 001100 21$: MOV #STACK, SP ;INITIALIZE THE SP
6118 024546 004767 001502 JSR PC, LEVEL5 ;GO GET BR5
6119 024552 004767 001466 JSR PC, LEVEL4 ;GO GET BR4
6120 024556 000233 SPL 3 ;SET CPU AT LEVEL 3
6121 024560 000240 NOP ;REQUIRED BECAUSE OF SPL
6122 ;FAILURE, BR4 CAME IN
6123 024562 004767 001534 22$: JSR PC, KILBR5 ;GET RID OF BR 5
6124 024566 005077 154350 CLR @STPS ;CLEAR BR4 INT ENABLE
6125 024572 104310 ERROR 310 ;BR4 CAME THRU ON BR5
6126
6127
6128 ;*****
6129 ;SECTION 8- PIR6 AND BR4
6130 024574 004767 001522 23$: JSR PC, KILBR5 ;GET RID OF BR 5
6131 024600 012767 024622 154302 MOV #24$, $LPERR ;SETUP ERROR LOOP
6132 024606 012737 024644 000064 MOV #25$, @#TPVEC ;SETUP BR4 INTERRUPT VECTOR.
6133 024614 012737 024656 000240 MOV #26$, @#PIRQVEC ;SETUP PIRQ VECTOR
6134 024622 012706 001076 24$: MOV #1076, SP ;INITIALIZE THE SP
6135 024626 012737 040000 177772 MOV #BIT14, @#PIRQ ;SET PIR LEVEL 6
6136 024634 004767 001404 JSR PC, LEVEL4 ;GO GET BR4
6137 024640 000233 SPL 3 ;SET CPU AT LEVEL 3
6138 024642 000240 NOP ;REQUIRED BECAUSE OF SPL
6139 ;FAILURE, BR4 CAME IN
6140 024644 005037 177772 25$: CLR @#PIRQ ;CLEAR PIR LEVEL 6
6141 024650 005077 154266 CLR @STPS ;CLEAR PRINTER INTER FLAG
6142 024654 104311 ERROR 311 ;BR4 CAME IN ON PIR6
6143
6144 ;*****
6145 ;SECTION 9- PIR 7 AND BR4
6146 024656 012767 024700 154224 26$: MOV #27$, $LPERR ;SETUP ERROR LOOP
6147 024664 012737 024722 000064 MOV #28$, @#TPVEC ;SETUP BR4 VECTOR
6148 024672 012737 024734 000240 MOV #29$, @#PIRQVEC ;SETUP PIRQ VECTOR
6149 024700 012706 001076 27$: MOV #1076, SP ;SETUP THE SP
6150 024704 012737 100000 177772 MOV #BIT15, @#PIRQ ;SET PIR LEVEL 7
6151 024712 004767 001326 JSR PC, LEVEL4 ;GO GET BR4
6152 024716 000233 SPL 3 ;LOWER CPU TO 3
6153 024720 000240 NOP ;REQUIRED BECAUSE OF SPL
6154 ;FAILURE, BR4 CAME IN
6155 024722 005077 154214 28$: CLR @STPS ;CLEAR PRINTER INTERR FLAG
6156 024726 005037 177772 CLR @#PIRQ ;CLEAR PIR LEVEL 7
6157 024732 104312 ERROR 312 ;BR4 CAME IN ON PIR7
6158
6159 ;*****
6160 024734 005077 154202 29$: CLR @STPS ;CLEAR BR4 INTERR FLAG
```

```
6161 024740 005037 177772 CLR @PIRQ ;CLEAR PIRQ LEVEL 7
6162 024744 005705 TST R5 ;IS THERE A BR5 DEVICE?
6163 024746 001041 BNE 32$ ;BRANCH TO SECTION 11 IF NO
6164 024750 012767 025016 154132 MOV #30$, $LPERR ;SETUP ERROR LOOP
6165 024756 012737 025040 000240 MOV #31$, @PIRQVEC ;SETUP PIRQ VECTOR
6166 024764 016700 154236 MOV INT5VEC, R0 ;GET ADDR OF BR 5 VECTOR
6167 024770 012720 025052 MOV #32$, (R0)+ ;SETUP BR5 VECTOR
6168 024774 032737 000020 177776 BIT #BIT4, @PSW ;IS T BIT ON?
6169 025002 001403 BEQ 74$ ;BRANCH IF NO
6170 025004 012710 000360 MOV #360, (R0)
6171 025010 000462 BR 75$
6172 025012 012710 000340 74$: MOV #PR7, (R0)
6173 025016 012706 001100 30$: MOV #STACK, SP ;SETUP THE SP
6174 025022 012737 010000 177772 MOV #BIT12, @PIRQ ;SET PIR4
6175 025030 004767 001220 JSR PC, LEVEL5 ;GO GET BR5
6176 025034 000233 SPL 3 ;LOWER CPU TO LEVEL 3
6177 025036 000240 NOP ;REQUIRED BECAUSE OF SPL
6178 ;FAILURE, PIR4 CAME IN
6179 025040 004767 001256 31$: JSR PC, KILBR5 ;GET RID OF BR 5
6180 025044 005037 177772 CLR @PIRQ ;CLEAR PIR LEVEL 4
6181 025050 104313 ERROR 313 ;PIR4 CAME IN ON BR5
6182 ;
6183 ;*****
6184 ;SECTION 11- BR6 AND PIR4
6185 025052 005037 177772 32$: CLR @PIRQ ;CLEAR PIR LEVEL 4
6186 025056 004767 001240 JSR PC, KILBR5 ;GET RID OF BR 5
6187 025062 032737 000400 177570 BIT #SW8, @SWR ;SWITCH 8 ON?
6188 025070 001004 BNE 80$ ;BRANCH IF YES
6189 025072 032737 000100 177570 BIT #SW6, @SWR ;IS BR6 TESTING DISABLED?
6190 025100 001003 BNE 33$ ;BRANCH IF YES
6191 025102 005767 154124 80$: TST INTER6 ;IS THERE A BR6 DEVICE?
6192 025106 001002 BNE 34$ ;BRANCH IF YES
6193 025110 005204 33$: INC R4 ;SET BR 6 DISABLE FLAG
6194 025112 000441 BR 37$ ;GO TO SECTION 12
6195 025114 012767 025162 153766 34$: MOV #35$, $LPERR ;SETUP ERROR LOOP
6196 025122 012737 025204 000240 MOV #36$, @PIRQVEC ;SETUP PIR VECTOR
6197 025130 016701 154100 MOV INT6VEC, R1 ;GET BR 6 VECTOR
6198 025134 012721 025216 MOV #37$, (R1)+ ;PU ADDRESS OF 37$ IN VECTOR
6199 025140 032737 000020 177776 BIT #BIT4, @PSW ;IS T BIT ON?
6200 025146 001403 BEQ 75$ ;BRANCH IF NO
6201 025150 012711 000360 MOV #360, (R1)
6202 025154 000402 BR 35$
6203 025156 012711 000340 75$: MOV #PR7, (R1) ;PUT PRIORITY OF 7 IN VECTOR+2
6204 025162 012706 001100 35$: MOV #STACK, SP ;INITIALIZE THE SP
6205 025166 012737 010000 177772 MOV #BIT12, @PIRQ ;SET PIR LEVEL 4
6206 025174 004767 001064 JSR PC, LEVEL6 ;GO GET BR6
6207 025200 000233 SPL 3 ;LOWER CPU TO LEVEL 3
6208 025202 000240 NOP ;REQUIRED BECAUSE OF SPL
6209 ;FAILURE, PIR4 CAME IN
6210 025204 005077 154026 36$: CLR @INT6ST ;CLEAR BR6 INTERR FLAG
6211 025210 005037 177772 CLR @PIRQ ;CLEAR PIR LEVEL 4
6212 025214 104314 ERROR 314 ;PIR4 CAME IN ON BR6
6213 ;
6214 ;*****
6215 ;SECTION 12- PIR4 AND STACK LIMIT YELLOW
6216 025216 005077 154014 37$: CLR @INT6ST ;CLEAR BR6 INTERR. FLAG
```

```
6217 025222 005037 177772          CLR    @#PIRQ          ;CLEAR LEVEL 4
6218 025226 012767 025250 153654    MOV    #38$, $LPERR    ;SETUP ERROR LOOP
6219 025234 012737 025266 000240    MOV    #39$, @#PIRQVEC ;SETUP PIRQ VECTOR
6220 025242 012737 025306 000004    MOV    #40$, @#ERRVEC  ;SETUP YELL ZONE VECTOR
6221 025250 012706 000376          MOV    #376, SP        ;SETUP THE SP
6222 025254 052737 010000 177772 38$:   BIS    #BIT12, @#PIRQ   ;SET LEVEL 4
6223 025262 000233          SPL    3              ;SET CPU AT LEVEL 3
6224 025264 011616          MOV    (SP), (SP)      ;EXECUTE TRAP CAUSING INSTR
6225          ;FAILURE, PIR 4 CAME THRU
6226 025266 012706 001100          39$:   MOV    #STACK, SP    ;RESET THE SP
6227 025272 012737 032666 000004    MOV    #CPUSPUR, @#ERRVEC ;RESTORE ERR VEC
6228 025300 005037 177772          CLR    @#PIRQ          ;CLEAR PIR LEVEL 4
6229 025304 104315          ERROR  315           ;PIR4 CAME IN ON SL YELLOW
6230          ;
6231          ;*****
6232          ;SECTION 13- BR5 AND PIR5
6233 025306 012706 001100          40$:   MOV    #STACK, SP    ;RESTORE THE SP
6234 025312 012737 032666 000004    MOV    #CPUSPUR, @#ERRVEC ;RESTORE LOCATION 4
6235 025320 005037 177772          CLR    @#PIRQ          ;CLEAR PIR 4
6236 025324 005705          TST    R5             ;IS BR5 DISABLED?
6237 025326 001105          BNE    49$           ;BRANCH IF YES TO SECTION 16
6238 025330 012767 025352 153552    MOV    #41$, $LPERR    ;SETUP ERROR LOOP
6239 025336 012777 025374 153662    MOV    #42$, @#INT5VEC ;SETUP BR5 VECTOR
6240 025344 012737 025406 000240    MOV    #43$, @#PIRQVEC ;SETUP PIRQ VECTOR
6241          ;
6242 025352 012706 001076          41$:   MOV    #1076, SP       ;INITIALIZE THE SP
6243 025356 012737 020000 177772    MOV    #BIT13, @#PIRQ  ;SET PIR LEVEL 5
6244 025364 004767 000664          JSR    PC, LEVEL5     ;GO GET BR5
6245          ;
6246 025370 000234          SPL    4             ;LOWER CPU TO LEVEL 4
6247 025372 000240          NOP                 ;REQUIRED BECAUSE OF SPL
6248          ;FAILURE, BR5 CAME IN
6249 025374 004767 000722          42$:   JSR    PC, KILBR5    ;GET RID OF BR 5
6250 025400 005037 177772          CLR    @#PIRQ          ;CLEAR PIR5
6251 025404 104316          ERROR  316           ;BR5 CAME IN ON PIR5
6252          ;
6253          ;*****
6254          ;SECTION 14- BR5 AND PIR6
6255 025406 012767 025430 153474          43$:   MOV    #44$, $LPERR    ;SETUP ERROR LOOP
6256 025414 012777 025452 153604    MOV    #45$, @#INT5VEC ;SETUP BR5 VEC
6257 025422 012737 025464 000240    MOV    #46$, @#PIRQVEC ;SETUP PIRQ VECTOR
6258 025430 012706 001076          44$:   MOV    #1076, SP       ;INITIALIZE THE SP
6259 025434 012737 040000 177772    MOV    #BIT14, @#PIRQ  ;SET PIR LEVEL 6
6260 025442 004767 000606          JSR    PC, LEVEL5     ;GO GET BR5
6261 025446 000234          SPL    4             ;SET CPU AT LEVEL 4
6262 025450 000240          NOP                 ;REQUIRED BECAUSE OF SPL
6263          ;FAILURE, BR5 CAME IN
6264 025452 004767 000644          45$:   JSR    PC, KILBR5    ;GET RID OF BR5
6265 025456 005037 177772          CLR    @#PIRQ          ;CLEAR PIR LEVEL 6
6266 025462 104317          ERROR  317           ;BR5 CAME IN ON PIR6
6267          ;
6268          ;*****
6269          ;SECTION 15- BR5 AND PIR7
6270 025464 012767 025506 153416          46$:   MOV    #47$, $LPERR    ;SETUP ERROR LOOP
6271 025472 012777 025530 153526    MOV    #48$, @#INT5VEC ;SETUP BR5 VECTOR
6272 025500 012737 025542 000240    MOV    #49$, @#PIRQVEC ;SETUP PIRQ VECTOR
```



```

6273 025506 012706 001076 47$: MOV #1076,SP ;INITIALIZE THE SP
6274 025512 012737 100000 177772 MOV #BIT15,@#PIRQ ;SET PIR LEVEL 7
6275 025520 004767 000530 JSR PC,LEVEL5 ;GO GET BR5
6276 025524 000234 SPL 4 ;ALLOW BRQ
6277 025526 000240 NOP ;REQUIRED BECAUSE OF SPL
6278 ;FAILURE, BR5 CAME IN
6279 025530 004767 000566 48$: JSR PC,KILBR5 ;GET RID OF BR5
6280 025534 005037 177772 CLR @#PIRQ ;CLEAR PIR LEVEL 7
6281 025540 104320 ERROR 320 ;BR5 CAME IN ON PIR7
6282
6283 ;*****
6284 ;SECTION 16- PIR5 AND BR6
6285 025542 004767 000554 49$: JSR PC,KILBR5 ;GET RID OF BR 5
6286 025546 005704 TST R4 ;IS BR6 TESTING DISABLED?
6287 025550 001030 BNE 52$ ;BRANCH IF YES TO SECTION 17
6288 025552 012767 025574 153330 MOV #50$,SLPERR ;SETUP ERROR LOOP
6289 025560 012737 025620 000240 MOV #51$,@#PIRQVEC ;SETUP PIR VECTOR
6290 025566 012777 025632 153440 MOV #52$,@#INT6VEC ;SETUP BR6 VECTOR
6291 025574 012706 001076 50$: MOV #1076,SP ;INITIALIZE THE SP
6292 025600 012737 020000 177772 MOV #BIT13,@#PIRQ ;SET IR LEVEL 5
6293 025606 004767 000452 JSR PC,LEVEL6 ;GO GET BR6
6294 025612 000234 SPL 4 ;ALLOW BRQ
6295 025614 000240 NOP ;REQUIRED BECAUSE OF SPL
6296 ;FAILURE, PIR 5 CAME IN
6297 025616 000000 HALT ;DEBUG ONLY
6298 025620 005077 153412 51$: CLR @#INT6ST ;CLEAR BR6 INTERR FLAG
6299 025624 005037 177772 CLR @#PIRQ ;CLEAR PIR LEVEL 5
6300 025630 104321 ERROR 321 ;PIR 5 CAME IN ON BR6
6301
6302 ;*****
6303 ;SECTION 17- PIR 5 AND STACK LIMIT YELLOW
6304 025632 005077 153400 52$: CLR @#INT6ST ;CLEAR BR6 INTERR FLAG
6305 025636 012767 025660 153244 MOV #53$,SLPERR ;SETUP ERROR LOOP
6306 025644 012737 025676 000240 MOV #54$,@#PIRQVEC ;SETUP PIRQVEC
6307 025652 012737 025716 000004 MOV #55$,@#ERRVEC ;SETUP LOCATION 4
6308 025660 012706 000376 53$: MOV #376,SP ;SETUP THE SP
6309 025664 052737 020000 177772 BIS #BIT13,@#PIRQ ;SET LEVEL 5
6310 025672 000234 SPL 4 ;SET CPU AT 4
6311 025674 011616 MOV (SP),(SP) ;EXECUTE YEL ZONE INSTR
6312 ;FAILURE, PIR5 CAME IN ON SL YELLOW
6313 025676 005037 177772 54$: CLR @#PIRQ ;CLEAR LEVEL 5
6314 025702 012737 032666 000004 MOV #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
6315 025710 012706 001100 MOV #STACK,SP ;RESET THE SP
6316 025714 104322 ERROR 322 ;PIR5 CAME IN ON SL YELLOW
6317
6318 ;*****
6319 ;SECTION 18- BR6 AND PIR6
6320 025716 012706 001100 55$: MOV #STACK,SP ;RESTORE THE SP
6321 025722 005704 TST R4 ;IS BR6 TESTING DISABLED?
6322 025724 001056 BNE 61$ ;BRANCH IF YES TO SECTION 20
6323 025726 012767 025750 153154 MOV #56$,SLPERR ;SETUP ERROR LOOP
6324 025734 012777 025772 153272 MOV #57$,@#INT6VEC ;SETUP BR6 INTERR VECTOR
6325 025742 012737 026004 000240 MOV #58$,@#PIRQVEC ;SETUP PIR VECTOR
6326 025750 012706 001076 56$: MOV #1076,SP ;SETUP THE SP
6327 025754 012737 040000 177772 MOV #BIT14,@#PIRQ ;SET PIR LEVEL 6
6328 025762 004767 000276 JSR PC,LEVEL6 ;GO GET BR6
    
```

```
6329 025766 000235          SPL      5          :LOWER CPU TO 5
6330 025770 000240          NOP          :REQUIRED BECAUSE OF SPL
6331          :FAILURE, BR6 CAME IN ON PIR6
6332 025772 005077 153240 57$: CLR      @INT6ST      :CLEAR BR6 INTERR FLAG
6333 025776 005037 177772      CLR      @PIRQ         :CLEAR LEVEL 6
6334 026002 104323          ERROR     323         :BR6 CAME IN ON PIR6
6335
6336          :*****
6337          :SECTION 19- BR6 AND PIR7
6338 026004 012767 026026 153076 58$: MOV      #59$, $LPERR      :SETUP ERROR LOOP
6339 026012 012777 026050 153214      MOV      #60$, @INT6VEC     :SETUP BR6 VECTOR
6340 026020 012737 026062 000240      MOV      #61$, @PIRQVEC    :SETUP PIRQ VECTOR
6341 026026 012706 001076          59$: MOV      #1076, SP       :SETUP THE SP
6342 026032 012737 100000 177772      MOV      #BIT15, @PIRQ     :SET PIR 7
6343 026040 004767 000220          JSR      PC, LEVEL6       :GO GET BR6
6344 026044 000235          SPL      5          :LOWER CPU TO 5
6345 026046 000240          NOP          :REQUIRED BECAUSE OF SPL
6346          :FAILURE, BR6 CAME IN ON PIR7
6347 026050 005077 153162 60$: CLR      @INT6ST      :CLEAR BR6 INTERR FLAG
6348 026054 005037 177772      CLR      @PIRQ         :CLEAR LEVEL 7
6349 026060 104324          ERROR     324         :BR6 CAME IN ON PIR7
6350
6351          :*****
6352          :SECTION 20- PIR6 AND SL YELLOW
6353 026062 012767 026104 153020 61$: MOV      #62$, $LPERR      :SETUP LOOP ADDRESS
6354 026070 012737 026122 000240      MOV      #63$, @PIRQVEC    :SETUP PIRQ VECTOR
6355 026076 012737 026134 000004      MOV      #64$, @ERRVEC     :SETUP LOCATION 4
6356 026104 012706 000376          62$: MOV      #376, SP       :SETUP THE SP
6357 026110 052737 040000 177772      BIS      #BIT14, @PIRQ     :SET LEVEL 6
6358 026116 000235          SPL      5          :SET CPU AT LEVEL 5
6359 026120 011616          MOV      (SP), (SP)      :EXECUTE YELL ZONE INSTR
6360          :FAILURE, PIR 6 CAME IN ON SL YELLOW
6361 026122 005037 177772 63$: CLR      @PIRQ         :CLEAR PIR 6
6362 026126 012706 001100      MOV      #STACK, SP      :RESTORE THE SP
6363 026132 104325          ERROR     325         :PIR 6 CAME IN ON SL YELLOW
6364
6365          :*****
6366          :SECTION 21-PIR7 AND STACK LIMIT YELLOW
6367 026134 005077 153076          64$: CLR      @INT6ST      :ENSURE BR6 FLAG CLEAR
6368 026140 012767 026162 152742      MOV      #65$, $LPERR      :SETUP ERROR LOOP
6369 026146 012737 026200 000240      MOV      #66$, @PIRQVEC    :SETUP PIRQ VECTOR
6370 026154 012737 026212 000004      MOV      #67$, @ERRVEC     :SETUP THE ERROR VECTOR
6371 026162 012706 000376          65$: MOV      #376, SP       :SETUP THE SP
6372 026166 052737 100000 177772      BIS      #BIT15, @PIRQ     :SET LEVEL 7
6373 026174 000236          SPL      6          :SET UP AT 6
6374 026176 011616          MOV      (SP), (SP)      :CAUSE YELL ZONE
6375          :FAILURE, PIR 7 CAME IN ON SL YELLOW
6376 026200 005037 177772 66$: CLR      @PIRQ         :CLEAR LEVEL 7
6377 026204 012706 001100      MOV      #STACK, SP      :RESTORE THE SP
6378 026210 104326          ERROR     326         :PIR7 CAME IN ON SL YELLOW
6379
6380          :END OF TEST
6381 026212 012706 001100          67$: MOV      #STACK, SP      :RESET THE SP
6382 026216 005037 177772      CLR      @PIRQ         :CLEAR LEVEL 7
6383 026222 012737 032666 000004      MOV      #CPUSPUR, @ERRVEC :RESTORE LOCATION 4
6384 026230 005037 177766          CLR      @CPUERR        :CLEAR OUT ERROR REG
```

```
6385 :***** CHGE2 *****
6386 026234 012777 000100 152674 MOV #BIT6,@STKS ;ALLOW KEYBD INTERRUPTS
6387 :*****
6388 026242 000456 BR TST50 ;:CONTINUE
6389
6390 ;ROUTINE'S TO GET BUS REQUESTS
6391 026244 052777 000100 152670 LEVEL4: BIS #100,@STPS ;GET A BR 4
6392 026252 000445 BR WAIT ;GO WAIT
6393 026254 012777 000311 152746 LEVEL5: MOV #311,@INT5ST ;START BR 5 INTERRUPT
6394 026262 000441 BR WAIT ;GO WAIT
6395 026264 026727 152744 000100 LEVEL6: CMP INT6VEC,#100 ;IS BR6 DEVICE KW11-L?
6396 026272 001004 BNE 1$ ;BRANCH IF NO
6397 026274 012777 000100 152734 MOV #BIT6,@INT6ST ;SET INTERR BIT
6398 026302 000431 BR WAIT ;GO WAIT
6399 026304 012737 000001 172544 1$: MOV #1,@#PLKC ;SET KW11-P COUNTER
6400 026312 012777 000105 152716 MOV #105,@INT6ST ;SET INTERR BIT
6401 026320 000422 BR WAIT
6402 026322 012777 026360 152676 KILBR5: MOV #2$,@INT5VEC ;SET UP INTER 5 VEC
6403 026330 042777 000100 152672 BIC #BIT6,@INT5ST ;ENSURE INTERRUPT FLAG CLEAR
6404 026336 005037 177772 CLR @PIRQ ;ENSURE PIRQ REG CLEAR
6405 026342 005077 152574 CLR @STPS
6406 026346 000230 SPL 0 ;LET BR 5 COME IN
6407 026350 005000 CLR R0
6408 026352 005200 3$: INC R0
6409 026354 001376 BNE 3$
6410 026356 000406 BR ENDGET ;DON'T CHANGE STACK
6411 026360 062706 000004 2$: ADD #4,SP ;RESTORE THE SP
6412 026364 000403 BR ENDGET
6413 026366 005000 WAIT: CLR R0 ;WAIT FOR
6414 026370 005200 2$: INC R0 ;THE INTERRUPT
6415 026372 001376 BNE 2$ ;TO COME IN
6416 026374 000237 ENDGET: SPL 7
6417 026376 000207 RTS PC ;RETURN
6418 :*****
6419 ;*TEST 50 GPR SET 1 SELECT TEST
6420 ;*
6421 ;* THIS TEST FIRST ENSURES THAT PSW BIT 11 SETS AND CLEARS.
6422 ;* IT THEN ENSURES THAT GRAC GDREG SET 1 AND GSREG SET 1 GOES
6423 ;* HIGH FOR THE MUX SELECTS LL,LH,AND HL.
6424 ;* MUX SELECT HH WILL BE TESTED IN SUPERVISOR MODE.
6425 :*****
6426 026400 000004 TST50: SCOPE
6427 026402 012767 027030 152564 MOV #TST51,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
6428 026410 012767 027030 152652 MOV #TST51,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
6429 026416 052737 004000 177776 BIS #BIT11,@#PSW ;SET REG SET SELECT BIT
6430 026424 032737 004000 177776 BIT #BIT11,@#PSW ;DID BIT 11 SET?
6431 026432 001004 BNE 20$ ;BRANCH IF YES
6432 026434 042737 004000 177776 BIC #BIT11,@#PSW
6433 026442 104327 ERROR 327 ;EITHER PSW BIT 11 DOES NOT SET OR TDCF
6434 ;CLK HI PS DOES NOT GO LOW OR PSW BIT
6435 ;11 DOES NOT GET TO OR THRU THE DMUX.
6436 026444 105037 177777 20$: CLRB @#PSW+1 ;CLEAR BIT 11
6437 026450 032737 004000 177776 BIT #BIT11,@#PSW ;DID BIT 11 CLEAR?
6438 026456 001402 BEQ 1$ ;BRANCH IF YES
6439 026460 104377 ERROR 377
6440 026462 000457 ;PSW BIT 11 DID NOT CLEAR
```

```

6441
6442
6443 026464 005067 152504
6444 026470 012767 026476 152412
6445 026476 012702 177777
6446 026502 052737 004000 177776
6447 026510 005002
6448 026512 042737 004000 177776
6449 026520 005702
6450 026522 001002
6451 026524 104330
6452 026526 000403
6453 026530 020202
6454 026532 001401
6455 026534 104331
6456
6457
6458 026536 012767 026544 152344
6459 026544 005002
6460 026546 052737 004000 177776
6461 026554 012202
6462 026556 042737 004000 177776
6463 026564 005702
6464 026566 001402
6465 026570 104332
6466 026572 000403
6467 026574 020202
6468 026576 001401
6469 026600 104333
6470
6471
6472 026602 012767 026610 152300
6473 026610 012705 026672
6474 026614 052737 004000 177776
6475 026622 012705 026646
6476 026626 020527 026646
6477 026632 001401
6478 026634 000475
6479 026636 020505
6480 026640 001401
6481 026642 000472
6482 026644 006400
6483 026646 026705 177774
6484 026652 001425
6485 026654 042737 004000 177776
6486 026662 012706 001100
6487 026666 104334
6488 026670 000416
6489 026672 022705 026672
6490 026676 001407
6491 026700 042737 004000 177776
6492 026706 012706 001100
6493 026712 104335
6494 026714 000404
6495 026716 042737 004000 177776
6496 026724 104336

```

```

:*****
:UPAD 5 TEST
1$: CLR $ESCAPE
MOV #10$, $LPERR
10$: MOV #-1, R2 ; SETUP R2
BIS #BIT11, @#PSW ; SELECT REG SET 1
CLR R12 ; CLEAR R12
BIC #BIT11, @#PSW ; GO BACK TO REG SET 0
TST R2 ; DID R2 CLEAR?
BNE 2$ ; BRANCH IF NO
ERROR 330 ; CLEAR R10 ACTUALLY CLEARED R2
BR 3$
2$: CMP R2, R2 ; WAS R2 SOURCE AFFECTED BY R12?
BEQ 3$ ; BRANCH IF NO
ERROR 331 ; R2 SRC WAS AFFECTED BY CLR R12
:*****
:UPAD 0 TEST
3$: MOV #11$, $LPERR
11$: CLR R2 ; SETUP R2
BIS #BIT11, @#PSW ; GO TO REG SET 1
MOV (R12)+, R12 ; EXECUTE A UPAD 0 INSTRUCTION
BIC #BIT11, @#PSW ; COME BACK TO SET 0.
TST R2 ; DID DESTINATION CHANGE
BEQ 4$ ; BRANCH IF NO
ERROR 332 ; R2 DST GOT CHANGED ON (R12)+
BR 5$
4$: CMP R2, R2 ; DID SRC R2 GET CHANGED?
BEQ 5$ ; BRANCH IF NO
ERROR 333 ; R2 SRC GOT CHANGED ON (R12)+
:*****
:UPAD 2 TEST
5$: MOV #12$, $LPERR
12$: MOV #9$, R5 ; SETUP R5
BIS #BIT11, @#PSW ; GO TO SET 1
MOV #6$, R15 ; SETUP R15
CMP R15, #6$ ; IS THERE STUCK BITS IN R15 SRC?
BEQ 7$ ; BRANCH IF NO
BR TST51 ; CAN'T DO THIS TEST, GO TO STUCK BIT TEST
7$: CMP R15, R15 ; IS THERE STUCK BITS IN R15 DST?
BEQ 8$ ; BRANCH IF NO
BR TST51 ; CAN'T DO THIS TEST, GO TO STUCK BIT TEST
8$: MARK 0 ; EXECUTE A UPAD 2 INSTRUCTION
6$: CMP 6$, R15 ; DID R15 DST GET LOADED?
BEQ 16$ ; BRANCH IF YES
BIC #BIT11, @#PSW ; GO BACK TO SET 0
MOV #STACK, SP ; RESTORE THE SP
ERROR 334 ; R15 DST BAD AFTER UPAD2
BR 16$
9$: CMP #9$, R15 ; IS DST ALSO BAD?
BEQ 15$ ; BRANCH IF YES
BIC #BIT11, @#PSW ; RESTORE THE SP
MOV #STACK, SP ; R15 SRC DOES NOT SELECT ON UPAD2
ERROR 335
BR 16$
15$: BIC #BIT11, @#PSW
ERROR 336 ; PDRD PS11 DOES NOT GET TO GRAC

```

```
6497  
6498  
6499 026726 012706 001100  
6500 026732 012767 026740 152150  
6501 026740 042737 004000 177776  
6502 026746 005004  
6503 026750 052737 004000 177776  
6504 026756 005204  
6505 026760 042737 004000 177776  
6506 026766 005704  
6507 026770 001401  
6508 026772 104341  
6509  
6510  
6511 026774 012767 027002 152106
```

```
*****  
:TEST GRAB E19(2 & 3)  
16$: MOV #STACK,SP ;RESET THE SP  
MOV #14$,SLPERR  
14$: BIC #BIT11,@#PSW ;GO BACK TO SET 0  
CLR R4 ;ENSURE R4 CLEAR  
BIS #BIT11,@#PSW ;GO TO SET 1  
INC R14 ;INCREMENT R14  
BIC #BIT11,@#PSW ;GO BACK TO SET 0  
TST R4 ;WAS R4 AFFECTED?  
BEQ 17$ ;BRANCH IF NO  
ERROR 341 ;GRAB DST SET 1 DID NOT GO LOW ON R14  
*****  
:TEST GRAB E18(2 & 3)  
17$: MOV #13$,SLPERR
```

```

6512 027002 052737 004000 177776 13$: BIS #BIT11,@#PSW ;GO TO SET 1
6513 027010 005004 CLR R14 ;ENSURE EVEN ADDRESS IN R14
6514 027012 012404 MOV (R14)+,R14 ;EXECUTE A UPAD 0
6515 027014 042737 004000 177776 BIC #BIT11,@#PSW ;GO BACK TO SET 0
6516 027022 005704 TST R4 ;WAS R4 AFFECTED?
6517 027024 001401 BEQ TST51 ;BRANCH IF NO
6518 027026 104342 ERROR 342 ;GRAB SRC SET 1 DID NOT GO LOW ON R14
6519

```

```

6520 *****
6521 *TEST 51 REGISTER SET 1 STUCK BIT TEST
6522 *
6523 * THIS TEST ENSURES THAT ALL BITS IN GPR'S R10 THRU R15 WORK
6524 *****

```

```

6525 027030 000004 TST51: SCOPE
6526 027032 012767 027244 152134 MOV #TST52,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
6527 027040 012767 027244 152222 MOV #TST52,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
6528 027046 005067 152110 CLR $TMP0 ;INITIALIZE PASS COUNT
6529 027052 052737 004000 177776 BIS #BIT11,@#PSW ;GO TO REG SET 1
6530 027060 012737 4$: MOV (PC)+,@(PC)+ ;SAVE EXPECTED VALUE
6531 027062 125252 5$: .WORD 125252
6532 027064 001164 .WORD $TMP1 ;ADDRESS OF $TMP1
6533 027066 012700 MOV (PC)+,R10 ;PUT PATTERN IN R10
6534 027070 125252 6$: .WORD 125252
6535 027072 022700 CMP (PC)+,R10 ;IS R10 DST OK?
6536 027074 125252 7$: .WORD 125252
6537 027076 001040 BNE 1$ ;BRANCH IF NO
6538 027100 020027 CMP R10,(PC)+ ;IS R10 SRC OK?
6539 027102 125252 8$: .WORD 125252
6540 027104 001034 BNE 2$ ;BRANCH IF NO
6541 027106 010001 MOV R10,R11
6542 027110 020001 CMP R10,R11 ;IS R11 DST OK?
6543 027112 001032 BNE 1$ ;BRANCH IF NO
6544 027114 020100 CMP R11,R10 ;IS R11 SRC OK?
6545 027116 001027 BNE 2$ ;BRANCH IF NO
6546 027120 010102 MOV R11,R12
6547 027122 020002 CMP R10,R12 ;IS R12 DST OK?
6548 027124 001025 BNE 1$ ;BRANCH IF NO
6549 027126 020200 CMP R12,R10 ;IS R12 SRC OK?
6550 027130 001022 BNE 2$ ;BRANCH IF NO
6551 027132 010003 MOV R10,R13
6552 027134 020003 CMP R10,R13 ;IS R13 DST OK
6553 027136 001020 BNE 1$ ;BRANCH IF NO
6554 027140 020300 CMP R13,R10 ;IS R13 SRC OK?
6555 027142 001015 BNE 2$ ;BRANCH IF NO
6556 027144 010004 MOV R10,R14
6557 027146 020004 CMP R10,R14 ;IS R14 DST OK?
6558 027150 001013 BNE 1$ ;BRANCH IF NO
6559 027152 020400 CMP R14,R10 ;IS R14 SRC OK?
6560 027154 001010 BNE 2$ ;BRANCH IF NO
6561 027156 010005 MOV R10,R15
6562 027160 020005 CMP R10,R15 ;IS R15 DST OK
6563 027162 001006 BNE 1$ ;BRANCH IF NO
6564 027164 020500 CMP R15,R10 ;IS R15 SRC OK
6565 027166 001410 BEQ 3$ ;BRANCH IF YES
6566 027170 042737 004000 177776 BIC #BIT11,@#PSW
6567 027176 104337 2$: ERROR 337 ;BAD BITS IN GPR SET 1 SRC

```

```
6568 027200 042737 004000 177776 1$: BIC #BIT11,@#PSW
6569 027206 104340 ERROR 340 ;BAD BITS IN GPR SET 1 DST
6570 027210 005767 151746 3$: TST $TMP0 ;IS THIS FIRST PASS?
6571 027214 001013 BNE TST52 ;BRANCH IF NO
6572 027216 005267 151740 INC $TMP0 ;SET PASS COUNT
6573 027222 005167 177634 COM 5$ ;GO TO
6574 027226 005167 177636 COM 6$ ;COMPLIMENT
6575 027232 005167 177636 COM 7$ ;PATTERN
6576 027236 005167 177640 COM 8$
6577 027242 000706 BR 4$ ;GO TEST COMPLIMENT PATTERN
```

```
6578
6579
6580 *****
6581 *TEST 52 PSW HIGH BYTE BIT TEST
6582 *
6583 * THIS TEST ENSURES THAT THE PRESENT AND PREVIOUS MODE BITS OF THE PSW
6584 * CAN BE SET AND CLEARED AND THAT THEY ARE NOT STUCK TOGETHER.
6585 *****
```

```
TST52: SCOPE
6586 027244 000004 MOV #TST53,NEXTTST ;SAVE ADDRESS OF NEXT TEST
6587 027246 012767 027414 152014 MOVB #120,@#PSW+1 ;SET TO SUPER & PREVIOUS SUPER
6588 027254 112737 000120 177777 CMPB #120,@#PSW+1 ;IS IT OK?
6589 027262 122737 000120 177777 BEQ 1$ ;BRANCH IF YES
6590 027270 001412 BEQ 1$ ;BRANCH IF YES
6591 027272 012767 050000 151662 MOV #50000,$TMP0 ;SAVE EXPECTED VALUE
6592 027272 012767 050000 151662 MOV @#PSW,$ERPSW ;SAVE ERROR VALUE
6593 027300 013767 177776 151700 BIC #377,$ERPSW ;MASK OFF HIGH BYTE
6594 027306 042767 000377 151672 ERROR 343 ;PATTERN FAILED
6595 027314 104343 ERROR 343 ;PATTERN FAILED
6596 027316 112737 000350 177777 1$: MOVB #350,@#PSW+1 ;SET COMPLIMENT PATTERN
6597 027324 122737 000350 177777 CMPB #350,@#PSW+1 ;IS IT OK?
6598 027332 001412 BEQ 2$ ;BRANCH IF YES
6599 027334 012767 164000 151620 MOV #164000,$TMP0 ;SAVE EXPECTED VALUE
6600 027334 012767 164000 151620 MOV @#PSW,$ERPSW ;SAVE ERROR VALUE
6601 027342 013767 177776 151636 BIC #377,$ERPSW ;MASK OFF HIGH BYTE
6602 027350 042767 000377 151630 ERROR 344 ;PATTERN FAILED
6603 027356 104344 ERROR 344 ;PATTERN FAILED
6604 027360 105037 177777 2$: CLRB @#PSW+1 ;CLEAR ALL BITS
6605 027364 105737 177777 TSTB @#PSW+1 ;DID THEY ALL CLEAR?
6606 027370 001411 BEQ TST53 ;BRANCH IF YES
6607 027372 005067 151564 CLR $TMP0 ;SAVE EXPECTED VALUE
6608 027376 013767 177776 151602 MOV @#PSW,$ERPSW ;SAVE ERROR VALUE
6609 027404 042767 000377 151574 BIC #377,$ERPSW ;MASK OFF HIGH BYTE
6610 027412 104345 ERROR 345 ;ALL BITS IN PSW DID NOT CLEAR
```

```
6609 *****
6610 *TEST 53 SP SELECTION TEST IN SUPER AND USER MODE
6611 *
6612 * THIS TEST ENSURES THAT THE CORRECT STACK POINTERS ARE
6613 * SELECTED IN SUPERVISOR AND USER MODE
6614 *****
```

```
TST53: SCOPE
6615 027414 000004 MOV #TST54,NEXTTST ;SAVE ADDRESS OF NEXT TEST
6616 027416 012767 027664 151644 ;UPAD 5-SSP
6617 027424 012767 027432 151456 MOV #5,$LPERR ;SETUP ERROR LOOP
6618 027432 012706 001100 5$: MOV #STACK,KSP ;SETUP THE KERNAL SP
6619 027436 052737 040000 177776 BIS #BIT14,@#PSW ;GO TO SUPER MODE
6620 027444 005006 CLR SSP ;CLEAR THE SUPER SP (UPAD 5)
6621 027446 042737 040000 177776 BIC #BIT14,@#PSW ;GO BACK TO KERNEL MODE
6622 027454 005706 TST KSP ;DID THE KSP CHANGE?
6623
```

```

6624 027456 001003          BNE      1$          ;BRANCH IF NO
6625 027460 012706 001100  MOV      #STACK,KSP ;RESTORE THE KSP
6626 027464 104346          ERROR    346        ;SUPER SP DOES NOT SELECT ON UPAD 5
6627
6628          ;UPAD 0-SSP
6629 027466 012767 027474 151414 1$:  MOV      #6$, $LPERR ;SETUP ERROR LOOP
6630 027474 012706 001100 6$:  MOV      #STACK,KSP ;INITIALIZE THE SP
6631 027500 052737 040000 177776  BIC      #BIT14,@#PSW ;GO TO SUPER MODE
6632 027506 012706 001776  MOV      #1776,SSP    ;SETUP SUPER SP
6633 027512 012606  MOV      (SSP)+,SSP   ;EXECUTE A UPAD 0 TYPE OPERATION
6634 027514 042737 040000 177776  BIC      #BIT14,@#PSW ;GO BACK TO KERNEL
6635 027522 020627 001100  CMP      KSP,#STACK  ;DID KSP CHANGE?
6636 027526 001403  BEQ      2$          ;BRANCH IF NO
6637 027530 012706 001100  MOV      #STACK,KSP ;RESTORE THE KSP
6638 027534 104347          ERROR    347        ;SUPER SP DOES NOT SELECTON UPAD 0
6639
6640          ;UPAD 5-USP
6641 027536 012767 027544 151344 2$:  MOV      #7$, $LPERR ;SETUP ERROR LOOP
6642 027544 052737 040000 177776 7$:  BIC      #BIT14,@#PSW ;GO TO SUPER MODE
6643 027552 012706 177777  MOV      #-1,SSP     ;SET THE SSP TO A NON-ZERO NUMBER
6644 027556 052737 100000 177776  BIC      #BIT15,@#PSW ;GO TO USER MODE
6645 027564 005006  CLR      USP        ;CLEAR R17
6646 027566 042737 100000 177776  BIC      #BIT15,@#PSW ;GO BACK TO SUPER MODE
6647 027574 005706  TST      SSP        ;DID SSP CLEAR?
6648 027576 001003  BNE      3$          ;BRANCH IF NO
6649 027600 105037 177777  CLRB    @#PSW+1    ;GO BACK TO KERNEL
6650 027604 104350          ERROR    350        ;USER SP DOES NOT SELECT ON UPAD 5
6651
6652          ;UPAD 0-USP
6653 027606 012767 027614 151274 3$:  MOV      #8$, $LPERR ;SETUP ERROR LOOP
6654 027614 052737 040000 177776 8$:  BIC      #BIT14,@#PSW ;GO TO SUPER MODE
6655 027622 005006  CLR      SSP        ;ENSURE SSP CLEAR
6656 027624 052737 100000 177776  BIC      #BIT15,@#PSW ;GO TO USER MODE
6657 027632 012706 001776  MOV      #1776,USP   ;SET USP TO EVEN NUMBER
6658 027636 012606  MOV      (USP)+,USP  ;EXECUTE A UPAD 0 TYPE INSTURCTION
6659 027640 042737 100000 177776  BIC      #BIT15,@#PSW ;GO BACK TO SUPER
6660 027646 005706  TST      SSP        ;DID SSP CHANGE?
6661 027650 001403  BEQ      4$          ;BRANCH IF NO
6662 027652 105037 177777  CLRB    @#PSW+1    ;GO BACK TO KERNEL
6663 027656 104351  ERROR    351        ;USER SP DOES NOT SELECT ON UPAD 0
6664 027660 105037 177777 4$:  CLRB    @#PSW+1    ;GO BACK TO KERNAL
6665
6666          ;CONTINUE
6667
6668          ;*****
6669          ;*TEST 54      SUPER AND USER SP BIT TEST
6670          ;*
6671          ;*      THIS TEST ENSURES THAT THE SUPERVISOR AND USER STACK POINTERS
6672          ;*      DON'T HAVE ANY STUCK BITS.
6673          ;*      *****
6674          ;TST54:  SCOPE
6675          ;MOV      #TST55,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
6676          ;MOV      #TST55,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
6677          ;CLR      $TMP1          ;CLEAR LOOP COUNT
6678          ;BIS      #BIT14,@#PSW  ;GO TO SUPER MODE
6679          ;MOV      (PC)+,SP        ;PUT PATTERN IN SP
          ;6$:  .WORD    52525        ;DATA PATTERN
    
```



6680	027720	022706				CMP	(PC)+,SP	:IS DST OK?
6681	027722	052525			7\$:	.WORD	52525	:DATA PATTERN
6682	027724	001403				BEQ	1\$	:BRANCH IF YES
6683	027726	105037	177777			CLRB	@#PSW+1	:GO BACK TO KERNEL
6684	027732	104352				ERROR	352	:DST FAILED
6685	027734	020627			1\$:	CMP	SP,(PC)+	:IS SRC OK?
6686	027736	052525			8\$:	.WORD	52525	:DATA PATTERN
6687	027740	001403				BEQ	2\$	:BRANCH IF YES
6688	027742	105037	177777			CLRB	@#PSW+1	:GO BACK TO KERNEL
6689	027746	104353				ERROR	353	:SRC FAILED
6690	027750	005737	177776		2\$:	TST	@#PSW	:IS BIT 15 SET?
6691	027754	100404				BMI	3\$	:BRANCH IF YES
6692	027756	052737	100000	177776		BIS	#BIT15,@#PSW	:GO TO USER MODE
6693	027764	000753				BR	4\$	:GO CHECK USER STACK POINTER
6694	027766	005767	151172		3\$:	TST	\$TMP1	:IS THIS SECOND PASS?
6695	027772	001014				BNE	5\$	:BRANCH IF YES
6696	027774	005167	177716			COM	6\$	:CHANGE TEST
6697	030000	005167	177716			COM	7\$	:TO COMPLIMENT
6698	030004	005167	177726			COM	8\$	:PATTERN
6699	030010	042737	100000	177776		BIC	#BIT15,@#PSW	:GO BACK TO SUPER MODE
6700	030016	005267	151142			INC	\$TMP1	:SET PASS COUNT
6701	030022	000734				BR	4\$	:GO TEST COMPLIMENT PATTERN
6702	030024	105037	177777		5\$:	CLRB	@#PSW+1	:GO BACK TO KERNAL
6703								:CONTINUE

6704  
6705  
6706  
6707  
6708  
6709  
6710  
6711  
6712  
6713  
6714  
6715  
6716  
6717  
6718  
6719  
6720  
6721  
6722  
6723  
6724  
6725  
6726  
6727  
6728  
6729  
6730  
6731  
6732  
6733  
6734  
6735

```
*****  
:TEST 55 MTP*DMO*DF6*PREVIOUS MODE(SUPER*USER)  
:  
:* THIS TEST ENSURES THAT THE CORRECT SP'S ARE SELECTED WHEN EXECUTING  
:* A MTP WITH DIFFERENT PREVIOUS MODE BITS SELECTED.  
:*****
```

6711	030030	000004				TST55:	SCOPE	
6712	030032	012767	030264	151230		MOV	#TST56,NEXTTST	:SAVE ADDRESS OF NEXT TEST
6714						:	SECTION 1-POP THE KSP AND PUT IT IN THE SSP	
6715	030040	012706	001100			MOV	#STACK,KSP	:SETUP THE KERNEL SP
6716	030044	012746	177777			MOV	#-1,-(KSP)	:PUT -1 ON THE STACK
6717	030050	005000				CLR	R0	:CLEAR ERROR COUNT
6718	030052	052737	010000	177776		BIS	#BIT12,@#PSW	:SET PREVIOUS MODE SUPER
6719	030060	106606				MTPD	SP	:EXECUTE INSTRUCTION UNDER TEST
6720	030062	022706	001100			CMP	#STACK,KSP	:DID THE KSP DST COME OUT OK?
6721	030066	001401				BEQ	2\$	:BRANCH IF YES
6722	030070	005200				INC	R0	:SET THE ERROR COUNT
6723	030072	020627	001100		2\$:	CMP	KSP,#STACK	:DID THE KSP SRC COME OUT OK?
6724	030076	001410				BEQ	3\$	:BRANCH IF YES
6725	030100	012706	001100			MOV	#STACK,KSP	:RESTORE THE SP
6726	030104	005700				TST	R0	:DID DST ALSO FAIL?
6727	030106	001002				BNE	4\$	:BRANCH IF YES
6728	030110	104354				ERROR	354	:KSP SRC WAS CHANGED
6729	030112	000431				BR	9\$	
6730	030114	104355			4\$:	ERROR	355	:BOTH KSP SRC AND DST CHANGED
6731	030116	000427				BR	9\$	
6732	030120	005700			3\$:	TST	R0	:DID KSP DST CHANGE
6733	030122	001404				BEQ	5\$	:BRANCH IF NO
6734	030124	012706	001100			MOV	#STACK,KSP	:RESTORE THE SP
6735	030130	104356				ERROR	356	:KSP DST WAS CHANGED

```

6736 030132 000421 BR 9$
6737 030134 052737 040000 177776 5$: BIS #BIT14,@#PSW ;GO TO SUPER MODE
6738 030142 022706 177777 CMP #-1,SSP ;DID SSP LOAD OK?
6739 030146 001412 BEQ 6$ ;BRANCH IF YES
6740 030150 010667 151000 MOV SSP,$REG0 ;SAVE THE SSP
6741 030154 005006 CLR SSP ;FOR LOOPING
6742 030156 042737 040000 177776 BIC #BIT14,@#PSW ;GO BACK TO KERNEL
6743 030164 012767 177777 150764 MOV #-1,$REG1 ;SAVE EXPECTED VALUE
6744 030172 104357 ERROR 357 ;SSP DID NOT LOAD PROPERLY
6745
6746 ;SECTION 2-POP THE KSP AND PUT IT IN THE USP
6747 030174 005006 6$: CLR SSP ;ENSURE SSP CLEAR FOR ITERATIONS
6748 030176 042737 040000 177776 9$: BIC #BIT14,@#PSW ;GO BACK TO KERNEL
6749 030204 012767 030212 150676 MOV #7$,$LPERR ;SET UP ERROR LOOP
6750 030212 012706 001100 7$: MOV #STACK,KSP ;SETUP THE SP
6751 030216 012746 177777 MOV #-1,-(KSP) ;PUT -1 ON THE KERNEL STACK
6752 030222 052737 030000 177776 BIS #30000,@#PSW ;SET PREVIOUS MODE USER
6753 030230 106606 MTPD SP ;EXECUTE INSTRUCTION UNDER TEST
6754 030232 052737 140000 177776 BIS #140000,@#PSW ;GO TO USER MODE
6755 030240 020627 177777 CMP USP,#-1 ;DID USER SP GET LOADED?
6756 030244 001404 BEQ 8$ ;BRANCH IF YES
6757 030246 005006 CLR USP ;CLEAR USER SP
6758 030250 105037 177777 CLRB @#PSW+1 ;GO BACK TO KERNEL
6759 030254 104360 ERROR 360 ;USER SP DID NOT LOAD ON MTP
6760 030256 005006 8$: CLR USP ;ENSURE USP CLEAR, IF ITERATING
6761 030260 105037 177777 CLRB @#PSW+1 ;GO BACK TO KERNEL
6762 ;CONTINUE
6763
6764 ;*****
6765 ;*TEST 56 MFP*DMO*DF6*PREVIOUS MODE SUPER
6766 ;*
6767 ;* THE ONLY POSSIBLE WAY THIS TEST CAN FAIL IS THAT
6768 ;* IRCC DMO (MFP+MTP) DOES NOT GO HIGH ON MFP.
6769 ;* THIS WILL ONLY HAPPEN IF THE IR DECODE ROM HAS
6770 ;* A BAD FIELD (R(MFP+MTP)).
6771 ;*****
6771 030264 000004 TST56: SCOPE
6772 030266 012767 030412 150700 MOV #TST57,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
6773 030274 012767 030412 150766 MOV #TST57,NEXTTST ;SAVE START ADDRESS OF NEXT TEST
6774
6775 ;PUSH THE SSP ONTO THE KERNEL STACK
6776 030302 012767 030346 150576 MOV #1$,$LPADR ;SETUP LOOP ADR
6777 030310 012767 030346 150572 MOV #1$,$LPERR ;SETUP ERROR LOOP
6778 030316 013767 177776 150644 MOV @#PSW,$TMP3 ;SAVE PSW
6779 030324 032737 000020 177776 BIT #BIT4,@#PSW ;IS T BIT ON?
6780 030332 001405 BEQ 1$ ;BRANCH IF NO
6781 030334 012746 000340 MOV #PR7,-(SP) ;PUT NEW PSW ON STACK
6782 030340 012746 030346 MOV #1$,-(SP) ;PUT RETURN ADDR ON STACK
6783 030344 000006 RTT ;TURN T BIT OFF
6784 030346 052737 040000 177776 1$: BIS #BIT14,@#PSW ;GO TO SUPER MODE
6785 030354 005006 CLR SSP ;CLEAR THE SUPER SP
6786 030356 105037 177777 CLRB @#PSW+1 ;GO BACK TO KERNEL
6787 030362 012706 001100 MOV #STACK,KSP ;SETUP THE KSP
6788 030366 012766 177777 177776 MOV #-1,-2(KSP) ;PUT KNOWN VALUE ON STACK
6789 030374 052737 010000 177776 BIS #BIT12,@#PSW ;SET PREVIOUS MODE SUPER
6790 030402 106506 MFPD SP ;EXECUTE INSTRUCTION UNDER TEST
6791 030404 005716 TST (KSP) ;DID STACK GET SUPER SP?
    
```

6792 030406 001401  
6793 030410 104361  
6794  
6795  
6796  
6797  
6798  
6799  
6800  
6801  
6802  
6803  
6804 030412 000004  
6805 030414 012767 030622 150646  
6806 030422 012706 001100  
6807 030426 052737 040000 177776  
6808 030434 012706 001064  
6809 030440 012716 030500  
6810 030444 012766 140340 000002  
6811 030452 052737 100000 177776  
6812 030460 012706 001060  
6813 030464 012716 030550  
6814 030470 012766 140340 000002  
6815 030476 000002  
6816  
6817 030500 022706 001070  
6818 030504 001004  
6819 030506 105037 177777  
6820 030512 104362  
6821 030514 000415  
6822 030516 042737 100000 177776  
6823 030524 022706 001070  
6824 030530 001004  
6825 030532 105037 177777  
6826 030536 104363  
6827 030540 000403  
6828 030542 105037 177777  
6829 030546 104364  
6830  
6831 030550 052737 140000 177776  
6832 030556 022706 001064  
6833 030562 001415  
6834 030564 042737 100000 177776  
6835 030572 022706 001064  
6836 030576 001004  
6837 030600 105037 177777  
6838 030604 104365  
6839 030606 000403  
6840 030610 105037 177777  
6841 030614 104366  
6842 030616 105037 177777  
6843  
6844  
6845  
6846  
6847

```
BEQ TST57 ;:BRANCH IF YES
ERROR 361 ;:IR DECODE ROM BAD

;:*****
;:*TEST 57 UPAD 7 IN USER MODE
;:*
;:* THIS TEST ENSURES THAT A UPAD 7(OCCURS IN RTI) CAUSES THE USER
;:* STACK POINTER TO BE USED TO FETCH THE NEW PS AND PC.
;:*
;:* IF PDRD PS14(1) DOES NOT GET TO THE GSAM(ON GRAC)
;:* THE TEST WILL BLOW UP.
;:*****
TST57: SCOPE
MOV #TST60,NEXTTST ;:SAVE ADDRESS OF NEXT TEST
MOV #STACK,KSP ;:SETUP THE KERNAL SP
BIS #BIT14,@#PSW ;:GO TO SUPER MODE
MOV #1064,SSP ;:SETUP THE SSP
MOV #1$, (SSP) ;:PUT ADDRESS OF 1$ ON SUPER STACK
MOV #140340,2(SSP) ;:PUT NEW PSW ON SUPER STACK
BIS #BIT15,@#PSW ;:GO TO USER MODE
MOV #1060,USP ;:SETUP THE USP
MOV #2$, (USP) ;:PUT ADDRESS OF 2$ ON USER STACK
MOV #140340,2(USP) ;:PUT NEW PSW ON USER STACK
RTI ;:EXECUTE INSTRUCTION THAT USES UPAD 7
;:FAILURE, SUPER STACK POINTER WAS READ
1$: CMP #1070,USP ;:DID USP DST GET WRITTEN?
BNE 3$ ;:BRANCH IF NO
CLRB @#PSW+1 ;:GO BACK TO KERNEL
ERROR 362 ;:SSP WAS READ AND USP WAS WRITTEN
BR 2$
3$: BIC #BIT15,@#PSW ;:GO TO SUPER MODE
CMP #1070,SSP ;:WAS THE SSP WRITTEN>
BNE 4$ ;:BRANCH IF NO
CLRB @#PSW+1 ;:GO BACK TO KERNEL
ERROR 363 ;:SSP WAS READ AND WRITTEN
BR 2$
4$: CLRB @#PSW+1 ;:GO BACK TO KERNEL
ERROR 364 ;:DON'T KNOW WHAT HAPPENED
;:READ OK,NOW CHECK WRITE
2$: BIS #140000,@#PSW ;:SETUP PSW
CMP #1064,USP ;:DID THE USP GET WRITTEN?
BEQ 5$ ;:BRANCH IF YES
BIC #BIT15,@#PSW ;:GO TO SUPER MODE
CMP #1064,SSP ;:DID THE SSP GET WRITTEN?
BNE 6$ ;:BRANCH IF NO
CLRB @#PSW+1 ;:GO TO KERNEL MODE
ERROR 365 ;:USP WAS READ BUT SSP WAS WRITTEN
BR 5$
6$: CLRB @#PSW+1 ;:GO TO KERNEL MODE
ERROR 366 ;:USP WAS READ BUT REG 7 WAS WRITTEN
5$: CLRB @#PSW+1 ;:GO TO KERNEL
;:CONTINUE

;:*****
;:*TEST 60 SPL*SUPERVISOR MODE
```

6848  
6849  
6850  
6851 030622 000004  
6852 030624 000237  
6853 030626 052737 040000 177776  
6854 030634 000230  
6855 030636 042737 040000 177776  
6856 030644 013767 177776 150334  
6857 030652 042767 177437 150326  
6858 030660 022767 000340 150320  
6859 030666 001404  
6860 030670 012767 000340 150264  
6861 030676 104367  
6862  
6863  
6864  
6865  
6866  
6867  
6868  
6869  
6870  
6871  
6872  
6873  
6874  
6875  
6876  
6877  
6878  
6879  
6880  
6881 030700 000004  
6882 030702 013767 177776 150254  
6883 030710 012767 174000 150244  
6884 030716 005767 150350  
6885 030722 001403  
6886 030724 052767 000400 150230  
6887 030732  
6888 030732 012767 031444 150330  
6889 030740 005067 150242  
6890  
6891  
6892 030744 012767 030752 150136  
6893 030752 012706 001072  
6894 030756 012716 030772  
6895 030762 012766 177757 000002  
6896 030770 000002  
6897 030772 005767 150274  
6898 030776 001405  
6899 031000 122737 000371 177777  
6900 031006 001413  
6901 031010 000404  
6902 031012 122737 000370 177777  
6903 031020 001406

```

: *
: * THIS TEST ENSURES THAT SPL DOES NOT LOAD THE PSW IN SUPER+USER MODE.
: *
: *****
TST60: SCOPE
SPL 7 ;SET CPU AT LEVEL 7
BIS #BIT14,@#PSW ;GO TO SUPER MODE
SPL 0 ;TRY AND CLEAR PRIORITIES
BIC #BIT14,@#PSW ;GO BACK TO KERNEL
MOV @#PSW,$ERPSW ;SAVE PSW
BIC #^C<PR7>,$ERPSW ;MASK OFF PRIORITIES
CMP #PR7,$ERPSW ;DID SPL CLR PRIORITIES
BEQ TST61 ;BRANCH IF NO
MOV #PR7,$TMP0 ;SAVE EXPECTED VALUE
ERROR 367 ;SPL WORKED IN SUPER MODE
: *****
: *TEST 61 PSW CLOCKING TEST
: *
: * THIS TEST ENSURES THAT ALL THE BITS IN THE PSW GET LOADED WHEN THE
: * FOLLOWING SIGNALS ARE TRUE:
: * 1) LOAD PS*KERNEL MODE, AND 2) LOAD PS*KERNEL DATI.
: *
: * IT ALSO ENSURES THAT THE PRESET LOGIC ON BITS 11, 12, 13,
: * 14, AND 15 FUNCTIONS PROPERLY.
: *
: * FOLLOWING IS A TABLE TO DESCRIBE THE PSW FOR EACH SECTION
: *
: * SECTION PSW AT START PSW ON STK(OR VECTOR) EXPEC PSW
: * 1 000XXX 174XXX 174XXX (175XXX IF KB11-E/EM)
: * 2 174XXX 000XXX 174XXX
: * 3 040XXX 134VX 174XXX
: * 4 144XXX 00LAXX 030XXX
: * 5 030XXX 000XXX 000XXX
: *****
TST61: SCOPE
MOV @#PSW,$TMP1 ;SAVE PSW
MOV #174000,$TMP0 ;SAVE EXPECTED VALUE OF PSW
TST KB11E ;KB11-E OR KB11-EM PROCESSOR?
BEQ 22$ ;BR IF NOT
BIS #400,$TMP0 ;KB11-E HAS PS08 WHICH SHOULD SET ON NEXT TEST
22$:
MOV #TST62,NEXTTST ;SAVE ADDRESS OF NEXT TEST
CLR $ERPSW ;ENSURE $ERPSW CLEAR
: *****
: *RTI IN KERNEL MODE WITH NEW PSW<15:11>174 (175 IN KB11-E) AND OLD PSW<15:11>000
: *
11$: MOV #11$,$LPERR ;SETUP ERROR LOOP
MOV #1072,$P ;SETUP THE SP
MOV #1$,(SP) ;PUT ADDRESS OF 1$ ON THE STACK
MOV #177757,2(SP) ;SET ALL BITS IN NEW PSW EXCEPT T
RTI ;EXECUTE INSTRUCTION
1$: TST KB11E ;IS THIS A KB11-E OR KB11-EM PROCESSOR?
BEQ 20$ ;BR IF NOT
CMPB #371,@#PSW+1 ;NEW PSW LOAD OK?
BEQ 2$ ;BR IF YES
BR 21$ ;OTHERWISE REPORT ERROR
20$: CMPB #370,@#PSW+1 ;DID NEW PSW LOAD OK?
BEQ 2$ ;BRANCH IF YES

```

```
6904 031022 113767 177777 150157 21$: MOV @PSW+1,$ERPSW+1 ;SAVE ERROR PSW
6905 031030 105037 177777 CLR @PSW+1 ;GO BACK TO KERNEL
6906 031034 104370 ERROR 370 ;A HIGH BYTE BIT DID NOT SET IN PSW
6907 *****
6908 :RTI IN USER MODE WITH NEW PSW<15:11>000 AND OLD PSW<15:11>174
6909 031036 012767 174000 150116 2$: MOV #174000,$TMP0 ;SAVE EXPECTED VALUE OF PSW
6910 031044 012767 031052 150036 MOV #12,$SLPERR ;SETUP ERROR LOOP
6911 031052 152737 000370 177777 12$: BISB #370,@PSW+1 ;SETUP THE PSW
6912 031060 012706 001072 MOV #1072,$USP ;SETUP USER SP
6913 031064 012716 031076 MOV #3,$($USP) ;PUT ADDRESS OF 3$ ON STACK
6914 031070 005066 000002 CLR 2($USP) ;PUT NEW PSW ON STACK
6915 031074 000002 RTI ;EXECUTE INSTRUCTION
6916 031076 122737 000370 177777 3$: CMPB #370,@PSW+1 ;DID PSW STAY THE SAME?
6917 031104 001406 BEQ 4$ ;BRANCH IF YES
6918 031106 113767 177777 150073 MOVB @PSW+1,$ERPSW+1 ;SAVE PSW
6919 031114 105037 177777 CLR @PSW+1 ;GO BACK TO KERNEL
6920 031120 104371 ERROR 371 ;A HIGH BYTE BIT CLEARED IN PSW
6921 *****
6922 :RTI IN SUPERVISOR MODE WITH NEW PSW<15:11>134 AND OLD PSW<15:11>040
6923 :CHECKS PRESETS ON BITS 11, 12, 13, AND 15
6924 031122 012767 031130 147760 4$: MOV #13,$SLPERR ;SETUP ERROR LOOP
6925 031130 012737 040340 177776 13$: MOV #40340,@PSW ;GO TO SUPER AND CLEAR PREVIOUS MODE AND REG BITS
6926 031136 012706 001072 MOV #1072,$SSP ;SET THE SSP
6927 031142 012716 031156 MOV #5,$($SSP) ;PUT ADDRESS OF 5$ ON STACK
6928 031146 012766 134000 000002 MOV #134000,2($SSP) ;PUT NEW PSW ON STACK
6929 031154 000002 RTI ;EXECUTE INSTRUCTION
6930 031156 122737 000370 177777 5$: CMPB #370,@PSW+1 ;DID ALL BITS SET?
6931 031164 001406 BEQ 10$ ;BRANCH IF YES
6932 031166 113767 177777 150013 MOVB @PSW+1,$ERPSW+1 ;SAVE PSW HIGH BYTE
6933 031174 105037 177777 CLR @PSW+1 ;GO BACK TO KERNEL
6934 031200 104372 ERROR 372 ;BITS <15,13:11> DID NOT PRESET
6935 *****
6936 :IOT IN USER MODE WITH NEW PSW<15:11>000 AND OLD PSW<15:11>144
6937 031202 012767 031210 147700 10$: MOV #14,$SLPERR ;SETUP ERROR LOOP
6938 031210 112737 000310 177777 14$: MOVB #310,@PSW+1 ;SETUP PSW
6939 031216 012737 031234 000020 MOV #6,$@IOTVEC ;PUT ADDRESS OF 6$ IN IOT VECTOR
6940 031224 012737 000340 000022 MOV #PR7,@IOTVEC+2 ;PUT NEW PSW IN IOT VECTOR+2
6941 031232 000004 IOT ;EXECUTE INSTRUCTION
6942 031234 122737 000060 177777 6$: CMPB #60,@PSW+1 ;DID NEW PSW COME UP OK?
6943 031242 001411 BEQ 7$ ;BRANCH IF YES
6944 031244 113767 177777 147735 MOVB @PSW+1,$ERPSW+1 ;SAVE PSW
6945 031252 012767 030000 147702 MOV #30000,$TMP0 ;SAVE EXPECTED VALUE
6946 031260 105037 177777 CLR @PSW+1 ;MAKE SURE KERNEL MODE
6947 031264 104373 ERROR 373 ;PSW HIGH BYTE WRONG
6948 *****
6949 :IOT IN KERNEL MODE WITH NEW PSW<15:11>000 AND OLD PSW<15:11>030
6950 031266 012767 031274 147614 7$: MOV #15,$SLPERR ;SETUP ERROR LOOP
6951 031274 112737 000060 177777 15$: MOVB #60,@PSW+1 ;SETUP PSW
6952 031302 012737 031320 000020 MOV #8,$@IOTVEC ;SETUP IOT VECTOR
6953 031310 012737 030340 000022 MOV #30340,@IOTVEC+2 ;SETUP NEW PSW
6954 031316 000004 IOT ;EXECUTE INSTRUCTION
6955 031320 105737 177777 8$: TSTB @PSW+1 ;IS PSW HIGH BYTE CLEAR?
6956 031324 001430 BEQ 9$ ;BRANCH IF YES
6957 031326 113767 177777 147653 MOVB @PSW+1,$ERPSW+1 ;SAVE PSW
6958 031334 005067 147622 CLR $TMP0 ;SAVE EXPECTED VALUE
6959 031340 104374 ERROR 374 ;PREVIOUS MODE BITS DID NOT CLEAR
```

```
6960  
6961  
6962 031342 012737 074357 177776  
6963 031350 000005  
6964 031352 013767 177776 147626  
6965 031360 026727 147622 074340  
6966 031366 001412  
6967 031370 012767 074340 147564  
6968 031376 005037 177776  
6969 031402 104377  
6970 031404 000461  
6971 031406 012737 033302 000020 9$:  
6972 031414 032767 000020 147542 16$:  
6973 031422 001410  
6974 031424 012706 001074  
6975 031430 016716 147634  
6976 031434 012766 000360 000002  
6977 031442 000002  
6978  
6979  
6980  
6981  
6982  
6983  
6984  
6985  
6986  
6987  
6988  
6989 031444 000004  
6990 031446 012767 031560 147614  
6991 031454 012706 001100  
6992 031460 012737 031506 000004  
6993 031466 052737 040000 177776  
6994 031474 000000  
6995  
6996 031476 105037 177777  
6997 031502 104377  
6998 031504 000417  
6999 031506 032737 000200 177766 1$:  
7000 031514 001010  
7001 031516 013767 177766 147430  
7002 031524 012767 000200 147430  
7003 031532 104377  
7004 031534 000420  
7005 031536 005037 177766 2$:  
7006 031542 005737 177766  
7007 031546 001401  
7008 031550 104256  
7009 031552 012737 032666 000004 3$:  
7010  
7011  
7012  
7013  
7014  
7015
```

```
::*****  
:RESET IN SUPER MODE  
MOV #74357,@#PSW ;SETUP PSW  
RESET ;EXECUTE INSTRUCTION  
MOV @#PSW,$ERPSW ;SAVE PSW  
CMP $ERPSW,#74340 ;WAS PSW OK?  
BEQ 16$ ;BRANCH IF YES  
MOV #74340,$TMPO ;SAVE EXPECTED VALUE  
CLR @#PSW  
ERROR 377 ;RESET AFFECTED BITS IN PSW  
461  
MOV #$$SCOPE,@#IOTVEC ;RESTORE IOTVEC  
BIT #BIT4,$TMP1 ;WAS T BIT ON?  
BEQ TST62 ;BRANCH IF NO  
MOV #1074,$SP ;SETUP THE STACK  
MOV NEXTTST,(SP) ;PUT ADDRESS OF NEXT TEST ON STACK  
MOV #360,2(SP) ;SET T BIT ON STACK  
RTI ;TURN T BIT ON  
*****
```

```
*TEST 62 ILLEGAL HALT  
* THIS TEST ENSURES THAT A HALT IN SUPER OR USER MODE WILL TRAP TO  
* LOCATION 4.  
*  
* IF BEN6 FAILS EXECUTION WOULD GO TO FET.04 WHICH WOULD  
* CAUSE THE HALT TO LOOK LIKE A NOP.  
* IF TMCE SET CONF GOES LOW THE PROCESSOR WILL HALT AT LOCALLOCATION 1$.  
*  
* THE CPU ERROR REGISTER BIT 7 IS ALSO TESTED HERE.  
*****
```

```
TST62: SCOPE  
MOV #TST63,NEXTTST ;SAVE ADDRESS OF NEXT TEST  
MOV #STACK,$SP ;SETUP THE SP  
MOV #1$,@#ERRVEC ;SETUP ERRVEC  
BIS #BIT14,@#PSW ;GO TO SUPER MODE  
HALT ;EXECUTE INSTRUCTION UNDER TEST  
:FAILURE, NO TRAP  
CLRB @#PSW+1 ;GO BACK TO KERNEL  
ERROR 377 ;ILLEGAL HALT DID NOT  
417 ;TRAP TO 4  
1$: BIT #BIT7,@#CPUERR ;IS ERROR REG OK?  
BNE 2$ ;BRANCH IF YES  
MOV @#CPUERR,$REGO ;SAVE FOR TYP0UT  
MOV #200,$TMPO ;SAVE EXPECTED VALUE  
ERROR 377 ;BIT 7 DID NOT SET  
420  
2$: CLR @#CPUERR ;CLEAR BIT 7  
TST @#CPUERR ;DID BIT 7 CLEAR?  
BEQ 3$ ;BRANCH IF YES  
ERROR 256 ;BIT 7 DID NOT CLEAR  
3$: MOV #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC  
;CONTINUE  
*****
```

```
*TEST 63 WAIT  
*  
* THIS TEST ENSURES THAT THE WAIT INSTRUCTION WORKS PROPERLY.  
* IT FIRST EXECUTES WITH A LEVEL 4 INTERRUPT.
```

```
7016 ;* THEN THE T BIT IS ENABLED TO ENSURE THAT THE INTERRUPT
7017 ;* OCCURS AND NOT THE T BIT TRAP.
7018 ;*****
7019 031560 000004 TST63: SCOPE
7020 031562 012767 032054 147500 MOV #TST64,NEXTTST ;SAVE ADDRESS OF NEXT TEST
7021 031570 012767 003706 147306 MOV #^D1990,$ICNT ;ADJUST ITERATION COUNT
7022 031576 012767 031604 147302 MOV #10$, $LPADR ;SETUP LOOP ADR
7023 031604 012737 031654 000064 10$: MOV #2$, @#TPVEC ;SETUP TELEPRINTER VECTOR
7024 031612 012767 031620 147270 MOV #1$, $LPERR ;SETUP ERROR LOOP
7025 031620 012706 001100 1$: MOV #STACK, SP ;INITIALIZE THE SP
7026 031624 000233 SPL 3
7027 031626 012777 000100 147306 MOV #BIT6, @STPS ;SET PRINTER INTERRUPT FLAG
7028 031634 012777 000015 147302 MOV #15, @STPB ;SEND CHARACTER
7029 031642 000001 WAIT ;EXECUTE INSTRUCTION UNDER TEST
7030 ;FAILURE
7031 031644 005077 147272 CLR @STPS ;CLEAR INTERRUPT ENABLE BIT
7032 031650 104377 ERROR 377 ;WAIT INSTRUCTION DID NOT
7033 031652 000440 440 ;WAIT FOR INTERRUPT
7034 ;TEST OK, NO TRY WITH T BIT
7035 031654 012767 031676 147226 2$: MOV #3$, $LPERR ;SETUP ERROR LOOP
7036 031662 012737 031732 000014 MOV #4$, @#TBITVEC ;SETUP T BIT VECTOR
7037 031670 012737 031754 000064 MOV #5$, @#TPVEC ;SETUP TELEPRINTER VECTOR
7038 031676 012706 001100 3$: MOV #STACK, SP ;INITIALIZE THE SP
7039 031702 012746 000020 MOV #20, -(SP) ;PUT SP ON STACK TO TURN ON T BIT
7040 031706 012746 031730 MOV #6$, -(SP) ;PUT RETURN ADDRESS ON STACK
7041 031712 052777 000100 147222 BIS #BIT6, @STPS ;SET PRINTER INTERRUPT BIT
7042 031720 012777 000015 147216 MOV #15, @STPB ;SEND CHARACTER
7043 031726 000006 RTT ;TURN T BIT ON
7044 031730 000001 6$: WAIT ;EXECUTE INSTRUCTION UNDER TEST
7045 ;FAILURE, T BIT CAME IN
7046 031732 005077 147204 4$: CLR @STPS ;CLEAR PRINTER STATUS
7047 031736 012706 001100 MOV #STACK, SP ;RESTORE THE SP
7048 031742 012737 032652 000014 MOV #SRTN, @#TBITVEC ;RESTORE T BIT VECTOR
7049 031750 104377 ERROR 377 ;T BIT CAUSED WAIT TO
7050 031752 000441 441 ;QUIT
7051 ;TEST OK, NOW TRY PIRQ
7052 031754 012737 032652 000014 5$: MOV #SRTN, @#TBITVEC ;RESTORE T BIT VECTOR
7053 031762 012767 032004 147120 MOV #7$, $LPERR ;SETUP ERROR LOOP
7054 031770 012737 032030 000064 MOV #8$, @#TPVEC ;SETUP TP VECTOR
7055 031776 012737 032044 000240 MOV #9$, @#PIRQVEC ;SETUP PIRQ VECTOR
7056 032004 012706 001100 7$: MOV #STACK, SP ;SETUP THE SP
7057 032010 012737 100000 177772 MOV #BIT15, @#PIRQ ;SET PIR LEVEL 7
7058 032016 012777 000015 147120 MOV #15, @STPB ;SEND CHARACTER
7059 032024 000233 SPL 3 ;LOWER CPU
7060 032026 000001 WAIT ;EXECUTE INSTRUCTION UNDER TEST
7061 ;FAILURE, PIRQ DID NOT INTERRUPT
7062 032030 005077 147106 8$: CLR @STPS ;CLEAR TP STATUS
7063 032034 005037 177772 CLR @#PIRQ ;CLEAR PIR LEVEL 7
7064 032040 104377 ERROR 377 ;PIRQ DID NOT CAUSE
7065 032042 000442 442 ;INTERRUPT
7066 ;TEST OK
7067 032044 005077 147072 9$: CLR @STPS ;CLEAR INTERRUPT FLAG
7068 032050 005037 177772 CLR @#PIRQ ;CLEAR PIR LEVEL 7
7069 ;CONTINUE
7070
7071 ;*****
```

7072  
7073  
7074  
7075  
7076  
7077  
7078  
7079  
7080  
7081  
7082  
7083  
7084 032054 000004  
7085 032056 012767 032126 147204  
7086 032064 005737 001272  
7087 032070 001002  
7088 032072 000167 000030  
7089 032076  
7090 032076 012703 000002  
7091 032102 000007  
7092 032104 042700 001400  
7093 032110 020003  
7094 032112 001405  
7095 032114 010337 001164  
7096 032120 010037 001162  
7097 032124 104173  
7098 032126  
7099 032126  
7100  
7101  
7102  
7103  
7104  
7105  
7106  
7107  
7108  
7109  
7110  
7111  
7112  
7113 032126 000004  
7114 032130 012767 032366 147132  
7115 032136 005767 146736  
7116 032142 001006  
7117 032144 032737 000001 177570  
7118 032152 001403  
7119 032154 104400 072571  
7120 032160 000502  
7121 032162 005737 000042  
7122 032166 001077  
7123 032170 012700 166667  
7124 032174 104400 072614  
7125 032200 032737 000200 177570  
7126 032206 001416  
7127 032210 012767 032232 146744

```

: *TEST 64 CHECK MFPT INSTRUCTION (KB-11E/EM ONLY)
: * THE MFPT INSTRUCTION IS NOT AVAILABLE ON THE 11/70 BUT
: * IF THIS IS AN KB-11E/EM THIS TEST IS RUN. MFPT RETURNS
: * DATA TO R0 IN THE FOLLOWING FORMAT:
: * BIT 0 - 1 INDICATES 11/44 CPU
: * BIT 1 - 1 INDICATES KB-11E/EM CPU (SHOULD ALWAYS COME UP IN THIS TEST)
: * BIT 8 - 1 INDICATES CISP PRESENT
: * BIT 9 - 1 INDICATES FP PRESENT
: *
: * THIS TEST MASKS OFF BITS 8 AND 9 AND ONLY CHECKS THAT BITS 0 AND 1
: * COME BACK CORRECTLY AND THAT THE OTHER BITS ARE CLEAR.
: *****
TST64: SCOPE
MOV #MMSET,NEXTTST ;SAVE ADDRESS OF NEXT TEST
TST @#KB11E ;IS THIS A KB11-E OR KB11-EM?
BNE DOMFPT ;BR IF IT IS
JMP DONE7 ;SKIP THIS TEST IF NOT. MFPT NOT THERE

DOMFPT:
MOV #2,R3 ;R3 IS DATA PATTERN. BIT 1 WILL ALWAYS BE SET
MFPT ;EXECUTE INSTRUCTION
BIC #1400,R0 ;MASK OFF CISP AND FP BITS
CMP R0,R3 ;MATCH?
BEQ DONE7 ;DONE IF SO
MOV R3,@#STMP1 ;SET UP EXPECTED (GOOD) DATA
MOV R0,@#STMP0 ;SET UP RECIEVED (BAD) DATA
ERROR 173 ;ERROR PRINTOUT

```

```

DONE7:
MMSET:
: *****

```

```

: *TEST 65 OPERATOR INTERVENTION TEST
: *
: * THIS TEST ENSURES THAT THE RESET AND WAIT FLOWS PUT R0
: * AND THE PC IN THE LIGHTS. THE TEST IS ONLY EXECUTED ON
: * PASS 1 AND CAN BE DISABLED ALTOGETHER WITH SWITCH 0.
: *
: * THE RESET LOOP IS STOPED BY CHANGING THE POSITION OF
: * SWITCH 7.
: *
: * THE WAIT IS EXITED BY TYPING A CHARACTER
: * ON THE TERMINAL.
: *****

```

```

TST65: SCOPE
MOV #SEOP,NEXTTST ;SAVE ADDRESS OF SEOP
TST $PASS ;IS THIS FIRST PASS?
BNE 1$ ;BRANCH IF NO
BIT #SW0,@#SWR ;IS SWITCH 0 ON?
BEQ 2$ ;BRANCH IF NO
TYPE ,EM717 ;TYPE MESSAGE (SKIPPING TEST)
1$: BR SEOP ;EXIT
2$: TST @#42 ;IS MONITOR ACT11?
BNE SEOP ;BRANCH IF YES
MOV #166667,R0 ;SETUP R0
TYPE ,EM720 ;TYPE MESSAGE
BIT #SW7,@#SWR ;IS SWITCH 7 ON?
BEQ 3$ ;BRANCH IF NO
MOV #4$+2,$TMP0 ;SAVE PC

```



```
7128 032216 016746 146740      MOV      $TMP0,-(SP)      ;;SAVE $TMP0 FOR TYPEOUT
7129                                ;;TYPE ADDRESS
7130 032222 104402              TYP0C                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7131 032224 104400 072753      TYPE      ,EM721        ;;TYPE MESSAGE
7132 032230 000005              4$: RESET                ;;EXECUTE INSTRUCTION
7133 032232 032737 000200 177570 BIT      #SW7,@#SWR      ;;HAS SWITCH 7 CHANGED?
7134 032240 001416              BEQ      5$              ;;BRANCH IF YES
7135 032242 000772              BR       4$              ;;LOOP
7136 032244 012767 032266 146710 3$: MOV      #6$+2,$TMP0   ;;SAVE PC
7137 032252 016746 146704      MOV      $TMP0,-(SP)    ;;SAVE $TMP0 FOR TYPEOUT
7138                                ;;TYPE ADDRESS
7139 032256 104402              TYP0C                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7140 032260 104400 072753      TYPE      ,EM721        ;;TYPE MESSAGE
7141 032264 000005              6$: RESET                ;;EXECUTE INSTRUCTION
7142 032266 032737 000200 177570 BIT      #SW7,@#SWR      ;;HAS SWITCH 7 CHANGED?
7143 032274 001773              BEQ      6$              ;;BRANCH IF NO
7144                                ;
7145                                ;WAIT TEST
7146 032276 104400 072614      5$: TYPE      ,EM720    ;;TYPE MESSAGE
7147 032302 012767 032346 146652 MOV      #7$,$TMP0      ;;SAVE PC
7148 032310 016746 146646      MOV      $TMP0,-(SP)    ;;SAVE $TMP0 FOR TYPEOUT
7149                                ;;TYPE ADDRESS
7150 032314 104402              TYP0C                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7151 032316 104400 073011      TYPE      ,EM722        ;;TYPE MESSAGE
7152 032322 005077 146612      CLR      @TKB            ;;CLEAR KEYBOARD BUFFER
7153 032326 012737 032346 000060 MOV      #7$,@TKVEC     ;;SETUP KEYBOARD VECTOR
7154 032334 052777 000100 146574 BIS      #BIT6,@TKS     ;;SET INTERRUPT ENABLE BIT
7155 032342 000233              SPL      3              ;;LOWER PRIORITY FOR INTERRUPT
7156 032344 000001              WAIT                    ;;EXECUTE INSTRUCTION
7157 032346 012737 034656 000060 7$: MOV      #QUIT,@TKVEC ;;RESTORE TKVECTOR
7158 032354 005077 146560      CLR      @TKB            ;;ENSURE BUFFER CLEAR
7159 032360 152777 000100 146550 BISB     #BIT6,@TKS     ;;SET INTERRUPT ENABLE FLAG
7160                                ;;CONTINUE
7161                                ;*****
7162                                ;
7163                                ;SBTTL END OF PASS ROUTINE
7164                                ;
7165                                ;*INCREMENT THE PASS NUMBER ($PASS)
7166                                ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
7167                                ;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
7168                                ;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
7169                                ;*IF SW12=1 INHIBIT TRACE TRAP
7170                                ;*IF THERES A MONITOR GO TO IT
7171                                ;*IF THERE ISN'T JUMP TO LOOP
7172                                ;
7173                                $EOP:
7174 032366 000004              SCOPE
7175 032370 005067 146506      CLR      $STNM          ;;ZERO THE TEST NUMBER
7176 032374 005067 146572      CLR      $TIMES         ;;ZERO THE NUMBER OF ITERATIONS
7177 032400 005267 146474      INC      $PASS          ;;INCREMENT THE PASS NUMBER
7178 032404 042767 100000 146466 BIC      #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
7179 032412 005327              DEC      (PC)+          ;;LOOP?
7180 032414 000001      $EOPCT: .WORD      1
7181 032416 003076      BGT      $DOAGN         ;;YES
7182 032420 012737      MOV      (PC)+,@(PC)+  ;;RESTORE COUNTER
7183 032422 000001      $ENDCT: .WORD      1
```

```
7184 032424 032414 $EOPCT
7185 032426 104400 032434 TYPE .65$ ::TYPE ASCIZ STRING
7186 032432 000407 BR 64$ ::GET OVER THE ASCIZ
7187 ::65$: .ASCIZ <12><15>/END PASS #/
7188 032452 64$:
7189 032452 016746 146422 MOV $PASS,-(SP) ::SAVE $PASS FOR TYPEOUT
7190 ::TYPE PASS NUMBER
7191 032456 104410 TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN
7192 032460 104400 032466 TYPE .67$ ::TYPE ASCIZ STRING
7193 032464 000421 BR 66$ ::GET OVER THE ASCIZ
7194 ::67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
7195 032530 66$:
7196 032530 016746 146356 MOV $ERTTL,-(SP) ::SAVE $ERTTL FOR TYPEOUT
7197 ::TOTAL NUMBER OF ERRORS
7198 032534 104410 TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN
7199 032536 104400 001203 TYPE .SCLRF ::TYPE CARRIAGE RETURN, LINE FEED
7200 032542 005067 146344 CLR $ERTTL ::CLEAR ERROR TOTAL
7201 032546 013700 000042 $GET42: MOV @#42,R0 ::GET MONITOR ADDRESS
7202 032552 001420 BEQ $DOAGN ::BRANCH IF NO MONITOR
7203 032554 005046 CLR -(SP) ::INSURE THE 'T' BIT IS CLEAR
7204 032556 012746 032564 MOV #$CLR.T,-(SP) ::SETUP FOR AN RTI OR RTT
7205 032562 000433 BR $RTRN ::GO DO AN RTI OR RTT TO LOAD THE PSW
7206 ::WITH A CLEARED 'T' BIT
7207 032564 $CLR.T:
7208 032564 012703 000001 MOV #1,R3 ::MAKE R3=1
7209 032570 004767 002120 JSR PC,CHAINQ ::GO RESTORE MONITOR
7210 032574 013700 000042 MOV @#42,R0 ::INSURE R0 CONTAINS THE MONITORS
7211 032600 001405 BEQ $DOAGN ::RETURN ADDRESS
7212 032602 000005 RESET ::CLEAR THE WORLD
7213 032604 004710 $ENDAD: JSR PC,(R0) ::GO TO MONITOR
7214 032606 000240 NOP ::SAVE ROOM
7215 032610 000240 NOP ::FOR
7216 032612 000240 NOP ::ACT11
7217 032614 $DOAGN:
7218 032614 013746 177776 MOV @#PS,-(SP) ::PUT THE PS ON THE STACK AND
7219 032620 042716 000020 BIC #20,(SP) ::CLEAR THE 'T' BIT
7220 032624 032737 010000 177570 BIT #BIT12,@#SWR ::RUN WITH TRACE TRAP?
7221 032632 001005 BNE 1$ ::BR IF NO
7222 032634 005167 000020 COM $TBIT ::IS IT TIME FOR TRACE TRAP
7223 032640 100402 BMI 1$ ::BR IF NO
7224 032642 052716 000020 BIS #20,(SP) ::SET TRACE TRAP
7225 032646 012746 032654 1$: MOV #$LOOP,-(SP) ::JUMP TO START OF TEST
7226 032652 000002 $RTRN: RTI ::RETURN--THIS IS CHANGED TO
7227 ::AN 'RTT' IF 'RTT' IS A LEGAL
7228 ::INSTRUCTION
7229 032654 $LOOP:
7230 032654 000137 005340 JMP @#LOOP ::RETURN
7231 032660 000000 $TBIT: .WORD 0 ::'T' BIT STATE INDICATOR
7232 032662 377 377 000 $ENULL: .BYTE -1,-1,0 ::NULL CHARACTER STRING
7233 .EVEN
7234 ::*****
7235 $SBTTL SPURIOUS ERROR HANDLER
7236 * THIS ROUTINE IS ENTERED BY AN UNEXPECTED TRAP TO 4 OR 114.
7237 * IF SWITCH 13 IS OFF, AN ERROR MESSAGE, THE ERROR REGISTER,
7238 * THE ERROR PC, AND THE TEST NUMBER ARE TYPED OUT.
7239 * IF SWITCH 13 IS ON, ONLY THE ERROR MESSAGE WILL BE TYPED.
```

```
7240
7241 032666 011667 146224
7242 032672 012706 001100
7243 032676 104400 032704
7244 032702 000414
7245
7246 032734
7247 032734 032737 020000 177570
7248 032742 001045
7249 032744 104400 032752
7250 032750 000417
7251
7252 033010
7253 033010 013767 177766 145144
7254 033016 116767 146060 146164
7255 033024 016746 146066
7256
7257 033030 104402
7258 033032 104400 034652
7259 033036 016746 146146
7260
7261 033042 104402
7262 033044 104400 034652
7263 033050 016746 146106
7264
7265 033054 104402
7266 033056 005037 177766
7267 033062 016767 146202 146104
7268 033070 104264
7269 033072 011667 146020
7270 033076 012706 001100
7271 033102 104400 033110
7272 033106 000415
7273
7274 033142
7275 033142 032737 020000 177570
7276 033150 001045
7277 033152 104400 033160
7278 033156 000417
7279
7280 033216
7281 033216 013767 177744 145736
7282 033224 116767 145652 145756
7283 033232 016746 145660
7284
7285 033236 104402
7286 033240 104400 034652
7287 033244 016746 145740
7288
7289 033250 104402
7290 033252 104400 034652
7291 033256 016746 145700
7292
7293 033262 104402
7294 033264 013737 177744 177744
7295 033272 016767 145772 145674
```

\*\*\*\*\*  
CPUSPUR:MOV (SP),\$ERRPC ;SAVE THE ERROR PC  
MOV #STACK,SP ;RESTORE THE SP  
TYPE ,65\$ ;TYPE ASCIZ STRING  
BR 64\$ ;GET OVER THE ASCIZ  
::65\$: .ASCIZ /UNEXPECTED TRAP TO 4/<15><12>  
64\$:  
BIT #BIT13,@#SWR ;IS INHIBIT ERROR TYPEOUT ON?  
BNE 1\$ ;BRANCH IF YES  
TYPE ,67\$ ;TYPE ASCIZ STRING  
BR 66\$ ;GET OVER THE ASCIZ  
::67\$: .ASCIZ /ERRORPC TSTNUM CPUERR REG/<15><12>  
66\$:  
MOV @#CPUERR,\$TMP0 ;GET CPU ERROR REG  
MOVB \$TSTNM,\$\$TSTNM ;GET TEST NUMBER  
MOV \$ERRPC,-(SP) ;SAVE \$ERRPC FOR TYPEOUT  
;TYPE ERROR PC  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
TYPE ,TWOSP  
MOV \$\$TSTNM,-(SP) ;SAVE \$\$TSTNM FOR TYPEOUT  
;TYPE TEST NUMBER  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
TYPE ,TWOSP  
MOV \$TMP0,-(SP) ;SAVE \$TMP0 FOR TYPEOUT  
;TYPE ERROR REGISTER  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
1\$: CLR @#CPUERR ;CLEAR CPU ERROR REG  
MOV NEXTTST,\$ESCAPE ;SETUP ESCAPE ADDRESS  
ERROR 264 ;MAKE THE ERROR CALL TO SYSMAC  
CACHSPU:MOV (SP),\$ERRPC ;SAVE ERROR PC  
MOV #STACK,SP ;RESTORE STACK  
TYPE ,65\$ ;TYPE ASCIZ STRING  
BR 64\$ ;GET OVER THE ASCIZ  
::65\$: .ASCIZ /UNEXPECTED TRAP TO 114/<15><12>  
64\$:  
BIT #BIT13,@#SWR ;IS SWITCH 13 ON?  
BNE 1\$ ;BRANCH IF YES  
TYPE ,67\$ ;TYPE ASCIZ STRING  
BR 66\$ ;GET OVER THE ASCIZ  
::67\$: .ASCIZ /ERRORPC TSTNUM MEMERR REG/<15><12>  
66\$:  
MOV @#MEMERR,\$TMP0 ;SAVE MEMORY ERROR REG  
MOVB \$TSTNM,\$\$TSTNM ;SAVE TEST NUMBER  
MOV \$ERRPC,-(SP) ;SAVE \$ERRPC FOR TYPEOUT  
;TYPE ERROR PC  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
TYPE ,TWOSP  
MOV \$\$TSTNM,-(SP) ;SAVE \$\$TSTNM FOR TYPEOUT  
;TYPE TEST NUMBER  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
TYPE ,TWOSP  
MOV \$TMP0,-(SP) ;SAVE \$TMP0 FOR TYPEOUT  
;TYPE MEM ERROR REG  
;GO TYPE--OCTAL ASCII(ALL DIGITS)  
1\$: MOV @#MEMERR,@#MEMERR ;CLEAR MEMERR REG.  
MOV NEXTTST,\$ESCAPE ;SETUP ESCAPE ADDRESS

7296 033300 104264  
7297  
7298  
7299  
7300  
7301  
7302  
7303  
7304  
7305  
7306  
7307  
7308  
7309  
7310  
7311  
7312 033302  
7313 033302 006137 177570  
7314 033306 100511  
7315  
7316 033310 000416  
7317  
7318 033312 013746 000004  
7319 033316 012737 033336 000004  
7320 033324 005737 177060  
7321 033330 012637 000004  
7322 033334 000463  
7323 033336 022626  
7324 033340 012637 000004  
7325 033344 000423  
7326 033346  
7327 033346 032737 000400 177570  
7328 033354 001404  
7329 033356 123767 177570 145516  
7330 033364 001462  
7331 033366 105767 145511  
7332 033372 001421  
7333 033374 126767 145515 145501  
7334 033402 101015  
7335 033404 032737 001000 177570  
7336 033412 001404  
7337 033414 016767 145470 145464  
7338 033422 000443  
7339 033424 105067 145453  
7340 033430 005067 145536  
7341 033434 000415  
7342 033436 032737 004000 177570  
7343 033444 001011  
7344 033446 005767 145426  
7345 033452 001406  
7346 033454 005267 145424  
7347 033460 026767 145506 145416  
7348 033466 002021  
7349 033470 012767 000001 145406  
7350 033476 016767 000044 145466  
7351 033504 105267 145372

```
ERROR 264
*****
.SBTTL SCOPE HANDLER ROUTINE
*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW11=1 INHIBIT ITERATIONS
*SW09=1 LOOP ON ERROR
*SW08=1 LOOP ON TEST IN SWR<7:0>
*CALL
* SCOPE ;;SCOPE=IOT

$SCOPE:
ROL @#SWR ;;LOOP ON PRESENT TEST?
BMI $OVER ;;YES IF SW14=1
*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5$,@#ERRVEC ;;SET FOR TIMEOUT
TST @#177060 ;;TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR $SVLAD ;;GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
BR 7$ ;;LOOP ON THE PRESENT TEST
6$: *****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,@#SWR ;;LOOP ON SPEC. TEST?
BEQ 2$ ;;BR IF NO
CMPB @#SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
BEQ $OVER ;;BR IF YES
2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
BEQ 3$ ;;BR IF NO
CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3$ ;;BR IF NO
BIT #BIT09,@#SWR ;;LOOP ON ERROR?
BEQ 4$ ;;BR IF NO
7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
BR $OVER
4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1$ ;;ESCAPE TO THE NEXT TEST
3$: BIT #BIT11,@#SWR ;;INHIBIT ITERATIONS?
BNE 1$ ;;BR IF YES
TST $PASS ;;IF FIRST PASS OF PROGRAM
BEQ 1$ ;; INHIBIT ITERATIONS
INC $ICNT ;;INCREMENT ITERATION COUNT
CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
BGE $OVER ;;BR IF MORE ITERATION REQUIRED
1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
$SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
```

```
7352 033510 011667 145372      MOV      (SP), $LPADR      ;;SAVE SCOPE LOOP ADDRESS
7353 033514 011667 145370      MOV      (SP), $LPERR     ;;SAVE ERROR LOOP ADDRESS
7354 033520 005067 145450      CLR      $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
7355 033524 112767 000001 145363  MOVVB   #1, $ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
7356 033532 016737 145344 177570 $OVER:  MOV      $STNM, @#DISPLAY ;;DISPLAY TEST NUMBER
7357 033540 016716 145342      MOV      $LPADR, (SP)    ;;FUDGE RETURN ADDRESS
7358 033544 000002      RTI                     ;;FIXES PS
7359 033546 003720      $MXCNT: 2000.           ;;MAX. NUMBER OF ITERATIONS
7360      ;;*****
7361
7362      .SBTTL  ERROR HANDLER ROUTINE
7363
7364      ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
7365      ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
7366      ;*AND GO TO ETYPDM ON ERROR
7367      ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7368      ;*SW15=1      HALT ON ERROR
7369      ;*            HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
7370      ;*SW13=1      INHIBIT ERROR TYPEOUTS
7371      ;*SW10=1      BELL ON ERROR
7372      ;*SW09=1      LOOP ON ERROR
7373      ;*CALL
7374      ;*      ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
7375
7376 033550      $ERROR:
7377 033550 116767 145326 145432  MOVVB   $STNM, $$STNM     ;;GET TEST NUMBER FOR TYPE OUT
7378 033556 105267 145321 7$:      INCB   $ERFLG          ;;SET THE ERROR FLAG
7379 033562 001775      BEQ    7$                ;;DON'T LET THE FLAG GO TO ZERO
7380 033564 016737 145312 177570  MOV      $STNM, @#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
7381 033572 005737 177570      TST    @#SWR             ;;HALT ON ERROR = 1?
7382 033576 100001      BPL    8$                ;;BRANCH IF NO
7383 033600 000000      HALT                    ;;YES--HALT
7384 033602 032737 002000 177570 8$:      BIT    #BIT10, @#SWR     ;;BELL ON ERROR?
7385 033610 001402      BEQ    1$                ;;NO - SKIP
7386 033612 104400 001176      TYPE   , $BELL           ;;RING BELL
7387 033616 005267 145270 1$:      INC    $ERTL            ;;COUNT THE NUMBER OF ERRORS
7388 033622 011667 145270      MOV    (SP), $ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
7389 033626 162767 000002 145262  SUB     #2, $ERRPC
7390 033634 117767 145256 145252  MOVVB   @ $ERRPC, $ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
7391 033642 032737 020000 177570  BIT    #BIT13, @#SWR     ;;SKIP TYPEOUT IF SET
7392 033650 001004      BNE    2$                ;;SKIP TYPEOUTS
7393 033652 004767 000056      JSR    PC, ETYPDM        ;;GO TO USER ERROR ROUTINE
7394 033656 104400 001203      TYPE   , $CRLF
7395 033662 005737 177570 2$:      TST    @#SWR             ;;HALT ON ERROR
7396 033666 100001      BPL    9$                ;;SKIP IF CONTINUE
7397 033670 000000      HALT                    ;;HALT ON ERROR!
7398 033672 022767 032604 144142 9$:      CMP    # $ENDAD, 42     ;;ACT-11?
7399 033700 001001      BNE    3$                ;;BRANCH IF NO
7400 033702 000000      HALT                    ;;YES
7401 033704 032737 001000 177570 3$:      BIT    #BIT09, @#SWR    ;;LOOP ON ERROR SWITCH SET?
7402 033712 001402      BEQ    4$                ;;BR IF NO
7403 033714 016716 145170      MOV    $LPERR, (SP)     ;;FUDGE RETURN FOR LOOPING
7404 033720 005767 145250 4$:      TST    $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
7405 033724 001402      BEQ    5$                ;;BR IF NONE
7406 033726 016716 145242      MOV    $ESCAPE, (SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
7407 033732
```

```
7408 033732 000002          RTI                ;;RETURN
7409                      ;:*****
7410                      ;SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
7411                      ;*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
7412                      ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
7413                      ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
7414                      ;:*****
7415 033734 026727 145250 000042 EITPDM: CMP      $$STSTM,#42      ;IS THIS TEST 42?
7416 033742 001464          BEQ      TYP5LE          ;BRANCH IF YES
7417 033744 026727 145240 000070 CMP      $$STSTM,#70      ;IS THIS TEST 70?
7418 033752 001460          BEQ      TYP5LE          ;BRANCH IF YES
7419 033754 104400 001203          TYPE     , $CRLF        ;:'CARRIAGE RETURN' & 'LINE FEED'
7420 033760 005000          CLR      R0              ;;PICKUP THE ITEM INDEX
7421 033762 153700 001114          BISB    @($ITEMB,R0
7422 033766 126727 145122 000377 CMPB    $ITEMB,#377      ;IS MESSAGE NUMBER OVER 377
7423 033774 001006          BNE     1$              ;BRANCH IF NO
7424 033776 016600 000002          MOV     2(SP),R0        ;GET RETURN PC FROM STACK
7425 034002 011000          MOV     (R0),R0        ;GET MESSAGE NUMBER
7426 034004 062766 000002 000002 ADD     #2,2(SP)        ;CORRECT PC
7427 034012 005300          1$: DEC     R0          ;;ADJUST THE INDEX SO THAT IT WILL
7428 034014 010001          MOV     R0,R1
7429 034016 070127 000006          MUL     #6,R1          ;;WORK FOR THE ERROR TABLE
7430 034022 010100          MOV     R1,R0
7431 034024 062700 001274          ADD     #$ERRTB,R0     ;;FORM TABLE POINTER
7432 034030 012067 000004          MOV     (R0)+,2$      ;;PICKUP 'ERROR MESSAGE' POINTER
7433 034034 001404          BEQ     3$              ;:SKIP TYPEOUT IF NO POINTER
7434 034036 104400          TYPE     , $CRLF        ;:TYPE THE 'ERROR MESSAGE'
7435 034040 000000          2$: .WORD 0           ;:'ERROR MESSAGE' POINTER GOES HERE
7436 034042 104400 001203          TYPE     , $CRLF        ;:'CARRIAGE RETURN' & 'LINE FEED'
7437 034046 012067 000004          3$: MOV     (R0)+,4$      ;:PICKUP 'DATA HEADER' POINTER
7438 034052 001404          BEQ     5$              ;:SKIP TYPEOUT IF 0
7439 034054 104400          TYPE     , $CRLF        ;:TYPE THE 'DATA HEADER'
7440 034056 000000          4$: .WORD 0           ;:'DATA HEADER' POINTER GOES HERE
7441 034060 104400 001203          TYPE     , $CRLF        ;:'CARRIAGE RETURN' & 'LINE FEED'
7442 034064 011000          5$: MOV     (R0),R0      ;:PICKUP 'DATA TABLE' POINTER
7443 034066 001003          BNE     7$              ;:GO TYPE THE DATA
7444 034070 104400 001203          6$: TYPE     , $CRLF        ;:'CARRIAGE RETURN' & 'LINE FEED'
7445 034074 000207          RTS     PC              ;:RETURN
7446 034076          7$:
7447 034076 013046          MOV     @(R0)+,-(SP)   ;;SAVE @(R0)+ FOR TYPEOUT
7448 034100 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7449 034102 005710          TST     (R0)          ;;IS THERE ANOTHER NUMBER?
7450 034104 001771          BEQ     6$              ;:BR IF NO
7451 034106 104400 034652          TYPE     ,TWOSP        ;:TYPE TWO(2) SPACES
7452 034112 000771          BR     7$              ;:LOOP
7453                      ;:*****
7454                      ;SBTTL  STACK LIMIT TEST TYPE OUT ROUTINE
7455                      ;* THIS ROUTINE TYPES THE ADDRESS AND STACK LIMIT REGISTER
7456                      ;*VALUES THAT FAILED IN TEST 153 OR 201 IN OCTAL AND BINARY.
7457                      ;:*****
7458 034114 104400          TYP5LE: TYPE          ;:TYPE
7459 034116 055274          EM263          ;:ERROR MESSAGE
7460 034120 104400          TYPE          ;:TYPE
7461 034122 055463          DH263          ;:DATA HEADER
7462 034124 016727 144766          MOV     $ERRPC,(PC)+  ;:GET ERROR PC
7463 034130 000000          1$: .WORD          ;:ERROR PC
```

```

7464 034132 016746 177772      MOV      1$,-(SP)      ;;SAVE 1$ FOR TYPEOUT
7465                                ;;TYPE IT
7466 034136 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7467 034140 104400      TYPE      ;;TYPE TWO SPACES
7468 034142 034652      T+WOSP
7469 034144 016727 145040      MOV      $$STSTM,(PC)+ ;;GET TEST NUMBER
7470 034150 000000      2$:      .WORD      0      ;;TEST NUMBER
7471 034152 016746 177772      MOV      2$,-(SP)      ;;SAVE 2$ FOR TYPEOUT
7472                                ;;TYPE IT
7473 034156 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7474 034160 104400      TYPE      ;;TYPE CR LF
7475 034162 001203      $CRLF
7476
7477 034164 104400      TYPE      ;TYPE DATA HEADER
7478 034166 055510      DH263A
7479 034170 005067 000020      CLR      $TYPEE      ;INITIALIZE $TYPEE
7480
7481 034174 005001      STYPE:    CLR      R1      ;ENSURE R1 CLEAR
7482 034176 005067 000134      CLR      SLDATA
7483 034202 005067 000174      CLR      SPDATA
7484 034206 012700 120000      MOV      #120000,R0    ;GET ERROR DATA BUFFER POINTER
7485 034212 122710      BUFFDP:  CMPB     (PC)+,(R0) ;IS THIS CORRECT TYPE?
7486 034214 000001      $TYPEE:  .WORD      1
7487 034216 001417      BEQ      TYPEHE      ;BRANCH IF YES, GO TYPE IT
7488 034220 062700 000004      ADD      #4,R0      ;STEP R0 TO NEXT ENTRY
7489 034224 020067 144724      NEXTEN:  CMP      R0,$REGO ;LAST ENTRY?
7490 034230 001401      BEQ      2$
7491 034232 000767      BR      BUFFDP      ;BRANCH IF YES
7492 034234 000241      2$:      CLC      ;GO CHECK NEXT ENTRY
7493 034236 062767 000002 177750      ADD      #2,$TYPEE    ;ENSURE C CLEAR
7494 034244 026727 177744 000014      CMP      $TYPEE,#14  ;SELECT NEXT ERROR TYPE
7495 034252 001350      BNE      STYPE      ;IS TYPE ROUTINE DONE?
7496 034254 000467      BR      $$DONE      ;BRANCH IF NO
7497                                ;RETURN
7498 034256 005701      ;OUTPUT THE TYPE HEADER IF FIRST ERROR OF THIS TYPE
7499 034260 001014      TYPEHE:  TST      R1      ;FIRST ERROR OF THIS TYPE?
7500 034262 016701 177726      BNE      4$
7501 034266 062701 056072      MOV      $TYPEE,R1    ;BRANCH IF NO
7502 034272 011167 000006      ADD      $TYPEE,R1    ;GET TYPE NUMBER
7503 034276 104400 034634      MOV      #INDEX,R1   ;GET ADDRESS OF TYPE HEADER ADDRESS
7504 034302 104400      MOV      (R1),3$     ;GET ADDRESS OF HEADER
7505 034304 000000      TYPE     ,SP16      ;TYPE SPACES
7506 034306 104400      3$:      TYPE     ;GO TYPE THE HEADER
7507 034310 001203      $CRLF    ;ADDRESS OF HEADER
7508                                ;TYPE A CRLF
7509 034312 126067 000001 000017      ;OUTPUT THE DATA
7510 034320 001005      4$:      CMPB     1(R0),SLDATA+1 ;IS SL DATA SAME AS LAST SL DATA?
7511 034322 104400      BNE      8$
7512 034324 034612      TYPE     ;BRANCH IF NO
7513 034326 062700 000002      SP38    ;TYPE 38 SPACES
7514 034332 000422      ADD      #2,R0      ;STEP R0 TO SP DATA
7515 034334 012027      BR      SPDATA-2    ;GO TO SP DATA OUTPUT
7516 034336 000000      8$:      MOV      (R0)+,(PC)+ ;GET SL DATA
7517 034340 042767 000377 177770      SLDATA: .WORD
7518 034346 016746 177764      BIC      #377,SLDATA ;MASK HIGH BYTE
7519                                ;;SAVE SLDATA FOR TYPEOUT
                                ;;TYPE IT

```

```
7520 034352 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7521 034354 016746 177756   MOV    SLDATA,-(SP)  ;;PUT DATA ON STACK
7522 034360 004767 000150   JSR    PC,TYPEBN    ;;GO TYPE BINARY
7523 034364 021067 000012   CMP    (R0),SPDATA  ;;IS SP DATA SAME AS LAST SP DATA?
7524 034370 001003          BNE     9$          ;;BRANCH IF NO
7525 034372 062700 000002   ADD    #2,R0        ;;STEP R0 TO NEXT SL DATA
7526 034376 000413          BR     TERM        ;;GO TERMINATE LINE
7527 034400 012027          9$: MOV    (R0)+,(PC)+ ;;GET SP ERROR DATA
7528 034402 000000   SPDATA: .WORD      ;;
7529 034404 104400 034650   TYPE    ,FOURSP    ;;TYPE FOUR SPACES
7530 034410 016746 177766   MOV    SPDATA,-(SP) ;;SAVE SPDATA FOR TYPEOUT
7531          ;;GO TYPE IT IN OCTAL
7532 034414 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7533 034416 016746 177760   MOV    SPDATA,-(SP) ;;PUT DATA ON STACK
7534 034422 004767 000106   JSR    PC,TYPEBN    ;;GO TYPE BINARY
7535 034426 104400          TERM: TYPE      ;;TYPE A CR LF
7536 034430 001203          $CRLF
7537 034432 000674          BR     NEXTEEN    ;;GO CHECK NEXT ENTRY
7538 034434 026727 144514 157774 $$$DONE: CMP    $REGO,#157774 ;;DID BUFFER OVERFLOW?
7539 034442 001033          BNE     1$        ;;BRANCH IF NO
7540 034444 104400 034452   TYPE    ,65$      ;;TYPE ASCIZ STRING
7541 034450 000430          BR     64$        ;;GET OVER THE ASCIZ
7542          ;;65$: .ASCIZ <15><12>/PROBABLY MORE ERRORS BUT BUFFER OVERFLOWED/<15><12>
7543          64$:
7544 034532 000207   1$: RTS    PC      ;;RETURN TO $ERROR
7545
7546 034534 016602 000002   TYPEBN: MOV    2(SP),R2 ;;GET NUMBER TO BE TYPED
7547 034540 104400          TYPE      ;;TYPE FOUR SPACES
7548 034542 034650          FOURSP
7549 034544 012703 000010   MOV    #10,R3      ;;SETUP SOB COUNT
7550 034550 112767 000060 000030 3$: MOVB   #60,BINARY  ;;PUT ASCII 0 IN LOCATION BINARY
7551 034556 006102          ROL     R2        ;;GET BIT TO BE TYPED
7552 034560 103005          BCC    1$        ;;BRANCH IF IT IS 0
7553 034562 005567 000020   ADC    BINARY     ;;MAKE IT ASCII
7554
7555 034566 104400 034606   TYPE    ,BINARY   ;;GO TYPE IT
7556 034572 000402          BR     2$        ;;GO TO END OF LOOP
7557 034574 104400 034651   1$: TYPE    ,THRESP ;;BIT WAS 0 TYPE 3 SPACES
7558 034600 077315          2$: SOB    R3,3$
7559 034602 012616          MOV    (SP)+,(SP) ;;READ JUST STACK
7560 034604 000207          RTS    PC        ;;RETURN
7561 034606 000 040 040 BINARY: .BYTE  0,40,40,0 ;;STORAGE FOR ASCII CHARACTERS & TERMINATOR
7562 034611 000
7563 034612 020040 020040 020040 SP38: .ASCII  / /
7564 034620 020040 020040 020040
7565 034626 020040 020040 020040
7566 034634 020040 020040 020040 SP16: .ASCII  / /
7567 034642 020040 020040 040
7568 034647 040 FIVESP: .ASCII //
7569 034650 040 FOURSP: .ASCII //
7570 034651 040 THRESP: .ASCII //
7571 034652 020040 000 TWOSP: .ASCIZ / /
7572          .EVEN
7573          ;*****
7574          .SBTTL MONITOR RESTORE ROUTINE
7575          ;*THIS ROUTINE IS ENTERED BY TYPING A CHARACTER ON THE KEYBOARD
```



```
7576 ;*IF THE CHARACTER IS NOT A CTRL C EXECUTION IS RETURNED TO THE
7577 ;*TEST FOLLOWING THE ONE THAT WAS INTERRUPTED.
7578 ;*IF IT IS A CTRL C THE MONITOR IS RESTORED AND THE
7579 ;*PROCESSOR HALTS.
7580 034656 005003 QUIT: CLR R3 ;NOT ENTERED THROUGH XXDP CHAIN
7581 034660 017700 144254 MOV @STKB,R0 ;GET CHARACTER
7582 034664 042700 000200 BIC #BIT7,R0 ;GET RID OF PARITY BIT
7583 034670 022700 000003 CMP #3,R0 ;WAS IT A CONTROL C?
7584 034674 001041 BNE DUMMY ;BRANCH IF NO
7585 034676 000005 RESET ;CLEAR THE WORLD
7586 034700 104400 034706 TYPE ,65$ ;TYPE ASCIZ STRING
7587 034704 000403 BR 64$ ;GET OVER THE ASCIZ
7588 ; 65$: .ASCIZ /^C/<15><12>
7589 034714 64$: CHAINQ: MOV #^D1500,R0 ;SETUP SOB COUNT
7590 034714 012700 002734 MOV #SUBTAB+2,R1 ;GET END ADDRESS OF PROGRAM
7591 034720 012701 073312 MOV #160000,R2 ;GET TOP ADDRESS OF MONITOR
7592 034724 012702 160000 1$: MOV (R1)+,-(R2) ;RESTORE THE MONITOR
7593 034730 012142 SOB R0,1$
7594 034732 0770C2 DEC R3
7595 034734 005303 BNE 2$
7596 034736 001001 RTS PC
7597 034740 000207 2$:
7598 034742 TYPE ,65$ ;:TYPE ASCIZ STRING
7599 034742 104400 034750 BR 64$ ;:GET OVER THE ASCIZ
7600 034746 000413 ;:65$: .ASCIZ /MONITOR RESTORED/<15><12><15><12>
7601 64$:
7602 034776 HALT
7603 034776 000000 DUMMY: CLR @STKB ;CLEAR KEYBOARD BUFFER
7604 035000 005077 144134 BISB #BIT6,@STKS ;RESET INTERRUPT ENABLE BIT
7605 035004 152777 000100 144124 TYPE ,65$ ;:TYPE ASCIZ STRING
7606 035012 104400 035020 BR 64$ ;:GET OVER THE ASCIZ
7607 035016 000417 ;:65$: .ASCIZ <15><12>/TYPE A CTRL C TO QUIT!!!!/<15><12>
7608 64$:
7609 035056 JMP @NEXTTST ;RETURN
7610 035056 000177 144206
7611 ;:*****
7612 ;.SBTTL CHECK TEST SEQUENCE ROUTINE
7613 ;*THIS ROUTINE IS CALLED IN THE SCOPE ROUTINE. IT VERIFYS
7614 ;*THAT A TEST HAS NOT BEEN SKIPPED.
7615 035062 016605 000002 SEQENC: MOV 2(SP),R5 ;GET ADDRESS OF SCOPE+2
7616 035066 162705 000002 SUB #2,R5 ;GET ADDRESS OF SCOPE CALL
7617 035072 005004 CLR R4 ;ENSURE R4 CLEAR
7618 035074 116704 144002 MOVB $TSTNM,R4 ;GET TEST NUMBER
7619 035100 006104 ROL R4 ;ADJUST
7620 035102 026405 073136 CMP ADRTAB(R4),R5 ;HAS TEST BEEN SKIPPED?
7621 035106 001523 BEQ 1$ ;BRANCH IF NO
7622 035110 006004 ROR R4
7623 035112 005304 DEC R4
7624 035114 010467 144034 MOV R4,$REGO ;SAVE PREVIOUS TST NUM FOR TYP0UT
7625
7626 ;:FIND OUT WHICH TEST THIS IS.
7627 035120 012767 000002 144034 MOV #2,$TMP0 ;SET BUFFER HEADER POINTER
7628 035126 012767 000176 144030 MOV #176,$TMP1 ;SET BUFFER END POINTER
7629 035134 012703 000100 MOV #100,R3 ;SET R3 AT MIDDLE OF BUFFER
7630 035140 026305 073136 4$: CMP ADRTAB(R3),R5 ;IS IT THIS TEST?
7631 035144 001426 BEQ 2$ ;BRANCH IF YES
```

```
7632 035146 101014          BHI      3$          ;MOVE UP TABLE
7633          ;CORRECT TEST IS FURTHER DOWN TABLE
7634 035150 010367 144006    MOV      R3,$TMP0    ;UPDATE HEADER POINTER
7635 035154 016702 144004    MOV      $TMP1,R2    ;GET END POINTER
7636 035160 166702 143776    SUB      $TMP0,R2    ;FIND RANGE OF REMAINING TABLE
7637 035164 000241          CLC
7638 035166 006002          ROR      R2          ;FIND MIDPOINT OF RANGE
7639 035170 060203          ADD      R2,R3      ;MAKE R3 MID POINT OF REMAINING TABLE
7640 035172 042703 000001    5$:     BIC      #BIT0,R3 ;ENSURE EVEN ADDRESS
7641 035176 000760          BR       4$          ;GO CHECK AGAIN
7642          ;
7643          ;CORRECT TEST IS FURTHER UP THE TABLE
7644 035200 010367 143760    3$:     MOV      R3,$TMP1 ;UPDATE END POINTER
7645 035204 010302          MOV      R3,R2      ;GET END POINTER
7646 035206 166702 143750    SUB      $TMP0,R2    ;FIND RANGE OF REMAINING TABLE
7647 035212 000241          CLC
7648 035214 006002          ROR      R2          ;FIND MID POINT OF RANGE
7649 035216 160203          SUB      R2,R3      ;MAKE R3 MIDPOINT OF REMAINING TABLE
7650 035220 000764          BR       5$          ;MAKE EVEN ADDRESS AND CHECK AGAIN
7651          ;
7652          ;FOUND THE CORRECT TEST
7653 035222 000241          2$:     CLC
7654 035224 006003          ROR      R3          ;GET TEST NUMBER
7655 035226 110367 143650    MOV      R3,$STNM    ;UPDATE TEST NUMBER
7656 035232 010367 143720    MOV      R3,$REG1    ;SAVE FOR TYPEOUT
7657 035236 104400 035244    TYPE    ,65$        ;:TYPE ASCIZ STRING
7658 035242 000404          BR       64$        ;:GET OVER THE ASCIZ
7659          ;:65$: .ASCIZ /TEST /
7660          64$:
7661 035254 016746 143674    MOV      $REG0,-(SP) ;:SAVE $REG0 FOR TYPEOUT
7662 035260 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
7663 035262 104400 035270    TYPE    ,67$        ;:TYPE ASCIZ STRING
7664 035266 000426          BR       66$        ;:GET OVER THE ASCIZ
7665          ;:67$: .ASCIZ / FAILED, CAUSING ECECUTION TO GO TO TEST /
7666          66$:
7667 035344 016746 143606    MOV      $REG1,-(SP) ;:SAVE $REG1 FOR TYPEOUT
7668 035350 104402          TYPOC    ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
7669 035352 104400 001203    TYPE    , $CRLF
7670 035356 000207          1$:     RTS      PC      ;RETURN
7671          ;:*****
7672          ;
7673          .SBTTL TYPE ROUTINE
7674          ;
7675          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
7676          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
7677          ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
7678          ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
7679          ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
7680          ;*
7681          ;*CALL:
7682          ;*1) USING A TRAP INSTRUCTION
7683          ;* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
7684          ;*OR
7685          ;* TYPE
7686          ;* MESADR
7687          ;*
```

```
7688      ;*2) USING A JSR INSTRUCTION
7689      :*      MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
7690      :*      JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
7691      :*      MESADDR      ;;FIRST ADDRESS OF MESSAGE
7692
7693      035360 105767 143565      $TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
7694      035364 100002      BPL      1$      ;;BR IF YES
7695      035366 000000      HALT      ;;HALT HERE IF NO TERMINAL
7696      035370 000407      BR      3$      ;;LEAVE
7697      035372 010046      1$:      MOV      RO,-(SP)      ;;SAVE RO
7698      035374 017600 000002      MOV      @2(SP),RO      ;;GET ADDRESS OF ASCIZ STRING
7699      035400 112046      2$:      MOVB      (RO)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
7700      035402 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
7701      035404 005726      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
7702      035406 012600      MOV      (SP)+,RO      ;;RESTORE RO
7703      035410 062716 000002      3$:      ADD      #2,(SP)      ;;ADJUST RETURN PC
7704      035414 000002      RTI      ;;RETURN
7705      035416 122716 000011      4$:      CMPB      #HT,(SP)      ;;BRANCH IF <HT>
7706      035422 001426      BEQ      8$
7707      035424 122716 000200      CMPB      #CRLF,(SP)      ;;BRANCH IF NOT
7708      035430 001004      BNE      5$
7709      035432 005726      TST      (SP)+      ;;POP <CR><LF> EQUIV
7710      035434 104400 001203      TYPE      $CRLF
7711      035440 000757      BR      2$      ;;GET NEXT CHARACTER
7712      035442 004767 000056      5$:      JSR      PC,$TYPEC      ;;GO TYPE THIS CHARACTER
7713      035446 126726 143476      6$:      CMPB      $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
7714      035452 001352      BNE      2$      ;;IF NO GO GET NEXT CHAR.
7715      035454 016746 143466      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
7716      ;;AND THE NULL CHAR.
7717      035460 105366 000001      7$:      DECB      1(SP)      ;;DOES A NULL NEED TO BE TYPED?
7718      035464 002770      BLT      6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
7719      035466 004767 000032      JSR      PC,$TYPEC      ;;GO TYPE A NULL
7720      035472 105367 000072      DECB      $CHARCNT      ;;DON'T COUNT THE NULL AS A CHARACTER
7721      035476 000770      BR      7$      ;;LOOP
7722
7723      ;;HORIZONTAL TAB PROCESSOR
7724
7725      035500 112716 000040      8$:      MOVB      #' ,(SP)      ;;REPLACE TAB WITH SPACE
7726      035504 004767 000014      9$:      JSR      PC,$TYPEC      ;;TYPE A SPACE
7727      035510 132767 000007 000052      BITB      #7,$CHARCNT      ;;BRANCH IF NOT AT
7728      035516 001372      BNE      9$      ;;TAB STOP
7729      035520 005726      TST      (SP)+      ;;POP SPACE OFF STACK
7730      035522 000726      BR      2$      ;;GET NEXT CHARACTER
7731      035524 105777 143412      $TYPEC:  TSTB      @$TPS      ;;WAIT UNTIL PRINTER IS READY
7732      035530 100375      BPL      $TYPEC
7733      035532 116677 000002 143404      MOVB      2(SP),@$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7734      035540 122766 000015 000002      CMPB      #CR,2(SP)      ;;BRANCH IF
7735      035546 001003      BNE      1$      ;;NOT <CR>
7736      035550 105067 000014      CLRB      $CHARCNT
7737      035554 000406      BR      $TYPEX
7738      035556 122766 000012 000002      1$:      CMPB      #LF,2(SP)      ;;EXIT
7739      035564 001402      BEQ      $TYPEX      ;;BRANCH IF
7740      035566 105227      INCB      (PC)+      ;;<LF>
7741      035570 000000      $CHARCNT: .WORD      0      ;;INC SPACE
7742      035572 000207      $TYPEX:  RTS      PC      ;;COUNT
7743
```

```
7744      ;:*****
7745
7746      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
7747
7748      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
7749      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
7750      ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
7751      ;*CALL:
7752      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7753      ;*      TYPOS      ;;CALL FOR TYPEOUT
7754      ;*      .BYTE  N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
7755      ;*      .BYTE  M      ;;M=1 OR 0
7756      ;*                               ;;1=TYPE LEADING ZEROS
7757      ;*                               ;;0=SUPPRESS LEADING ZEROS
7758
7759      ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
7760      ;*$TYPOS OR $TYPOC
7761      ;*CALL:
7762      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7763      ;*      TYPON      ;;CALL FOR TYPEOUT
7764
7765      ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
7766      ;*CALL:
7767      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7768      ;*      TYPOC      ;;CALL FOR TYPEOUT
7769
7770      035574 017646 000000      $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
7771      035600 116667 000001 000211  MOVB      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
7772      035606 112667 000207      MOVB      (SP)+, $SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
7773      035612 062716 000002      ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
7774      035616 000406      BR      $TYPON
7775      035620 112767 000001 000171  $TYPOC: MOVB      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
7776      035626 112767 000006 000165  MOVB      #6, $SOMODE+1      ;;SET FOR SIX(6) DIGITS
7777      035634 112767 000005 000154  $TYPON: MOVB      #5, $SOCNT      ;;SET THE ITERATION COUNT
7778      035642 010346      MOV      R3,-(SP)      ;;SAVE R3
7779      035644 010446      MOV      R4,-(SP)      ;;SAVE R4
7780      035646 010546      MOV      R5,-(SP)      ;;SAVE R5
7781      035650 116704 000145      MOVB      $SOMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
7782      035654 005404      NEG      R4
7783      035656 062704 000006      ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
7784      035662 110467 000132      MOVB      R4, $SOMODE      ;;SAVE IT FOR USE
7785      035666 116704 000125      MOVB      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
7786      035672 016605 000012      MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
7787      035676 005003      CLR      R3      ;;CLEAR THE OUTPUT WORD
7788      035700 006105      1$: ROL      R5      ;;ROTATE MSB INTO 'C'
7789      035702 000404      BR      3$      ;;GO DO MSB
7790      035704 006105      2$: ROL      R5      ;;FORM THIS DIGIT
7791      035706 006105      ROL      R5
7792      035710 006105      ROL      R5
7793      035712 010503      MOV      R5, R3
7794      035714 006103      3$: ROL      R3      ;;GET LSB OF THIS DIGIT
7795      035716 105367 000076      DECB      $SOMODE      ;;TYPE THIS DIGIT?
7796      035722 100016      BPL      7$      ;;BR IF NO
7797      035724 042703 177770      BIC      #177770, R3      ;;GET RID OF JUNK
7798      035730 001002      BNE      4$      ;;TEST FOR 0
7799      035732 005704      TST      R4      ;;SUPPRESS THIS 0?
```

```
7800 035734 001403          B&Q      5$          ;;BR IF YES
7801 035736 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
7802 035740 052703 000060    BIS      #'0,R3        ;;MAKE THIS DIGIT ASCII
7803 035744 052703 000040    5$: BIS      #' ,R3        ;;MAKE ASCII IF NOT ALREADY
7804 035750 110367 000040    MOVB     R3,8$         ;;SAVE FOR TYPING
7805 035754 104400 036014    TYPE     ,8$          ;;GO TYPE THIS DIGIT
7806 035760 105367 000032    7$: DECB    $OCNT      ;;COUNT BY 1
7807 035764 003347          BGT      2$          ;;BR IF MORE TO GO
7808 035766 002402          BLT      6$          ;;BR IF DONE
7809 035770 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
7810 035772 000744          BR       2$          ;;GO DO THE LAST DIGIT
7811 035774 012605          6$: MOV     (SP)+,R5     ;;RESTORE R5
7812 035776 012604          MOV     (SP)+,R4     ;;RESTORE R4
7813 036000 012603          MOV     (SP)+,R3     ;;RESTORE R3
7814 036002 016666 000002 000004  MOV     2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
7815 036010 012616          MOV     (SP)+,(SP)
7816 036012 000002          RTI
7817 036014          000          ;;RETURN
7818 036015          000          ;;STORAGE FOR ASCII DIGIT
7819 036016          000          ;;TERMINATOR FOR TYPE ROUTINE
7820 036017          000          ;;OCTAL DIGIT COUNTER
7821 036020 000000          $OCNT: .BYTE 0        ;;ZERO FILL SWITCH
7822          $OFILL: .BYTE 0    ;;NUMBER OF DIGITS TO TYPE
7823          $OMODE: .WORD 0
7824          ;;*****
7825          .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7826          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7827          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7828          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7829          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7830          ;*REPLACED WITH SPACES.
7831          ;*CALL:
7832          ;*   MOV     NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
7833          ;*   TYPDS          ;;GO TO THE ROUTINE
7834
7835          $TYPDS:
7836          MOV     R0,-(SP)          ;;PUSH R0 ON STACK
7837          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
7838          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
7839          MOV     R3,-(SP)          ;;PUSH R3 ON STACK
7840          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
7841          MOV     #20200,-(SP)     ;;SET BLANK SWITCH AND SIGN
7842          MOV     20(SP),R5        ;;GET THE INPUT NUMBER
7843          BPL     1$              ;;BR IF INPUT IS POS.
7844          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
7845          MOVB   #'-,1(SP)        ;;MAKE THE ASCII NUMBER NEG.
7846          036056 005000          1$: CLR     R0              ;;ZERO THE CONSTANTS INDEX
7847          036060 012703 036236    MOV     #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
7848          036064 112723 000040    MOVB   #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
7849          036070 005002          2$: CLR     R2              ;;CLEAR THE BCD NUMBER
7850          036072 016001 036226    MOV     $DTBL(R0),R1  ;;GET THE CONSTANT
7851          036076 160105          3$: SUB     R1,R5        ;;FORM THIS BCD DIGIT
7852          036100 002402          BLT     4$              ;;BR IF DONE
7853          036102 005202          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
7854          036104 000774          BR      3$
7855          036106 060105          4$: ADD     R1,R5        ;;ADD BACK THE CONSTANT
```

```
7856 036110 005702          TST      R2          ;;CHECK IF BCD DIGIT=0
7857 036112 001002          BNE      5$          ;;FALL THROUGH IF 0
7858 036114 105716          TSTB     (SP)        ;;STILL DOING LEADING 0'S?
7859 036116 100407          BMI      7$          ;;BR IF YES
7860 036120 106316          5$: ASLB     (SP)        ;;MSD?
7861 036122 103003          BCC      6$          ;;BR IF NO
7862 036124 116663 000001 177777 MOVVB    1(SP),-1(R3) ;;YES--SET THE SIGN
7863 036132 052702 000060 6$: BIS     #'0,R2     ;;MAKE THE BCD DIGIT ASCII
7864 036136 052702 000040 7$: BIS     #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
7865 036142 110223          MOVVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
7866 036144 005720          TST     (R0)+        ;;JUST INCREMENTING
7867 036146 020027 000010          CMP     R0,#10      ;;CHECK THE TABLE INDEX
7868 036152 002746          BLT     2$          ;;GO DO THE NEXT DIGIT
7869 036154 003002          BGT     8$          ;;GO TO EXIT
7870 036156 010502          MOV     R5,R2        ;;GET THE LSD
7871 036160 000764          BR      6$          ;;GO CHANGE TO ASCII
7872 036162 105726          8$: TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
7873 036164 100003          BPL     9$          ;;BR IF NO
7874 036166 116663 177777 177776 MOVVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
7875 036174 105013          9$: CLRB   (R3)        ;;SET THE TERMINATOR
7876 036176 012605          MOV     (SP)+,R5     ;;POP STACK INTO R5
7877 036200 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
7878 036202 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
7879 036204 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
7880 036206 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
7881 036210 104400 036236          TYPE    $DBLK        ;;NOW TYPE THE NUMBER
7882 036214 016666 000002 000004 MOV     2(SP),4(SP)   ;;ADJUST THE STACK
7883 036222 012616          MOV     (SP)+,(SP)
7884 036224 000002          RTI                          ;;RETURN TO USER
7885 036226 023420          $DTBL: 10000.
7886 036230 001750          1000.
7887 036232 000144          100.
7888 036234 000012          10.
7889 036236 000004          $DBLK: .BLKW 4
7890
7891
7892
7893          .SBTTL TRAP DECODER
7894
7895          ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
7896          ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7897          ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7898          ;*GO TO THAT ROUTINE.
7899 036246 010046          $TRAP: MOV     R0,-(SP) ;;SAVE R0
7900 036250 016600 000002          MOV     2(SP),R0     ;;GET TRAP ADDRESS
7901 036254 005740          TST     -(R0)        ;;BACKUP BY 2
7902 036256 111000          MOVVB   (R0),R0     ;;GET RIGHT BYTE OF TRAP
7903 036260 016000 036266          MOV     $TRPAD(R0),R0 ;;INDEX TO TABLE
7904 036264 000200          RTS     R0          ;;GO TO ROUTINE
7905
7906
7907          .SBTTL TRAP TABLE
7908
7909          ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
7910          ;*BY THE 'TRAP' INSTRUCTION.
7911
```

7912  
7913  
7914 036266  
7915 036266 035360  
7916 036270 035620  
7917 036272 035574  
7918 036274 035634  
7919 036276 036022  
7920  
7921  
7922  
7923  
7924  
7925 036300 012737 036422 000024  
7926 036306 012737 000340 000026  
7927 036314 010046  
7928 036316 010146  
7929 036320 010246  
7930 036322 010346  
7931 036324 010446  
7932 036326 010546  
7933 036330 010667 000072  
7934 036334 012737 036346 000024  
7935 036342 000000  
7936 036344 000776  
7937  
7938  
7939 036346 016706 000054  
7940 036352 005067 000050  
7941 036356 005267 000044  
7942 036362 001375  
7943 036364 012605  
7944 036366 012604  
7945 036370 012603  
7946 036372 012602  
7947 036374 012601  
7948 036376 012600  
7949 036400 012737 036300 000024  
7950 036406 012737 000340 000026  
7951 036414 104400  
7952 036416 036430  
7953 036420 000002  
7954 036422 000000  
7955 036424 000776  
7956 036426 000000  
7957 036430 005015 047520 042527  
7958 036436 000122  
7959  
7960 036440 041600 052520 052440  
7961 036446 042116 051105 052040  
7962 036454 051505 020124 047506  
7963 036462 047125 020104 047524  
7964 036470 041040 020105 020101  
7965 036476 000  
7966 036477 113 030502 026461  
7967 036504 027502 020103 051117

```
ROUTINE
-----
$TRPAD:
$TYPE      ::CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
$TYPOC     ::CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS     ::CALL=TYPOS     TRAP+4(104404)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON     ::CALL=TYPON     TRAP+6(104406)  TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS     ::CALL=TYPDS     TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
:*****
.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE
$PWRDN: MOV    # $ILLUP, @#PWRVEC  ::SET FOR FAST UP
        MOV    #340, @#PWRVEC+2  ::PRIO:7
        MOV    R0, -(SP)          ::PUSH R0 ON STACK
        MOV    R1, -(SP)          ::PUSH R1 ON STACK
        MOV    R2, -(SP)          ::PUSH R2 ON STACK
        MOV    R3, -(SP)          ::PUSH R3 ON STACK
        MOV    R4, -(SP)          ::PUSH R4 ON STACK
        MOV    R5, -(SP)          ::PUSH R5 ON STACK
        MOV    SP, $SAVR6         ::SAVE SP
        MOV    # $PWRUP, @#PWRVEC ::SET UP VECTOR
        HALT
        BR     .-2                ::HANG UP

:POWER UP ROUTINE
$PWRUP: MOV    $SAVR6, SP         ::GET SP
        CLR    $SAVR6            ::WAIT LOOP FOR THE TTY
1$:      INC    $SAVR6            ::WAIT FOR THE INC
        BNE    1$                ::OF WORD
        MOV    (SP)+, R5          ::POP STACK INTO R5
        MOV    (SP)+, R4          ::POP STACK INTO R4
        MOV    (SP)+, R3          ::POP STACK INTO R3
        MOV    (SP)+, R2          ::POP STACK INTO R2
        MOV    (SP)+, R1          ::POP STACK INTO R1
        MOV    (SP)+, R0          ::POP STACK INTO R0
        MOV    # $PWRDN, @#PWRVEC ::SET UP THE POWER DOWN VECTOR
        MOV    #340, @#PWRVEC+2  ::PRIO:7
        TYPE   $POWER            ::REPORT THE POWER FAILURE
        $PWRMG: .WORD $POWER     ::POWER FAIL MESSAGE POINTER
        RTI
$ILLUP: HALT                    ::THE POWER UP SEQUENCE WAS STARTED
        BR     .-2                ::BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0                        ::PUT THE SP HERE
$POWER: .ASCIZ <15><12>'POWER'

.EVEN
MSG1: .ASCIZ<CRLF> 'CPU UNDER TEST FOUND TO BE A '

MSG3: .ASCIZ 'KB11-B/C OR KB11-CM' '<CRLF>
```

7968	036512	045440	030502	026461		
7969	036520	046503	020040	020040		
7970	036526	020040	020040	020040		
7971	036534	020040	020040	020040		
7972	036542	000200				
7973	036544	020040	000040		MSG5:	.ASCIZ '' ''
7974	036550	047200	052117	035105	MSG6:	.ASCIZ <CRLF>'NOTE:''
7975	036556	000				
7976		036560				.EVEN
7977						
7978	036560	050123	020114	040506	EM1:	.ASCIZ /SPL FAILED OR PSW PRIORITY BITS STUCK/
7979	036566	046111	042105	047440		
7980	036574	020122	051520	020127		
7981	036602	051120	047511	044522		
7982	036610	054524	041040	052111		
7983	036616	020123	052123	041525		
7984	036624	000113				
7985	036626	051105	047522	020122	DH1:	.ASCII /ERROR PC SPL 5 PSW SPL 2 PSW TST NUM/<CRLF>
7986	036634	041520	020040	051440		
7987	036642	046120	032440	050040		
7988	036650	053523	020040	020040		
7989	036656	020040	051440	046120		
7990	036664	031040	050040	053523		
7991	036672	020040	052040	052123		
7992	036700	047040	046525	200		
7993	036705	011	042440	050130		.ASCIZ / EXPECT ACTUAL EXPECT ACTUAL/
7994	036712	041505	020124	041501		
7995	036720	052524	046101	020040		
7996	036726	042440	050130	041505		
7997	036734	020124	041501	052524		
7998	036742	046101	000			
7999		036746				.EVEN
8000	036746	001116	001214	001206	DT1:	.WORD \$ERRPC,\$SPR5,\$ERPSW,\$SPR2,\$TMP0,\$STSTNM,0
8001	036754	001212	001162	001210		
8002	036762	000000				
8003	036764	051120	032440	047440	EM2:	.ASCIZ /PR 5 OK PR 2 BAD/
8004	036772	020113	051120	031040		
8005	037000	041040	042101	000		
8006	037005	120	020122	020062	EM3:	.ASCIZ /PR 2 OK BUT PR 5 BAD/
8007	037012	045517	041040	052125		
8008	037020	050040	020122	020065		
8009	037026	040502	000104			
8010	037032	040522	043103	042440	EM4:	.ASCIZ /RACF E3 BAD(NOT GOING HIGH ON SPL)/
8011	037040	020063	040502	024104		
8012	037046	047516	020124	047507		
8013	037054	047111	020107	044510		
8014	037062	044107	047440	020116		
8015	037070	050123	024514	000		
8016	037075	105	051122	051117	DH4:	.ASCIZ /ERRORPC TEST NUMBER/
8017	037102	041520	052040	051505		
8018	037110	020124	052516	041115		
8019	037116	051105	000			
8020		037122				.EVEN
8021	037122	001116	001210	000000	DT4:	.WORD \$ERRPC,\$STSTNM,0
8022	037130	040522	043103	042440	EM5:	.ASCIZ /RACF E5 BAD(AFIR04(1)*AFIR03(1)*AFIR05(1)*(RTS:CCOP))/
8023	037136	020065	040502	024104		



8024	037144	043101	051111	032060					
8025	037152	030450	025051	043101					
8026	037160	051111	031460	030450					
8027	037166	025051	043101	051111					
8028	037174	032460	030450	025051					
8029	037202	051050	051524	041472					
8030	037210	047503	024520	000051					
8031	037216	042522	042523	020124	EM6:	.ASCIZ	/RESET DIDN'T SEND OUT INIT/		
8032	037224	044504	047104	052047					
8033	037232	051440	047105	020104					
8034	037240	052517	020124	047111					
8035	037246	052111	000						
8036	037251	122	041501	020106	EM7:	.ASCIZ	/RACF E5 BAD/		
8037	037256	032505	041040	042101					
8038	037264	000							
8039	037265	122	041501	020106	EM10:	.ASCIZ	/RACF E8 BAD/		
8040	037272	034105	041040	042101					
8041	037300	000							
8042	037301	122	041501	020106	EM11:	.ASCIZ	'RACF E17 BAD(AFIR07(0)*X/CLASS''		
8043	037306	030505	020067	040502					
8044	037314	024104	043101	051111					
8045	037322	033460	030050	025051					
8046	037330	027530	046103	051501					
8047	037336	000123							
8048	037340	040522	043103	042440	EM12:	.ASCIZ	'RACF E5 BAD(AFIR06(0)*X/CLASS)''		
8049	037346	020065	040502	024104					
8050	037354	043101	051111	033060					
8051	037362	030050	025051	027530					
8052	037370	046103	051501	024523					
8053	037376	000							
8054	037377	120	041103	042040	EM13:	.ASCIZ	/PCB DIDN'T LOAD FROM R5/		
8055	037404	042111	023516	020124					
8056	037412	047514	042101	043040					
8057	037420	047522	020115	032522					
8058	037426	000							
8059	037427	123	020120	044504	EM14:	.ASCIZ	/SP DIDN'T LOAD PROPERLY/		
8060	037434	047104	052047	046040					
8061	037442	040517	020104	051120					
8062	037450	050117	051105	054514					
8063	037456	000							
8064	037457	105	051122	051117	DH14:	.ASCII	/ERRORPC SP TST NUM/<CRLF>		
8065	037464	041520	020040	020040					
8066	037472	020040	051440	004520					
8067	037500	052040	052123	047040					
8068	037506	046525	200						
8069	037511	040	020040	020040		.ASCIZ	/ EXPECT ACTUAL/		
8070	037516	020040	042440	050130					
8071	037524	041505	020124	040440					
8072	037532	052103	040525	000114					
8073						.EVEN			
8074	037540	001116	001156	001154	DT14:	.WORD	\$ERRPC,\$REG1,\$REG0,\$\$STNM,0		
8075	037546	001210	000000						
8076	037552	032522	042040	042111	EM15:	.ASCIZ	/R5 DIDN'T LOAD PROPERLY/		
8077	037560	023516	020124	047514					
8078	037566	042101	050040	047522					
8079	037574	042520	046122	000131					

8080	037602	051105	047522	050122	DH15:	.ASCII	/ERRORPC	R5	TST NUM/<CRLF>
8081	037610	020103	020040	020040					
8082	037616	020040	032522	020040					
8083	037624	020040	020040	020040					
8084	037632	051524	020124	052516					
8085	037640	100115							
8086	037642	020011	054105	042520		.ASCIZ	/	EXPECT	ACTUAL/
8087	037650	052103	020040	041501					
8088	037656	052524	046101	000					
8089	037663	122	041501	020106	EM16:	.ASCIZ	'RACF X/CLASS DIDN'T GO HIGH'		
8090	037670	027530	046103	051501					
8091	037676	020123	044504	047104					
8092	037704	052047	043440	020117					
8093	037712	044510	044107	000					
8094	037717	122	041501	020106	EM17:	.ASCIZ	/RACF E34 BAD OR NOT GETTING THRU RACL RADR05/		
8095	037724	031505	020064	040502					
8096	037732	020104	051117	047040					
8097	037740	052117	043440	052105					
8098	037746	044524	043516	052040					
8099	037754	051110	020125	040522					
8100	037762	046103	051040	042101					
8101	037770	030122	000065						
8102	037774	051107	045101	051440	EM20:	.ASCIZ	/GRAJ SC05 L DOESN'T GET THRU TO RACK BRCAB04 L AS A HIGH/		
8103	040002	030103	020065	020114					
8104	040010	047504	051505	023516					
8105	040016	020124	042507	020124					
8106	040024	044124	052522	052040					
8107	040032	020117	040522	045503					
8108	040040	041040	041522	041101					
8109	040046	032060	046040	040440					
8110	040054	020123	020101	044510					
8111	040062	044107	000						
8112	040065	122	020061	044504	EM21:	.ASCIZ	/R1 DIDN'T SHIFT/		
8113	040072	047104	052047	051440					
8114	040100	044510	052106	000					
8115	040105	122	020061	044123	EM22:	.ASCII	/R1 SHIFTED BUT CARRY DIDN'T SET/<CRLF>		
8116	040112	043111	042524	020104					
8117	040120	052502	020124	040503					
8118	040126	051122	020131	044504					
8119	040134	047104	052047	051440					
8120	040142	052105	200						
8121	040145	123	044510	052106		.ASCIZ	/SHIFT COUNTER COULD BE STUCK/		
8122	040152	041440	052517	052116					
8123	040160	051105	041440	052517					
8124	040166	042114	041040	020105					
8125	040174	052123	041525	000113					
8126	040202	051107	045101	051440	EM23:	.ASCIZ	/GRAJ SC05 L DOESN'T GET THRU TO RACK BRCAB04 L AS A LOW/		
8127	040210	030103	020065	020114					
8128	040216	047504	051505	023516					
8129	040224	020124	042507	020124					
8130	040232	044124	052522	052040					
8131	040240	020117	040522	045503					
8132	040246	041040	041522	041101					
8133	040254	032060	046040	040440					
8134	040262	020123	020101	047514					
8135	040270	000127							

8136	040272	051501	020110	044522	EM24:	.ASCIZ /ASH RIGHT DIDN'T SIGN FILL/
8137	040300	044107	020124	044504		
8138	040306	047104	052047	051440		
8139	040314	043511	020116	044506		
8140	040322	046114	000			
8141	040325	122	020061	044504	EM25:	.ASCIZ /R1 DIDN'T SHIFT CORRECTLY/
8142	040332	047104	052047	051440		
8143	040340	044510	052106	041440		
8144	040346	051117	042522	052103		
8145	040354	054514	000			
8146	040357	105	051122	051117	DH25:	.ASCII- /ERRORPC R1 TST NUM/<CRLF>
8147	040364	041520	020040	020040		
8148	040372	020040	051040	004461		
8149	040400	052040	052123	047040		
8150	040406	046525	200			
8151	040411	011	042440	050130		.ASCIZ / EXPECT ACTUAL/
8152	040416	041505	020124	040440		
8153	040424	052103	040525	000114		
8154						
8155	040432	001116	001164	001156	DT25:	.EVEN \$ERRPC,\$TMP1,\$REG1,\$\$TSTNM,0
8156	040440	001210	000000			
8157	040444	030522	051440	044510	EM26:	.ASCIZ /R1 SHIFTED BUT CARRY DIDN'T SET/
8158	040452	052106	042105	041040		
8159	040460	052125	041440	051101		
8160	040466	054522	042040	042111		
8161	040474	023516	020124	042523		
8162	040502	000124				
8163	040504	051501	027110	030062	EM27:	.ASCIZ /ASH.20 DIDN'T LOAD CC'S CORRECTLY/
8164	040512	042040	042111	023516		
8165	040520	020124	047514	042101		
8166	040526	041440	023503	020123		
8167	040534	047503	051122	041505		
8168	040542	046124	000131			
8169	040546	030522	051440	044510	EM30:	.ASCIZ /R1 SHIFTED WHEN SHIFT COUNT=0/
8170	040554	052106	042105	053440		
8171	040562	042510	020116	044123		
8172	040570	043111	020124	047503		
8173	040576	047125	036524	000060		
8174	040604	051501	027110	030064	EM31:	.ASCIZ /ASH.40 DIDN'T LOAD CC'S CORRECTLY/
8175	040612	042040	042111	023516		
8176	040620	020124	047514	042101		
8177	040626	041440	023503	020123		
8178	040634	047503	051122	041505		
8179	040642	046124	000131			
8180	040646	040522	043103	052440	EM32:	.ASCIZ 'RACF U/CLASS DOESN'T GO HIGH ON ASH''
8181	040654	041457	040514	051523		
8182	040662	042040	042517	047123		
8183	040670	052047	043440	020117		
8184	040676	044510	044107	047440		
8185	040704	020116	051501	000110		
8186	040712	051501	027110	030060	EM33:	.ASCIZ /ASH.00 FAILED/
8187	040720	043040	044501	042514		
8188	040726	000104				
8189	040730	051111	041103	042440	EM34:	.ASCIZ /IRCB E35(B2) NOT GOING LOW/
8190	040736	032463	041050	024462		
8191	040744	047040	052117	043440		

8192	040752	044517	043516	046040	
8193	040760	053517	000		
8194	040763	111	041522	020102	EM35: .ASCII /IRCB (MUL:ASHC)+MFP L DIDN'T GO LOW OR IRCB E37 BAD/<CRLF>
8195	040770	046450	046125	040472	
8196	040776	044123	024503	046453	
8197	041004	050106	046040	042040	
8198	041012	042111	023516	020124	
8199	041020	047507	046040	053517	
8200	041026	047440	020122	051111	
8201	041034	041103	042440	033463	
8202	041042	041040	042101	200	
8203	041047	117	020122	051111	.ASCIZ /OR IRCB E35(B1) STUCK LOW/
8204	041054	041103	042440	032463	
8205	041062	041050	024461	051440	
8206	041070	052524	045503	046040	
8207	041076	053517	000		
8208	041101	122	041501	020105	EM36: .ASCIZ /RACE (MUL:ASHC+MFP) DIDN'T GO HIGH OR RACE E44 BAD/
8209	041106	046450	046125	040472	
8210	041114	044123	025503	043115	
8211	041122	024520	042040	042111	
8212	041130	023516	020124	047507	
8213	041136	044040	043511	020110	
8214	041144	051117	051040	041501	
8215	041152	020105	032105	020064	
8216	041160	040502	000104		
8217	041164	040522	042503	042440	EM37: .ASCIZ /RACE E45 BAD (AFIRO4(1)*(MUL:ASHC+MFP))/
8218	041172	032464	041040	042101	
8219	041200	024040	043101	051111	
8220	041206	032060	030450	025051	
8221	041214	046450	046125	040472	
8222	041222	044123	025503	043115	
8223	041230	024520	000051		
8224	041234	040522	042503	042440	EM40: .ASCIZ /RACE E33 BAD (AFIRO5(1)*(MUL:ASHC+MFP))/
8225	041242	031463	041040	042101	
8226	041250	024040	043101	051111	
8227	041256	032460	030450	025051	
8228	041264	046450	046125	040472	
8229	041272	044123	025503	043115	
8230	041300	024520	000051		
8231	041304	030122	042040	042111	EM41: .ASCIZ /RO DIDN'T SIGN FILL ON RIGHT SHIFT/
8232	041312	023516	020124	044523	
8233	041320	047107	043040	046111	
8234	041326	020114	047117	051040	
8235	041334	043511	052110	051440	
8236	041342	044510	052106	000	
8237	041347	105	051122	051117	DH41: .ASCII /ERRORPC RO TST NUM/<CRLF>
8238	041354	041520	020040	020040	
8239	041362	020040	020040	030122	
8240	041370	020011	052040	052123	
8241	041376	047040	046525	200	
8242	041403	011	020040	054105	.ASCIZ / EXPECT ACTUAL/
8243	041410	042520	052103	020040	
8244	041416	041501	052524	046101	
8245	041424	000			
8246		041426			
8247	041426	001116	001162	001154	DT41: .EVEN \$ERRPC,\$TMPO,\$REGO,\$\$TSTNM,0

8248	041434	001210	000000						
8249	041440	040502	020104	041503	EM42:	.ASCIZ	/BAD CC'S ON RIGHT SHIFT/		
8250	041446	051447	047440	020116					
8251	041454	044522	044107	020124					
8252	041462	044123	043111	000124					
8253	041470	051105	047522	050122	DH42:	.ASCII	/ERRORPC	PSW	TST NUM/<CRLF>
8254	041476	020103	020040	020040					
8255	041504	020040	051520	004527					
8256	041512	020040	051524	020124					
8257	041520	052516	100115						
8258	041524	020011	054105	042520		.ASCIZ	/	EXPECT ACTUAL/	
8259	041532	052103	020040	041501					
8260	041540	052524	046101	000					
8261		041546				.EVEN			
8262	041546	001116	001162	001206	DT42:	.WORD	\$ERRPC,\$TMPO,\$ERPSW,\$STSTNM,0		
8263	041554	001210	000000						
8264	041560	030122	030074	020076	EM43:	.ASCIZ	/R0<0> DIDN'T GO TO R1<15>/		
8265	041566	044504	047104	052047					
8266	041574	043440	020117	047524					
8267	041602	051040	036061	032461					
8268	041610	000076							
8269	041612	051105	047522	050122	DH43:	.ASCII	/ERRORPC	R0	R1 TST NUM/<CRLF>
8270	041620	020103	020040	020040					
8271	041626	020040	030122	020011					
8272	041634	020040	020040	020040					
8273	041642	051040	004461	020040					
8274	041650	051524	020124	052516					
8275	041656	100115							
8276	041660	020011	054105	042520		.ASCIZ	/	EXPECT ACTUAL	EXPECT ACTUAL/
8277	041666	052103	020040	041501					
8278	041674	052524	046101	020040					
8279	041702	042440	050130	041505					
8280	041710	020124	040440	052103					
8281	041716	040525	000114						
8282						.EVEN			
8283	041722	001116	001162	001154	DT43:	.WORD	\$ERRPC,\$TMPO,\$REG0,\$TMP1,\$REG1,\$STSTNM,0		
8284	041730	001164	001156	001210					
8285	041736	000000							
8286	041740	030122	042040	042111	EM44:	.ASCIZ	/R0 DIDN'T GET SHIFTED LEFT PROPERLY/		
8287	041746	023516	020124	042507					
8288	041754	020124	044123	043111					
8289	041762	042524	020104	042514					
8290	041770	052106	050040	047522					
8291	041776	042520	046122	000131					
8292	042004	040502	020104	041503	EM45:	.ASCIZ	/BAD CC'S ON LEFT SHIFT/		
8293	042012	051447	047440	020116					
8294	042020	042514	052106	051440					
8295	042026	044510	052106	000					
8296	042033	115	050106	020124	EM46:	.ASCIZ	/MFPT LOADED R0 INCORRECTLY/		
8297	042040	047514	042101	042105					
8298	042046	051040	020060	047111					
8299	042054	047503	051122	041505					
8300	042062	046124	000131						
8301	042066	051105	047522	050122	DH46:	.ASCII	/ERRORPC	R0	TST NUM/<CRLF>
8302	042074	020103	020040	020040					
8303	042102	020040	030122	020011					



8360	042556	047507	047111	020107	
8361	042564	047514	000127		
8362	042570	051501	027103	030060	EM54: .ASCIZ /ASC.00 FAILED/
8363	042576	043040	044501	042514	
8364	042604	000104			
8365	042606	024122	052515	035114	EM55: .ASCIZ /R(MUL:ASHC+MFP) FIELD IN INSTR DECODE ROM BAD/
8366	042614	051501	041510	046453	
8367	042622	050106	020051	044506	
8368	042630	046105	020104	047111	
8369	042636	044440	051516	051124	
8370	042644	042040	041505	042117	
8371	042652	020105	047522	020115	
8372	042660	040502	000104		
8373	042664	051107	042101	042040	EM56: .ASCIZ /GRAD DROO STUCK HIGH OR RACK E64(C1) BAD/
8374	042672	030122	020060	052123	
8375	042700	041525	020113	044510	
8376	042706	044107	047440	020122	
8377	042714	040522	045503	042440	
8378	042722	032066	041450	024461	
8379	042730	041040	042101	000	

8380	042735	107	040522	020104	EM57:	.ASCIZ /GRAD DROO STUCK LOW OR RACK E64(C1) BAD/
8381	042742	051104	030060	051440		
8382	042750	052524	045503	046040		
8383	042756	053517	047440	020122		
8384	042764	040522	045503	042440		
8385	042772	032066	041450	024461		
8386	043000	041040	042101	000		
8387	043005	107	040522	020112	EM60:	.ASCII /GRAJ SC=0 L NOT GETTING TO RACK E50(C1) OR/<CRLF>
8388	043012	041523	030075	046040		
8389	043020	047040	052117	043440		
8390	043026	052105	044524	043516		
8391	043034	052040	020117	040522		
8392	043042	045503	042440	030065		
8393	043050	041450	024461	047440		
8394	043056	100122				
8395	043060	042440	030065	041040		.ASCIZ / E50 BAD (FAILED LOW)/
8396	043066	042101	024040	040506		
8397	043074	046111	042105	046040		
8398	043102	053517	000051			
8399	043106	030122	047440	020122	EM61:	.ASCIZ /R0 OR R1 OR BOTH BAD ON POSITIVE MULTIPLICAND/
8400	043114	030522	047440	020122		
8401	043122	047502	044124	041040		
8402	043130	042101	047440	020116		
8403	043136	047520	044523	044524		
8404	043144	042526	046440	046125		
8405	043152	044524	046120	041511		
8406	043160	047101	000104			
8407	043164	040502	020104	041503	EM62:	.ASCIZ /BAD CC'S ON POSITIVE MULTIPLICAND/
8408	043172	051447	047440	020116		
8409	043200	047520	044523	044524		
8410	043206	042526	046440	046125		
8411	043214	044524	046120	041511		
8412	043222	047101	000104			
8413	043226	030122	041040	042101	EM63:	.ASCIZ /RC BAD ON NEGATIVE MULTIPLICAND/
8414	043234	047440	020116	042516		
8415	043242	040507	044524	042526		
8416	043250	046440	046125	044524		
8417	043256	046120	041511	047101		
8418	043264	000104				
8419	043266	030522	041040	042101	EM64:	.ASCIZ /R1 BAD ON NEGATIVE MULTIPLICAND/
8420	043274	047440	020116	042516		
8421	043302	040507	044524	042526		
8422	043310	046440	046125	044524		
8423	043316	046120	041511	047101		
8424	043324	000104				
8425	043326	040502	020104	041503	EM65:	.ASCIZ /BAD CC'S DUE TO STATE MUL.50/
8426	043334	051447	042040	042525		
8427	043342	052040	020117	052123		
8428	043350	052101	020105	052515		
8429	043356	027114	030065	000		
8430	043363	103	042040	042111	EM66:	.ASCIZ /C DIDN'T SET ON OVERFLOW/
8431	043370	023516	020124	042523		
8432	043376	020124	047117	047440		
8433	043404	042526	043122	047514		
8434	043412	000127				
8435	043414	020103	044504	047104	EM67:	.ASCIZ /C DIDN'T SET ON UNDERFLOW/



8436	043422	052047	051440	052105		
8437	043430	047440	020116	047125		
8438	043436	042504	043122	047514		
8439	043444	000127				
8440	043446	052515	027114	030060	EM70:	.ASCIZ /MUL.00 FAILED/
8441	043454	043040	044501	042514		
8442	043462	000104				
8443	043464	051501	027110	030063	EM72:	.ASCIZ /ASH.30 DIDN'T LOAD CC'S CORRECTLY/
8444	043472	042040	042111	023516		
8445	043500	020124	047514	042101		
8446	043506	041440	023503	020123		
8447	043514	047503	051122	041505		
8448	043522	046124	000131			
8449	043526	051501	027110	030464	EM73:	.ASCIZ /ASH.41 FAILED/
8450	043534	043040	044501	042514		
8451	043542	000104				
8452	043544	051501	027110	030464	EM74:	.ASCIZ /ASH.41 DIDN'T LOAD CC'S CORRECTLY/
8453	043552	042040	042111	023516		
8454	043560	020124	047514	042101		
8455	043566	041440	023503	020123		
8456	043574	047503	051122	041505		
8457	043602	046124	000131			
8458	043606	051111	043103	055040	EM75:	.ASCIZ /IRCF 22(1) DOESN'T GO HIGH OR RACK E49(B1) STUCK LOW/
8459	043614	024062	024461	042040		
8460	043622	042517	047123	052047		
8461	043630	043440	020117	044510		
8462	043636	044107	047440	020122		
8463	043644	040522	045503	042440		
8464	043652	034464	041050	024461		
8465	043660	051440	052524	045503		
8466	043666	046040	053517	000		
8467	043673	103	023503	020123	EM76:	.ASCIZ /CC'S BAD IN DVE.00/
8468	043700	040502	020104	047111		
8469	043706	042040	042526	030056		
8470	043714	000060				
8471	043716	041503	051447	041040	EM77:	.ASCIZ /CC'S BAD IN DVC.70/
8472	043724	042101	044440	020116		
8473	043732	053104	027103	030067		
8474	043740	000				
8475	043741	122	041501	020113	EM100:	.ASCIZ /RACK E50(B0) STUCK HIGH/
8476	043746	032505	024060	030102		
8477	043754	020051	052123	041525		
8478	043762	020113	044510	044107		
8479	043770	000				
8480						
8481	043771	107	040522	020112	EM101:	.ASCIZ /GRAJ DIV SUB L NOT GOING LOW OR NOT GETTING THRU/<CRLF>
8482	043776	044504	020126	052523		
8483	044004	020102	020114	047516		
8484	044012	020124	047507	047111		
8485	044020	020107	047514	020127		
8486	044026	051117	047040	052117		
8487	044034	043440	052105	044524		
8488	044042	043516	052040	051110		
8489	044050	100125	000			
8490	044053	124	020117	040522		.ASCIZ /TO RACK E49 OR E49(D0) STUCK HIGH/
8491	044060	045503	042440	034464		

8492	044066	047440	020122	032105	
8493	044074	024071	030104	020051	
8494	044102	052123	041525	020113	
8495	044110	044510	044107	000	
8496	044115	121	047525	020124	EM102: .ASCIZ /QUOT OK REMAINDER BAD (ROM STATE FAILURE)/
8497	044122	045517	051040	046505	
8498	044130	044501	042116	051105	
8499	044136	041040	042101	024040	
8500	044144	047522	020115	052123	
8501	044152	052101	020105	040506	
8502	044160	046111	051125	024505	
8503	044166	000			
8504	044167	122	041501	020113	EM103: .ASCIZ /RACK E49(A1) STUCK LOW/
8505	044174	032105	024071	030501	
8506	044202	020051	052123	041525	
8507	044210	020113	047514	000127	
8508	044216	051107	045101	042040	EM104: .ASCIZ /GRAJ DIV SUB L NOT GOING HIGH OR RACK E49(D0) STUCK LOW/
8509	044224	053111	051440	041125	
8510	044232	046040	047040	052117	
8511	044240	043440	044517	043516	
8512	044246	044040	043511	020110	
8513	044254	051117	051040	041501	
8514	044262	020113	032105	024071	
8515	044270	030104	020051	052123	
8516	044276	041525	020113	047514	
8517	044304	000127			
8518	044306	052521	047524	042511	EM105: .ASCIZ /QUOTIENT & REMAINDER BAD/
8519	044314	052116	023040	051040	
8520	044322	046505	044501	042116	
8521	044330	051105	041040	042101	
8522	044336	000			
8523	044337	107	040522	020110	EM106: .ASCIZ /GRAH SR15 NOT GETTING TO RACK E64 OR E64(B0) STUCK HIGH/
8524	044344	051123	032461	047040	
8525	044352	052117	043440	052105	
8526	044360	044524	043516	052040	
8527	044366	020117	040522	045503	
8528	044374	042440	032066	047440	
8529	044402	020122	033105	024064	
8530	044410	030102	020051	052123	
8531	044416	041525	020113	044510	
8532	044424	044107	000		
8533	044427	111	041522	020110	EM110: .ASCIZ /IRCH N(1) NOT GETTING TO RACK E63 OR E63(D0) STUCK HIGH/
8534	044434	024116	024461	047040	
8535	044442	052117	043440	052105	
8536	044450	044524	043516	052040	
8537	044456	020117	040522	045503	
8538	044464	042440	031466	047440	
8539	044472	020122	033105	024063	
8540	044500	030104	020051	052123	
8541	044506	041525	020113	044510	
8542	044514	044107	000		
8543	044517	122	041501	020113	EM111: .ASCIZ /RACK E49(B1) STUCK HIGH/
8544	044524	032105	024071	030502	
8545	044532	020051	052123	041525	
8546	044540	020113	044510	044107	
8547	044546	000			

8548	044547	107	040522	020112	EM112: .ASCIZ /GRAJ DIV QUIT L NOT GOING HIGH OR NOT GETTING/<CRLF>
8549	044554	044504	020126	052521	
8550	044562	052111	046040	047040	
8551	044570	052117	043440	044517	
8552	044576	043516	044040	043511	
8553	044604	020110	051117	047040	
8554	044612	052117	043440	052105	
8555	044620	044524	043516	000200	
8556	044626	047524	051040	041501	.ASCIZ /TO RACK E63 OR E63(CO) STUCK LOW/
8557	044634	020113	033105	020063	
8558	044642	051117	042440	031466	
8559	044650	041450	024460	051440	
8560	044656	052524	045503	046040	
8561	044664	053517	000		
8562	044667	122	041501	020113	EM113: .ASCIZ /RACK E50(B0) STUCK LOW/
8563	044674	032505	024060	030102	
8564	044702	020051	052123	041525	
8565	044710	020113	047514	000127	
8566	044716	040522	045503	042440	EM114: .ASCIZ /RACK E64(B0) STUCK LOW/
8567	044724	032066	041050	024460	
8568	044732	051440	052524	045503	
8569	044740	046040	053517	000	
8570	044745	104	041526	031056	EM115: .ASCIZ /DVC.20,DVC.40,DVC.80,OR DVC.90 FAILED/
8571	044752	026060	053104	027103	
8572	044760	030064	042054	041526	
8573	044766	034056	026060	051117	
8574	044774	042040	041526	034456	
8575	045002	020060	040506	046111	
8576	045010	042105	000		
8577	045013	102	042101	041440	EM117: .ASCIZ /BAD CC'S IN DVC.90 OR RACK E63(D0) STUCK LOW/
8578	045020	023503	020123	047111	
8579	045026	042040	041526	034456	
8580	045034	020060	051117	051040	
8581	045042	041501	020113	033105	
8582	045050	024063	030104	020051	
8583	045056	052123	041525	020113	
8584	045064	047514	000127		
8585	045070	051107	045101	042040	EM120: .ASCIZ /GRAJ DIV QUIT DIDN'T GO LOW OR RACK E63(CO) STUCK HIGH/
8586	045076	053111	050440	044525	
8587	045104	020124	044504	047104	
8588	045112	052047	043440	020117	
8589	045120	047514	020127	051117	
8590	045126	051040	041501	020113	
8591	045134	033105	024063	030103	
8592	045142	020051	052123	041525	
8593	045150	020113	044510	044107	
8594	045156	000			
8595	045157	103	023503	020123	EM121: .ASCIZ /CC'S BAD DUE TO EITHER DIV.30 OR DVE.20/
8596	045164	040502	020104	052504	
8597	045172	020105	047524	042440	
8598	045200	052111	042510	020122	
8599	045206	044504	027126	030063	
8600	045214	047440	020122	053104	
8601	045222	027105	030062	000	
8602	045227	107	040522	020112	EM122: .ASCIZ /GRAJ E5 BAD(Z2(O)*LEFT SAVE(1))/
8603	045234	032505	041040	042101	

8604	045242	055050	024062	024460	
8605	045250	046052	043105	020124	
8606	045256	040523	042526	030450	
8607	045264	024451	000		
8608	045267	122	041501	020113	EM123: .ASCIZ /RACK E63(D0) STUCK LOW/
8609	045274	033105	024063	030104	
8610	045302	020051	052123	041525	
8611	045310	020113	047514	000127	
8612	045316	041503	051447	042040	EM124: .ASCIZ /CC'S DIDN'T LOAD PROPERLY/
8613	045324	042111	023516	020124	
8614	045332	047514	042101	050040	
8615	045340	047522	042520	046122	
8616	045346	000131			
8617	045350	051107	045101	042440	EM125: .ASCIZ /GRAJ E5(N(1)*SR15(1)) BAD OR ROM STATE BAD/
8618	045356	024065	024116	024461	
8619	045364	051452	030522	024065	
8620	045372	024461	020051	040502	
8621	045400	020104	051117	051040	
8622	045406	046517	051440	040524	
8623	045414	042524	041040	042101	
8624	045422	000			
8625	045423	121	047525	020124	EM127: .ASCIZ /QUOT BAD REMAINDER OK (ROM STATE FAILURE)/
8626	045430	040502	020104	042522	
8627	045436	040515	047111	042504	
8628	045444	020122	045517	024040	
8629	045452	047522	020115	052123	
8630	045460	052101	020105	040506	
8631	045466	046111	051125	024505	
8632	045474	000			
8633	045475	121	047525	044524	EM130: .ASCIZ /QUOTIENT BAD (ROM STATE FAILURE)/
8634	045502	047105	020124	040502	
8635	045510	020104	051050	046517	
8636	045516	051440	040524	042524	
8637	045524	043040	044501	052514	
8638	045532	042522	000051		
8639	045536	040502	020104	041503	EM134: .ASCIZ /BAD CC'S ON DIV OVERFLOW, STATE DVD.10/
8640	045544	051447	047440	020116	
8641	045552	044504	020126	053117	
8642	045560	051105	046106	053517	
8643	045566	020054	052123	052101	
8644	045574	020105	053104	027104	
8645	045602	030061	000		
8646	045605	122	020060	040502	EM135: .ASCIZ /RO BAD/
8647	045612	000104			
8648	045614	052123	052101	020105	EM136: .ASCIZ /STATE MTP.00 DIDN'T INC SP/
8649	045622	052115	027120	030060	
8650	045630	042040	042111	023516	
8651	045636	020124	047111	020103	
8652	045644	050123	000		
8653	045647	122	041501	020106	EM140: .ASCIZ 'RACF X/CLASS DOESN'T GO HIGH'
8654	045654	027530	046103	051501	
8655	045662	020123	047504	051505	
8656	045670	023516	020124	047507	
8657	045676	044040	043511	000110	
8658	045704	052115	027120	030061	EM141: .ASCIZ /MTP.10 FAILED TO RELOAD THE DR/
8659	045712	043040	044501	042514	

8660	045720	020104	047524	051040	
8661	045726	046105	040517	020104	
8662	045734	044124	020105	051104	
8663	045742	000			
8664	045743	123	020120	047514	EM142: .ASCIZ /SP LOADED INCORRECTLY/
8665	045750	042101	042105	044440	
8666	045756	041516	051117	042522	
8667	045764	052103	054514	000	
8668	045771	115	050124	030456	EM143: .ASCIZ /MTP.10 DIDN'T PUT PCB IN DR/
8669	045776	020060	044504	047104	
8670	046004	052047	050040	052125	
8671	046012	050040	041103	044440	
8672	046020	020116	051104	000	
8673	046025	122	041501	020106	EM144: .ASCIZ /RACF E20(4) STUCK HIGH/
8674	046032	031105	024060	024464	
8675	046040	051440	052524	045503	
8676	046046	044040	043511	000110	
8677	046054	043115	027120	030061	EM145: .ASCIZ /MFP.10 DIDN'T DECREMENT THE SP/
8678	046062	042040	042111	023516	
8679	046070	020124	042504	051103	
8680	046076	046505	047105	020124	
8681	046104	044124	020105	050123	
8682	046112	000			
8683	046113	122	020060	044504	EM146: .ASCIZ /RO DIDN'T GET PUT ON THE STACK/
8684	046120	047104	052047	043440	
8685	046126	052105	050040	052125	
8686	046134	047440	020116	044124	
8687	046142	020105	052123	041501	
8688	046150	000113			
8689	046152	040502	020104	041503	EM147: .ASCIZ /BAD CC'S, CC CONTROL ROM/
8690	046160	051447	020054	041503	
8691	046166	041440	047117	051124	
8692	046174	046117	051040	046517	
8693	046202	000			
8694	046203	115	050106	030056	EM151: .ASCIZ /MFP.00 BAD/
8695	046210	020060	040502	000104	
8696	046216	040502	020104	041503	EM152: .ASCIZ /BAD CC'S DUE TO MFP.00/
8697	046224	051447	042040	042525	
8698	046232	052040	020117	043115	
8699	046240	027120	030060	000	
8700	046245	124	050122	030056	EM155: .ASCIZ /TRP.01 FAILED TO LOAD BR/
8701	046252	020061	040506	046111	
8702	046260	042105	052040	020117	
8703	046266	047514	042101	041040	
8704	046274	000122			
8705	046276	051111	042103	044440	EM156: .ASCII /IRCD IOT DOESN'T GO LOW OR DAPE TV04 DOES/<CRLF>
8706	046304	052117	042040	042517	
8707	046312	047123	052047	043440	
8708	046320	020117	047514	020127	
8709	046326	051117	042040	050101	
8710	046334	020105	053124	032060	
8711	046342	042040	042517	100123	
8712	046350	047516	020124	047507	.ASCIZ /NOT GO HIGH OR DOESN'T GET TO THE ALU/
8713	046356	044040	043511	020110	
8714	046364	051117	042040	042517	
8715	046372	047123	052047	043440	

8716	046400	052105	052040	020117	
8717	046406	044124	020105	046101	
8718	046414	000125			
8719	046416	051124	027120	030460	EM157: .ASCIZ /TRP.01 FAILED TO LOAD DR/
8720	046424	043040	044501	042514	
8721	046432	020104	047524	046040	
8722	046440	040517	020104	051104	
8723	046446	000			
8724	046447	124	050122	030056	EM160: .ASCIZ /TRP.00 FAILED TO LOAD BR/
8725	046454	020060	040506	046111	
8726	046462	042105	052040	020117	
8727	046470	047514	042101	041040	
8728	046476	000122			
8729	046500	051111	042103	047440	EM161: .ASCII /IRCD OPCODE3 DOESN'T GO LOW OR DOESN'T/<CRLF>
8730	046506	041520	042117	931505	
8731	046514	042040	042517	047123	
8732	046522	052047	043440	020117	
8733	046530	047514	020127	051117	
8734	046536	042040	042517	047123	
8735	046544	052047	200		
8736	046547	107	052105	052040	.ASCIZ /GET THRU TO DAPE TV03/
8737	046554	051110	020125	047524	
8738	046562	042040	050101	020105	
8739	046570	053124	031460	000	
8740	046575	124	050122	030056	EM162: .ASCIZ /TRP.00 FAILED TO LOAD DR/
8741	046602	020060	040506	046111	
8742	046610	042105	052040	020117	
8743	046616	047514	042101	042040	
8744	046624	000122			
8745	046626	044502	020124	040506	EM163: .ASCIZ /BIT FAILED IN PIRQ REG/
8746	046634	046111	042105	044440	
8747	046642	020116	044520	050522	
8748	046650	051040	043505	000	
8749	046655	105	051122	051117	DH163: .ASCII /ERRORPC PIRQ TST NUM/<CRLF>
8750	046662	041520	020040	020040	
8751	046670	020040	050040	051111	
8752	046676	004521	020040	051524	
8753	046704	020124	052516	100115	
8754	046712	020011	054105	042520	.ASCIZ / EXPECT ACTUAL/
8755	046720	052103	020040	041501	
8756	046726	052524	046101	000	
8757		046734			.EVEN
8758	046734	001116	001162	001216	DT163: .WORD \$ERRPC,\$TMPO,\$EPIRQ,\$\$TSTNM,0
8759	046742	001210	000000		
8760	046746	042506	027124	030060	EM164: .ASCIZ /FET.00 HAD BAD BEN FIELD/
8761	046754	044040	042101	041040	
8762	046762	042101	041040	047105	
8763	046770	043040	042511	042114	
8764	046776	000			
8765	046777	124	041515	020102	EM165: .ASCIZ /TMCB E62(1) BAD OR TMCB HONOR PIR 1 NOT GOING LOW/
8766	047004	033105	024062	024461	
8767	047012	041040	042101	047440	
8768	047020	020122	046524	041103	
8769	047026	044040	047117	051117	
8770	047034	050040	051111	030440	
8771	047042	047040	052117	043440	

8772	047050	044517	043516	046040	
8773	047056	053517	000		
8774	047061	124	041515	020102	EM166: .ASCII /TMCB E51(9) OR E55(10,11) OR E62 BAD OR/<CRLF>
8775	047066	032505	024061	024471	
8776	047074	047440	020122	032505	
8777	047102	024065	030061	030454	
8778	047110	024461	047440	020122	
8779	047116	033105	020062	040502	
8780	047124	020104	051117	200	
8781	047131	124	041515	020101	.ASCIZ /TMCA INH BELOW BR6 STUCK LOW/
8782	047136	047111	020110	042502	
8783	047144	047514	020127	051102	
8784	047152	020066	052123	041525	
8785	047160	020113	047514	000127	
8786	047166	046524	040503	040440	EM167: .ASCIZ /TMCA ABOVE BR7 MIGHT BE STUCK LOW/
8787	047174	047502	042526	041040	
8788	047202	033522	046440	043511	
8789	047210	052110	041040	020105	
8790	047216	052123	041525	020113	
8791	047224	047514	000127		
8792	047230	046524	042503	041040	EM170: .ASCIZ /TMCE BRQ CLOCK MIGHT BE STUCK LOW/
8793	047236	050522	041440	047514	
8794	047244	045503	046440	043511	
8795	047252	052110	041040	020105	
8796	047260	052123	041525	020113	
8797	047266	047514	000127		
8798	047272	046524	041103	050040	EM171: .ASCII /TMCB PF(0)*(SF+TF) NOT GOING HIGH OR NOT/<CRLF>
8799	047300	024106	024460	024052	
8800	047306	043123	052053	024506	
8801	047314	047040	052117	043440	
8802	047322	044517	043516	044040	
8803	047330	043511	020110	051117	
8804	047336	047040	052117	200	
8805	047343	107	052105	044524	.ASCIZ /GETTING TO RACK E50 OR RACK E50(A1) BAD/
8806	047350	043516	052040	020117	
8807	047356	040522	045503	042440	
8808	047364	030065	047440	020122	
8809	047372	040522	045503	042440	
8810	047400	030065	040450	024461	
8811	047406	041040	042101	000	
8812	047413	124	041515	020102	EM172: .ASCII /TMCB PF(0)*(SF+-TF) NOT GOING LOW OR/<CRLF>
8813	047420	043120	030050	025051	
8814	047426	051450	025506	052055	
8815	047434	024506	047040	052117	
8816	047442	043440	044517	043516	
8817	047450	046040	053517	047440	
8818	047456	100122			
8819	047460	047516	020124	042507	.ASCII /NOT GETTING TO RACK E64 OR RACK E64(A1) BAD/<CRLF>
8820	047466	052124	047111	020107	
8821	047474	047524	051040	041501	
8822	047502	020113	033105	020064	
8823	047510	051117	051040	041501	
8824	047516	020113	033105	024064	
8825	047524	030501	020051	040502	
8826	047532	100104			
8827	047534	051117	052040	041515	.ASCII /OR TMCB PIRQ NOT GETTING TO DAPE OR DAPE TV05*07/<CRLF>

8828	047542	020102	044520	050522	
8829	047550	047040	052117	043440	
8830	047556	052105	044524	043516	
8831	047564	052040	020117	040504	
8832	047572	042520	047440	020122	
8833	047600	040504	042520	052040	
8834	047606	030126	025065	033460	
8835	047614	200			
8836	047615	116	052117	043440	.ASCIZ /NOT GOING HIGH OR NOT GETTING TO THE ALU/
8837	047622	044517	043516	044040	
8838	047630	043511	020110	051117	
8839	047636	047040	052117	043440	
8840	047644	052105	044524	043516	
8841	047652	052040	020117	044124	
8842	047660	020105	046101	000125	
8843	047666	046524	041103	042440	EM174: .ASCIZ /TMCB E62(2) BAD OR TMCB HONOR PIR 2 NOT GOING LOW/
8844	047674	031066	031050	020051	
8845	047702	040502	020104	051117	
8846	047710	052040	041515	020102	
8847	047716	047510	047516	020122	
8848	047724	044520	020122	020062	
8849	047732	047516	020124	047507	
8850	047740	047111	020107	047514	
8851	047746	000127			
8852	047750	046524	041103	042440	EM175: .ASCIZ /TMCB E63 BAD/
8853	047756	031466	041040	042101	
8854	047764	000			
8855	047765	114	053105	046105	EM176: .ASCIZ /LEVEL 2 INTERRUPT WHEN LEVEL 1 ON/
8856	047772	031040	044440	052116	
8857	050000	051105	052522	052120	
8858	050006	053440	042510	020116	
8859	050014	042514	042526	020114	
8860	050022	020061	047117	000	
8861	050027	124	041515	020102	EM177: .ASCIZ /TMCB E62(3) BAD OR TMCB HONOR PIR 3 NOT GOING LOW/
8862	050034	033105	024062	024463	
8863	050042	041040	042101	047440	
8864	050050	020122	046524	041103	
8865	050056	044040	047117	051117	
8866	050064	050040	051111	031440	
8867	050072	047040	052117	043440	
8868	050100	044517	043516	046040	
8869	050106	053517	000		
8870	050111	114	053105	046105	EM201: .ASCIZ /LEVEL 2 INTERRUPT WHEN CPU LEVEL 2 ON/
8871	050116	031040	044440	052116	
8872	050124	051105	052522	052120	
8873	050132	053440	042510	020116	
8874	050140	050103	020125	042514	
8875	050146	042526	020114	020062	
8876	050154	047117	000		
8877	050157	105	051122	051117	DH201: .ASCIZ /ERRORPC PIRQ TST NUM/
8878	050164	041520	020040	044520	
8879	050172	050522	020040	020040	
8880	050200	051524	020124	052516	
8881	050206	000115			
8882					
8883	050210	001116	001216	001210	DT201: .EVEN .WORD \$ERRPC,\$EPIRQ,\$\$TSTNM,0



8884	050216	000000				
8885	050220	046524	041103	042440	EM202:	.ASCIZ /TMCB E62(5) BAD OR TMCA HONOR PIR 4 NOT GOING LOW/
8886	050226	031066	032450	020051		
8887	050234	040502	020104	051117		
8888	050242	052040	041515	020101		
8889	050250	047510	047516	020122		
8890	050256	044520	020122	020064		
8891	050264	047516	020124	047507		
8892	050272	047111	020107	047514		
8893	050300	000127				
8894	050302	042514	042526	020114	EM204:	.ASCIZ /LEVEL 3 INTERRUPT WHEN CPU LEVEL 3 ON/
8895	050310	020063	047111	042524		
8896	050316	051122	050125	020124		
8897	050324	044127	047105	041440		
8898	050332	052520	046040	053105		
8899	050340	046105	031440	047440		
8900	050346	000116				
8901	050350	046524	041103	042440	EM205:	.ASCIZ /TMCB E62(11) BAD OR TMCA HONOR PIR 5 NOT GOING LOW/
8902	050356	031066	030450	024461		
8903	050364	041040	042101	047440		
8904	050372	020122	046524	040503		
8905	050400	044040	047117	051117		
8906	050406	050040	051111	032440		
8907	050414	047040	052117	043440		
8908	050422	044517	043516	046040		
8909	050430	053517	000			
8910	050433	114	053105	046105	EM207:	.ASCIZ /LEVEL 4 INTERRUPT WHEN CPU LEVEL 4 ON/
8911	050440	032040	044440	052116		
8912	050446	051105	052522	052120		
8913	050454	053440	042510	020116		
8914	050462	050103	020125	042514		
8915	050470	042526	020114	020064		
8916	050476	047117	000			
8917	050501	124	041515	020102	EM210:	.ASCIZ /TMCB E51(11) BAD OR TMCB E55(8-9) BAD/
8918	050506	032505	024061	030461		
8919	050514	020051	040502	020104		
8920	050522	051117	052040	041515		
8921	050530	020102	032505	024065		
8922	050536	026470	024471	041040		
8923	050544	042101	000			
8924	050547	124	041515	020102	EM211:	.ASCIZ /TMCB E70(1) BAD OR TMCA HONOR PIR6 NOT GOING LOW/
8925	050554	033505	024060	024461		
8926	050562	041040	042101	047440		
8927	050570	020122	046524	040503		
8928	050576	044040	047117	051117		
8929	050604	050040	051111	020066		
8930	050612	047516	020124	047507		
8931	050620	047111	020107	047514		
8932	050626	000127				
8933	050630	046524	041103	042440	EM212:	.ASCIZ /TMCB E63(12) BAD OR TMCB E61(1) BAD/
8934	050636	031466	030450	024462		
8935	050644	041040	042101	047440		
8936	050652	020122	046524	041103		
8937	050660	042440	030466	030450		
8938	050666	020051	040502	000104		
8939	050674	042514	042526	020114	EM213:	.ASCIZ /LEVEL 5 INTERRUPT WHEN CPU LEVEL 5 ON/

8940	050702	020065	047111	042524	
8941	050710	051122	050125	020124	
8942	050716	044127	047105	041440	
8943	050724	052520	046040	053105	
8944	050732	046105	032440	047440	
8945	050740	000116			
8946	050742	046524	041103	042440	EM214: .ASCIZ /TMCB E70(6) BAD OR TMCA HONOR PIR 7 NOT GOING LOW/
8947	050750	030067	033050	020051	
8948	050756	040502	020104	051117	
8949	050764	052040	041515	020101	
8950	050772	047510	047516	020122	
8951	051000	044520	020122	020067	
8952	051006	047516	020124	047507	
8953	051014	047111	020107	047514	
8954	051022	000127			
8955	051024	042514	042526	020114	EM216: .ASCIZ /LEVEL 6 INTERRUPT WHEN CPU LEVEL 6 ON/
8956	051032	020066	047111	042524	
8957	051040	051122	050125	020124	
8958	051046	044127	047105	041440	
8959	051054	052520	046040	053105	
8960	051062	046105	033040	047440	
8961	051070	000116			
8962	051072	044524	042515	052517	EM217: .ASCIZ /TIMEOUT ON DATI DIDN'T WORK/
8963	051100	020124	047117	042040	
8964	051106	052101	020111	044504	
8965	051114	047104	052047	053440	
8966	051122	051117	000113		
8967	051126	046524	041503	040440	EM220: .ASCII /TMCC AERF(1) L NOT GOING LOW/<CRLF>
8968	051134	051105	024106	024461	
8969	051142	046040	047040	052117	
8970	051150	043440	044517	043516	
8971	051156	046040	053517	200	
8972	051163	117	020122	046524	.ASCIZ /OR TMCB E53(11) BAD/
8973	051170	041103	042440	031465	
8974	051176	030450	024461	041040	
8975	051204	042101	000		
8976	051207	124	046511	047505	EM221: .ASCIZ /TIMEOUT ON DATO DIDN'T WORK/
8977	051214	052125	047440	020116	
8978	051222	040504	047524	042040	
8979	051230	042111	023516	020124	
8980	051236	047527	045522	000	
8981	051243	124	041515	020102	EM222: .ASCIZ /TMCB PS07(0) NOT GETTING TO TMCB E77 OR E77 BAD/
8982	051250	051520	033460	030050	
8983	051256	020051	047516	020124	
8984	051264	042507	052124	047111	
8985	051272	020107	047524	052040	
8986	051300	041515	020102	033505	
8987	051306	020067	051117	042440	
8988	051314	033467	041040	042101	
8989	051322	000			
8990	051323	102	032122	023040	EM223: .ASCIZ /BR4 & BR6 FAILED/
8991	051330	041040	033122	043040	
8992	051336	044501	042514	000104	
8993	051344	051102	020064	040506	EM224: .ASCII /BR4 FAILED. EITHER TMCB HONOR BR4 NOT GOING LOW OR/<CRLF>
8994	051352	046111	042105	020056	
8995	051360	044505	044124	051105	

8996	051366	052040	041515	020102	
8997	051374	047510	047516	020122	
8998	051402	051102	020064	047516	
8999	051410	020124	047507	047111	
9000	051416	020107	047514	020127	
9001	051424	051117	200		
9002	051427	124	041515	020102	.ASCIZ /TMCB E62(4) BAD OR INTERRUPT OR BG LOGIC ON UBC BAD/
9003	051434	033105	024062	024464	
9004	051442	041040	042101	047440	
9005	051450	020122	047111	042524	
9006	051456	051122	050125	020124	
9007	051464	051117	041040	020107	
9008	051472	047514	044507	020103	
9009	051500	047117	052440	041502	
9010	051506	041040	042101	000	
9011	051513	102	032122	043040	EM225: .ASCII /BR4 FAILED BR6 OK EITHER TMCB HONOR BR4 NOT GOING/<CRLF>
9012	051520	044501	042514	020104	
9013	051526	051102	020066	045517	
9014	051534	042440	052111	042510	
9015	051542	020122	046524	041103	
9016	051550	044040	047117	051117	
9017	051556	041040	032122	047040	
9018	051564	052117	043440	044517	
9019	051572	043516	200		
9020	051575	114	053517	047440	.ASCIZ /LOW OR TMCB E62(4) BAD/
9021	051602	020122	046524	041103	
9022	051610	042440	031066	032050	
9023	051616	020051	040502	000104	
9024	051624	046524	040503	044040	EM226: .ASCIZ /TMCA HONOR BR5 NOT GOING LOW OR TMCB E62(6) BAD/
9025	051632	047117	051117	041040	
9026	051640	032522	047040	052117	
9027	051646	043440	044517	043516	
9028	051654	046040	053517	047440	
9029	051662	020122	046524	041103	
9030	051670	042440	031066	033050	
9031	051676	020051	040502	000104	
9032	051704	046524	040503	044040	EM227: .ASCIZ /TMCA HONOR BR6 NOT GOING LOW OR TMCB E62(12) BAD/
9033	051712	047117	051117	041040	
9034	051720	033122	047040	052117	
9035	051726	043440	044517	043516	
9036	051734	046040	053517	047440	
9037	051742	020122	046524	041103	
9038	051750	042440	031066	030450	
9039	051756	024462	041040	042101	
9040	051764	000			
9041	051765	131	046105	055040	EM230: .ASCII /YEL ZONE FAILED EITHER TMCD SL YEL NOT GOING HIGH OR/<CRLF>
9042	051772	047117	020105	040506	
9043	052000	046111	042105	042440	
9044	052006	052111	042510	020122	
9045	052014	046524	042103	051440	
9046	052022	020114	042531	020114	
9047	052030	047516	020124	047507	
9048	052036	047111	020107	044510	
9049	052044	044107	047440	100122	
9050	052052	051117	052040	041515	.ASCII /OR TMCA HONOR SLY NOT GOING LOW OR TMCB E70(3) BAD/<CRLF>
9051	052060	020101	047510	047516	

9052	052066	020122	046123	020131	
9053	052074	047516	020124	047507	
9054	052102	047111	020107	047514	
9055	052110	020127	051117	052040	
9056	052116	041515	020102	033505	
9057	052124	024060	024463	041040	
9058	052132	042101	200		
9059	052135	117	020122	042502	.ASCII /OR BEN13 FAILED- EITHER TMCA HONOR SLY NOT GETTING/<CRLF>
9060	052142	030516	020063	040506	
9061	052150	046111	042105	020055	
9062	052156	044505	044124	051105	
9063	052164	052040	041515	020101	
9064	052172	047510	047516	020122	
9065	052200	046123	020131	047516	
9066	052206	020124	042507	052124	
9067	052214	047111	100107		
9068	052220	047524	052040	041515	.ASCIIZ /TO TMCB E53 OR E53(3) BAD/
9069	052226	020102	032505	020063	
9070	052234	051117	042440	031465	
9071	052242	031450	020051	040502	
9072	052250	000104			
9073	052252	046524	041503	042440	EM232: .ASCIIZ /TMCC E16(8) NOT GOING LOW OR TMCC E36 BAD/
9074	052260	033061	034050	020051	
9075	052266	047516	020124	047507	
9076	052274	047111	020107	047514	
9077	052302	020127	051117	052040	
9078	052310	041515	020103	031505	
9079	052316	020066	040502	000104	
9080	052324	042120	041522	051440	EM233: .ASCII /PDRC STACK LIMIT NOT GETTING TO TMCD AS A LOW OR/<CRLF>
9081	052332	040524	045503	046040	
9082	052340	046511	052111	047040	
9083	052346	052117	043440	052105	
9084	052354	044524	043516	052040	
9085	052362	020117	046524	042103	
9086	052370	040440	020123	020101	
9087	052376	047514	020127	051117	
9088	052404	200			
9089	052405	124	041515	020104	.ASCIIZ /TMCD E8 BAD/
9090	052412	034105	041040	042101	
9091	052420	000			
9092	052421	125	041502	020103	EM234: .ASCII /UBCC DATI NOT GETTING TO TMCC AS A LOW OR EITHER/<CRLF>
9093	052426	040504	044524	047040	
9094	052434	052117	043440	052105	
9095	052442	044524	043516	052040	
9096	052450	020117	046524	041503	
9097	052456	040440	020123	020101	
9098	052464	047514	020127	051117	
9099	052472	042440	052111	042510	
9100	052500	100122			
9101	052502	046524	041503	042440	.ASCIIZ /TMCC E30 OR E36 BAD/
9102	052510	030063	047440	020122	
9103	052516	031505	020066	040502	
9104	052524	000104			
9105	052526	046524	042103	051440	EM235: .ASCII /TMCD SL RED NOT GOING LOW OR TMCC ABORT/<CRLF>
9106	052534	020114	042522	020104	
9107	052542	047516	020124	047507	

9108	052550	047111	020107	047514	
9109	052556	020127	051117	052040	
9110	052564	041515	020103	041101	
9111	052572	051117	100124		
9112	052576	047516	020124	047507	.ASCII /NOT GOING LOW /
9113	052604	047111	020107	047514	
9114	052612	020127			
9115	052614	051117	052040	041515	.ASCIZ /OR TMCB E50(6) DIDN'T GO HIGH ON TMCC SERF(1)I/
9116	052622	020102	032505	024060	
9117	052630	024466	042040	042111	
9118	052636	023516	020124	047507	
9119	052644	044040	043511	020110	
9120	052652	047117	052040	041515	
9121	052660	020103	042523	043122	
9122	052666	030450	046051	000	
9123	052673	124	041515	020103	EM237: .ASCII /TMCC SERF(1) NOT GOING LOW OR/<CRLF>
9124	052700	042523	043122	030450	
9125	052706	020051	047516	020124	
9126	052714	047507	047111	020107	
9127	052722	047514	020127	051117	
9128	052730	200			
9129	052731	116	052117	043440	.ASCIZ /NOT GETTING TO TMCB E50(2&1)/
9130	052736	052105	044524	043516	
9131	052744	052040	020117	046524	
9132	052752	041103	042440	030065	
9133	052760	031050	030446	000051	
9134	052766	046524	041103	050040	EM240: .ASCII /TMCB PF(0)*(SF+-TF) NOT GOING HIGH OR/<CRLF>
9135	052774	024106	024460	024052	
9136	053002	043123	026453	043124	
9137	053010	020051	047516	020124	
9138	053016	047507	047111	020107	
9139	053024	044510	044107	047440	
9140	053032	100122			
9141	053034	047516	020124	042507	.ASCIZ /NOT GETTING TO RACK BRCAB04/
9142	053042	052124	047111	020107	
9143	053050	047524	051040	041501	
9144	053056	020113	051102	040503	
9145	053064	030102	000064		
9146	053070	041523	042503	051440	EM241: .ASCII /SCCE STACK OVERFLOW NOT GOING HIGH OR/<CRLF>
9147	053076	040524	045503	047440	
9148	053104	042526	043122	047514	
9149	053112	020127	047516	020124	
9150	053120	047507	047111	020107	
9151	053126	044510	044107	047440	
9152	053134	100122			
9153	053136	047516	020124	042507	.ASCIZ /NOT GETTING TO TMCD E31 OR E31 BAD/
9154	053144	052124	047111	020107	
9155	053152	047524	052040	041515	
9156	053160	020104	031505	020061	
9157	053166	051117	042440	030463	
9158	053174	041040	042101	000	
9159	053201	120	051104	020103	EM242: .ASCII /PDRC RED ZONE NOT GOING HIGH OR/<CRLF>
9160	053206	042522	020104	047532	
9161	053214	042516	047040	052117	
9162	053222	043440	044517	043516	
9163	053230	044040	043511	020110	

9164	053236	051117	200		
9165	053241	116	052117	043440	.ASCIZ /NOT GETTING TO TMCD E31 OR TMCD E31 BAD OR SL REG BIT 0 BAD/
9166	053246	052105	044524	043516	
9167	053254	052040	020117	046524	
9168	053262	042103	042440	030463	
9169	053270	047440	020122	046524	
9170	053276	042103	042440	030463	
9171	053304	041040	042101	047440	
9172	053312	020122	046123	051040	
9173	053320	043505	041040	052111	
9174	053326	030040	041040	042101	
9175	053334	000			
9176	053335	065	032062	030060	EM243: .ASCIZ /52400 PATTERN FAILED, 125000 PATTERN OK/
9177	053342	050040	052101	042524	
9178	053350	047122	043040	044501	
9179	053356	042514	026104	030440	
9180	053364	032462	030060	020060	
9181	053372	040520	052124	051105	
9182	053400	020116	045517	000	
9183	053405	105	051122	051117	DH243: .ASCII /ERRORPC SL REG TST NUM/<CRLF>
9184	053412	041520	020040	020040	
9185	053420	020040	046123	051040	
9186	053426	043505	020040	020040	
9187	053434	020040	051524	020124	
9188	053442	052516	100115		
9189	053446	020011	054105	042520	.ASCIZ / EXPECT ACTUAL/
9190	053454	052103	020040	041501	
9191	053462	052524	046101	000	
9192		053470			
9193	053470	001116	001162	001222	DT243: .EVEN \$ERRPC,\$TMPO,E2STKIM,\$\$TSTNM,0
9194	053476	001210	000000		
9195	053502	031061	030065	030060	EM244: .ASCIZ /125000 PATTERN FAILED 52400 PATTERN OK/
9196	053510	050040	052101	042524	
9197	053516	047122	043040	044501	
9198	053524	042514	020104	031065	
9199	053532	030064	020060	040520	
9200	053540	052124	051105	020116	
9201	053546	045517	000		
9202		053552			
9203	053552	001116	001162	001220	DT244: .EVEN \$ERRPC,\$TMPO,E1STKLM,\$\$TSTNM,0
9204	053560	001210	000000		
9205	053564	041523	042503	051440	EM245: .ASCII /SCCE SL ADDRESS NOT GETTING TO TMCD OR/<CRLF>
9206	053572	020114	042101	051104	
9207	053600	051505	020123	047516	
9208	053606	020124	042507	052124	
9209	053614	047111	020107	047524	
9210	053622	052040	041515	020104	
9211	053630	051117	200		
9212	053633	124	041515	020104	.ASCIZ /TMCD E28 OR E14 BAD/
9213	053640	031105	020070	051117	
9214	053646	042440	032061	041040	
9215	053654	042101	000		
9216	053657	124	041515	020104	EM246: .ASCII /TMCD LOW BYTE EN DOESN'T GO LOW OR/<CRLF>
9217	053664	047514	020127	054502	
9218	053672	042524	042440	020116	
9219	053700	047504	051505	023516	

9220	053706	020124	047507	046040	
9221	053714	053517	047440	100122	
9222	053722	047516	020124	042507	.ASCIZ /NOT GETTING THRU TO THE DMUX (PDRE) AS A LOW /
9223	053730	052124	047111	020107	
9224	053736	044124	052522	052040	
9225	053744	020117	044124	020105	
9226	053752	046504	054125	024040	
9227	053760	042120	042522	020051	
9228	053766	051501	040440	046040	
9229	053774	053517	000040		
9230	054000	046524	042103	042040	EM247: .ASCII /TMCD DMX S1 STUCK HIGH OR IT DOESN'T/<CRLF>
9231	054006	054115	051440	020061	
9232	054014	052123	041525	020113	
9233	054022	044510	044107	047440	
9234	054030	020122	052111	042040	
9235	054036	042517	047123	052047	
9236	054044	200			
9237	054045	107	052105	052040	.ASCIZ /GET THRU TO THE DMUX(PDRE) AS A LOW/
9238	054052	051110	020125	047524	
9239	054060	052040	042510	042040	
9240	054066	052515	024130	042120	
9241	054074	042522	020051	051501	
9242	054102	040440	046040	053517	
9243	054110	000			
9244	054111	102	052117	020110	EM250: .ASCIZ /BOTH PATTERNS FAILED/
9245	054116	040520	052124	051105	
9246	054124	051516	043040	044501	
9247	054132	042514	000104		
9248	054136	051105	047522	050122	DH250: .ASCII /ERRORPC SL REG SL REG TST NUM/<CRLF>
9249	054144	020103	020040	020040	
9250	054152	051440	020114	042522	
9251	054160	004507	020040	020040	
9252	054166	020040	046123	051040	
9253	054174	043505	020040	020040	
9254	054202	052040	052123	047040	
9255	054210	046525	200		
9256	054213	011	042440	050130	.ASCIZ / EXPECT ACTUAL EXPECT ACTUAL/
9257	054220	041505	020124	040440	
9258	054226	052103	040525	020114	
9259	054234	042440	050130	041505	
9260	054242	020124	040440	052103	
9261	054250	040525	000114		
9262					
9263	054254	001116	001162	001220	DT250: .EVEN \$ERRPC,\$TMP0,E1STKLM,\$TMP1,E2STKLM,\$\$TSTM,0
9264	054262	001164	001222	001210	
9265	054270	000000			
9266	054272	046524	042103	054440	EM251: .ASCII /TMCD YEL ZONE DIDN'T GO LOW OR IT DIDN'T GET THRU TO E31/<CRLF>
9267	054300	046105	055040	047117	
9268	054306	020105	044504	047104	
9269	054314	052047	043440	020117	
9270	054322	047514	020127	051117	
9271	054330	044440	020124	044504	
9272	054336	047104	052047	043440	
9273	054344	052105	052040	051110	
9274	054352	020125	047524	042440	
9275	054360	030463	200		

9276	054363	117	020122	046524		
9277	054370	042503	041440	041501		.ASCIZ /OR TMCE CACHE BEND DIDN'T GO HIGH ON SL RED/
9278	054376	042510	041040	047105		
9279	054404	020104	044504	047104		
9280	054412	052047	043440	020117		
9281	054420	044510	044107	047440		
9282	054426	020116	046123	051040		
9283	054434	042105	000			
9284	054437	124	041515	020104	EM252:	.ASCIZ /TMCD YEL ZONE DOESN'T GO LOW ON ADR 240/
9285	054444	042531	020114	047532		
9286	054452	042516	042040	042517		
9287	054460	047123	052047	043440		
9288	054466	020117	047514	020127		
9289	054474	047117	040440	051104		
9290	054502	031040	030064	000		
9291	054507	124	041515	020104	EM253:	.ASCIZ /TMCD YEL ZONE DOESN'T GO LOW ON ADR 140/
9292	054514	042531	020114	047532		
9293	054522	042516	042040	042517		
9294	054530	047123	052047	043440		
9295	054536	020117	047514	020127		
9296	054544	047117	040440	051104		
9297	054552	030440	030064	000		
9298	054557	124	041515	020104	EM254:	.ASCIZ /TMCD YEL ZONE DOESN'T GO HIGH OR DIDN'T GET THRU TO E31/
9299	054564	042531	020114	047532		
9300	054572	042516	042040	042517		
9301	054600	047123	052047	043440		
9302	054606	020117	044510	044107		
9303	054614	047440	020122	044504		
9304	054622	047104	052047	043440		
9305	054630	052105	052040	051110		
9306	054636	020125	047524	042440		
9307	054644	030463	000			
9308	054647	125	044516	052502	EM255:	.ASCIZ /UNIBUS TIMEOUT BIT IN CPU ERROR REG DIDN'T SET/
9309	054654	020123	044524	042515		
9310	054662	052517	020124	044502		
9311	054670	020124	047111	041440		
9312	054676	052520	042440	051122		
9313	054704	051117	051040	043505		
9314	054712	042040	042111	023516		
9315	054720	020124	042523	000124		
9316	054726	051105	047522	050122	DH255:	.ASCII /ERRORPC CPUERR REG TST NUM/<CRLF>
9317	054734	020103	020040	041440		
9318	054742	052520	051105	020122		
9319	054750	042522	020107	020040		
9320	054756	051524	020124	052516		
9321	054764	100115				
9322	054766	020011	054105	042520		.ASCIZ / EXPECT ACTUAL/
9323	054774	052103	020040	041501		
9324	055002	052524	046101	000		
9325	055007	103	052520	042440	EM256:	.ASCIZ /CPU ERRPROR REG DIDN'T CLEAR/
9326	055014	051122	051120	051117		
9327	055022	051040	043505	042040		
9328	055030	042111	023516	020124		
9329	055036	046103	040505	000122		
9330	055044	042531	020114	047532	EM257:	.ASCIZ /YEL ZONE BIT IN CPU ERROR REG DIDN'T SET/
9331	055052	042516	041040	052111		



9332 055060 044440 020116 050103  
9333 055066 020125 051105 047522  
9334 055074 020122 042522 020107  
9335 055102 044504 047104 052047  
9336 055110 051440 052105 000  
9337 055115 124 041515 020104  
9338 055122 032105 032050 020051  
9339 055130 047516 020124 047507  
9340 055136 047111 020107 047514  
9341 055144 020127 051117 042440  
9342 055152 020064 040502 000104  
9343 055160 042522 042101 055040  
9344 055166 047117 020105 044502  
9345 055174 020124 047111 041440  
9346 055202 052520 042440 051122  
9347 055210 051117 051040 043505  
9348 055216 042040 042111 023516  
9349 055224 020124 042523 000124  
9350 055232 046524 042103 042440  
9351 055240 032461 033050 020051  
9352 055246 051117 042440 034061  
9353 055254 030450 024464 047440  
9354 055262 020122 030505 020070  
9355 055270 040502 000104  
9356 055274 047506 046114 053517  
9357 055302 047111 020107 051511  
9358 055310 040440 046040 051511  
9359 055316 020124 043117 052040  
9360 055324 042510 051440 040524  
9361 055332 045503 046040 046511  
9362 055340 052111 051040 043505  
9363 055346 200  
9364 055347 046 051440 020120  
9365 055354 040526 052514 051505  
9366 055362 052040 040510 020124  
9367 055370 040503 051525 042105  
9368 055376 040440 020116 051105  
9369 055404 047522 027122 052040  
9370 055412 042510 020131 051101  
9371 055420 100105  
9372 055422 051107 052517 042520  
9373 055430 020104 041501 047503  
9374 055436 042122 047111 020107  
9375 055444 047524 042440 051122  
9376 055452 051117 052040 050131  
9377 055460 051505 000  
9378 055463 105 051122 051117  
9379 055470 041520 052040 051505  
9380 055476 020124 052516 041115  
9381 055504 051105 000200  
9382 055510 051411 040524 045503  
9383 055516 046040 046511 052111  
9384 055524 051040 043505 051511  
9385 055532 042524 004522 020011  
9386 055540 020040 020040 020040  
9387 055546 052123 041501 020113

EM260: .ASCIZ /TMCD E4(4) NOT GOING LOW OR E4 BAD/

EM261: .ASCIZ /READ ZONE BIT IN CPU ERROR REG DIDN'T SET/

EM262: .ASCIZ /TMCD E15(6) OR E18(14) OR E18 BAD/

EM263: .ASCII /FOLLOWING IS A LIST OF THE STACK LIMIT REG/<CRLF>

.ASCII /8 SP VALUES THAT CAUSED AN ERROR. THEY ARE/<CRLF>

.ASCIZ /GROUPED ACCORDING TO ERROR TYPES/

DH263: .ASCIZ /ERRORPC TEST NUMBER/<CRLF>

DH263A: .ASCII / STACK LIMIT REGISTER STACK POINTER/<CRLF>

9388	055554	047520	047111	042524	
9389	055562	100122			
9390	055564	041517	040524	020114	.ASCII /OCTAL 15 14 13 12 11 10 9 8 OCTAL/
9391	055572	020040	030440	020065	
9392	055600	032061	030440	020063	
9393	055606	031061	030440	020061	
9394	055614	030061	020040	020071	
9395	055622	034040	020040	020040	
9396	055630	020040	041517	040524	
9397	055636	114			
9398	055637	040	020040	030440	.ASCIZ / 15 14 13 12 11 10 9 8/<CRLF>
9399	055644	020065	032061	030440	
9400	055652	020063	031061	030440	
9401	055660	020061	030061	020040	
9402	055666	020071	034040	000200	
9403	055674	042522	020104	051124	DH263B: .ASCIZ /RED TRAP ON YEL ADR/
9404	055702	050101	047440	020116	
9405	055710	042531	020114	042101	
9406	055716	000122			
9407	055720	042522	020104	051124	DH263C: .ASCIZ /RED TRAP ON LEGAL ADR/
9408	055726	050101	047440	020116	
9409	055734	042514	040507	020114	
9410	055742	042101	000122		
9411	055746	042531	046114	053517	DH263D: .ASCIZ /YELLOW TRAP ON RED ADR/
9412	055754	052040	040522	020120	
9413	055762	047117	051040	042105	
9414	055770	040440	051104	000	
9415	055775	131	046105	052040	DH263E: .ASCIZ /YEL TRAP ON LEGAL ADR/
9416	056002	040522	020120	047117	
9417	056010	046040	043505	046101	
9418	056016	040440	051104	000	
9419	056023	116	020117	051124	DH263F: .ASCIZ /NO TRAP ON RED ADR/
9420	056030	050101	047440	020116	
9421	056036	042522	020104	042101	
9422	056044	000122			
9423	056046	047516	052040	040522	DH263G: .ASCIZ /NO TRAP ON YEL ADR/
9424	056054	020120	047117	054440	
9425	056062	046105	040440	051104	
9426	056070	000			
9427		056072			
9428	056072	055674			INDEX: .EVEN
9429	056074	055720			DH263B
9430	056076	055746			DH263C
9431	056100	055775			DH263D
9432	056102	056023			DH263E
9433	056104	056046			DH263F
9434	056106	047507	047111	020107	DH263G
9435	056114	047524	047040	054105	EM264: .ASCIZ /GOING TO NEXT TEST/
9436	056122	020124	042524	052123	
9437	056130	000			
9438					
9439	056131	116	052117	035105	EM265: .ASCII /NOTE: IF NONE OF THE ODD ADR ERRORS TRAP/<CRLF>
9440	056136	044440	020106	0475	
9441	056144	042516	047440	020106	
9442	056152	044124	020105	042117	
9443	056160	020104	042101	020122	

9444	056166	051105	047522	051522	
9445	056174	052040	040522	100120	
9446	056202	020040	020040	020040	.ASCII / THEN EITHER TMCC ODD ADRS ERR NOT GETTING TO/<CRLF>
9447	056210	044124	047105	042440	
9448	056216	052111	042510	020122	
9449	056224	046524	041503	047440	
9450	056232	042104	040440	051104	
9451	056240	020123	051105	020122	
9452	056246	047516	020124	042507	
9453	056254	052124	047111	020107	
9454	056262	047524	200		
9455	056265	040	020040	020040	.ASCII / TMCC BUS ERROR OR DAPB BAMX00 NOT/<CRLF>
9456	056272	052040	041515	020103	
9457	056300	052502	020123	051105	
9458	056306	047522	020122	051117	
9459	056314	042040	050101	020102	
9460	056322	040502	054115	030060	
9461	056330	047040	052117	200	
9462	056335	040	020040	020040	.ASCII / GETTING TO TMCC E7 AS A HIGH/<CRLF><CRLF>
9463	056342	043440	052105	044524	
9464	056350	043516	052040	020117	
9465	056356	046524	041503	042440	
9466	056364	020067	051501	040440	
9467	056372	044040	043511	100110	
9468	056400	200			
9469	056401	116	044505	044124	.ASCIIZ /NEITHER -BYIN NOR DATI CAUSED A TRAP/<CRLF>
9470	056406	051105	026440	054502	
9471	056414	047111	047040	051117	
9472	056422	042040	052101	020111	
9473	056430	040503	051525	042105	
9474	056436	040440	052040	040522	
9475	056444	100120	000		
9476	056447	111	041522	020104	EM266: .ASCII /IRCD BYIN DOESN'T GET TO TMCC E12 AS A HIGH/<CRLF>
9477	056454	054502	047111	042040	
9478	056462	042517	047123	052047	
9479	056470	043440	052105	052040	
9480	056476	020117	046524	041503	
9481	056504	042440	031061	040440	
9482	056512	020123	020101	044510	
9483	056520	044107	200		
9484	056523	117	020122	046524	.ASCIIZ /OR TMCC E12(9,8) BAD OR E7(2) BAD/
9485	056530	041503	042440	031061	
9486	056536	034450	034054	020051	
9487	056544	040502	020104	051117	
9488	056552	042440	024067	024462	
9489	056560	041040	042101	000	
9490	056565	104	050101	020102	EM267: .ASCIIZ /DAPB BAMX00 DOESN'T GET TO TMCC E7(4) OR E7 BAD/
9491	056572	040502	054115	030060	
9492	056600	042040	042517	047123	
9493	056606	052047	043440	052105	
9494	056614	052040	020117	046524	
9495	056622	041503	042440	024067	
9496	056630	024464	047440	020122	
9497	056636	033505	041040	042101	
9498	056644	000			
9499	056645	122	041501	020103	EM270: .ASCII /RACC UBSC00 DOESN'T GET TO TMCC E5(13) AS/<CRLF>

9500	056652	041125	041523	030060	
9501	056660	042040	042517	047123	
9502	056666	052047	043440	052105	
9503	056674	052040	020117	046524	
9504	056702	041503	042440	024065	
9505	056710	031461	020051	051501	
9506	056716	200			
9507	056717	101	046040	053517	.ASCIZ /A LOW OR E5(13) BAD/
9508	056724	047440	020122	032505	
9509	056732	030450	024463	041040	
9510	056740	042101	000		
9511	056743	122	041501	020103	EM271: .ASCII /RACC UBSC02 DOESN'T GET TO TMCC E12(5) AS/<CRLF>
9512	056750	041125	041523	031060	
9513	056756	042040	042517	047123	
9514	056764	052047	043440	052105	
9515	056772	052040	020117	046524	
9516	057000	041503	042440	031061	
9517	057006	032450	020051	051501	
9518	057014	200			
9519	057015	101	044040	043511	.ASCIZ /A HIGH OR E12(6) DOESN'T GO LOW OR E5(12) BAD/
9520	057022	020110	051117	042440	
9521	057030	031061	033050	020051	
9522	057036	047504	051505	023516	
9523	057044	020124	047507	046040	
9524	057052	053517	047440	020122	
9525	057060	032505	030450	024462	
9526	057066	041040	042101	000	
9527	057073	123	031515	033465	EM272: .ASCIZ /SM357*SRC1 DATI FAILED TO TRAP/
9528	057100	051452	041522	020061	
9529	057106	040504	044524	043040	
9530	057114	044501	042514	020104	
9531	057122	047524	052040	040522	
9532	057130	000120			
9533	057132	042117	020104	042101	EM273: .ASCIZ /ODD ADR BIT IN CPUERR REG DOESN'T SET/
9534	057140	020122	044502	020124	
9535	057146	047111	041440	052520	
9536	057154	051105	020122	042522	
9537	057162	020107	047504	051505	
9538	057170	023516	020124	042523	
9539	057176	000124			
9540	057200	047516	052040	040522	EM274: .ASCIZ /NO TRAP ON DATO/
9541	057206	020120	047117	042040	
9542	057214	052101	000117		
9543	057220	051520	032060	030450	EM275: .ASCII /PS04(1) DOESN'T GET TO TMCB E74(9) AS A HIGH/<CRLF>
9544	057226	020051	047504	051505	
9545	057234	023516	020124	042507	
9546	057242	020124	047524	052040	
9547	057250	041515	020102	033505	
9548	057256	024064	024471	040440	
9549	057264	020123	020101	044510	
9550	057272	044107	200		
9551	057275	117	020122	052111	.ASCII /OR IT DOESN'T GET TO E51(10) AS A LOW OR E51 BAD/<CRLF>
9552	057302	042040	042517	047123	
9553	057310	052047	043440	052105	
9554	057316	052040	020117	032505	
9555	057324	024061	030061	020051	

9556	057332	051501	040440	046040	
9557	057340	053517	047440	020122	
9558	057346	032505	020061	040502	
9559	057354	100104			
9560	057356	051117	044440	041522	.ASCII /OR IRCD RTT DOESN'T GET TO TMCB E74(11) AS A HIGH/<CRLF>
9561	057364	020104	052122	020124	
9562	057372	047504	051505	023516	
9563	057400	020124	042507	020124	
9564	057406	047524	052040	041515	
9565	057414	020102	033505	024064	
9566	057422	030461	020051	051501	
9567	057430	040440	044040	043511	
9568	057436	100110			
9569	057440	051117	052040	041515	.ASCII /OR TMCB HONOR T DIDN'T GO LOW OR TMCB TOK DID/<CRLF>
9570	057446	020102	047510	047516	
9571	057454	020122	020124	044504	
9572	057462	047104	052047	043440	
9573	057470	020117	047514	020127	
9574	057476	051117	052040	041515	
9575	057504	020102	047524	020113	
9576	057512	044504	100104		
9577	057516	047516	020124	047507	.ASCIIZ /NOT GO LOW OR IT DIDN'T GET THRU TMCB E53/
9578	057524	046040	053517	047440	
9579	057532	020122	052111	042040	
9580	057540	042111	023516	020124	
9581	057546	042507	020124	044124	
9582	057554	052522	052040	041515	
9583	057562	020102	032505	000063	
9584	057570	044502	020124	020064	EM276: .ASCIIZ /BIT 4 IN PSW DOESN'T SET/
9585	057576	047111	050040	053523	
9586	057604	042040	042517	047123	
9587	057612	052047	051440	052105	
9588	057620	000			
9589	057621	123	052105	044524	EM277: .ASCIIZ /SETTING PS<08> FAILED TO DISABLE T BIT TRAP/
9590	057626	043516	050040	036123	
9591	057634	034060	020076	040506	
9592	057642	046111	042105	052040	
9593	057650	020117	044504	040523	
9594	057656	046102	020105	020124	
9595	057664	044502	020124	051124	
9596	057672	050101	000		
9597	057675	124	041515	020102	EM300: .ASCIIZ /TMCB TOK DOESN'T GET TO DAPE E7(11) AS A LOW OR E7 BAD/
9598	057702	047524	020113	047504	
9599	057710	051505	023516	020124	
9600	057716	042507	020124	047524	
9601	057724	042040	050101	020105	
9602	057732	033505	030450	024461	
9603	057740	040440	020123	020101	
9604	057746	047514	020127	051117	
9605	057754	042440	020067	040502	
9606	057762	000104			
9607	057764	051111	042103	051040	EM301: .ASCII /IRCD RTT DOESN'T GET TO TMCB E74 AS A LOW/<CRLF>
9608	057772	052124	042040	042517	
9609	060000	047123	052047	043440	
9610	060006	052105	052040	020117	
9611	060014	046524	041103	042440	

9612	060022	032067	040440	020123	
9613	060030	020101	047514	100127	
9614	060036	051117	042440	032067	.ASCIZ /OR E74 BAD/
9615	060044	041040	042101	000	
9616	060051	124	041515	020102	EM302: .ASCIZ /TMCB E58(5) DIDN'T GO LOW OR E64 BAD/
9617	060056	032505	024070	024465	
9618	060064	042040	042111	023516	
9619	060072	020124	047507	046040	
9620	060100	053517	047440	020122	
9621	060106	033105	020064	040502	
9622	060114	000104			
9623	060116	046524	040503	040440	EM303: .ASCII /TMCA ABOVE BR7 DOESN'T GO LOW ON A YEL ZONE OR/<CRLF>
9624	060124	047502	042526	041040	
9625	060132	033522	042040	042517	
9626	060140	047123	052047	043440	
9627	060146	020117	047514	020127	
9628	060154	047117	040440	054440	
9629	060162	046105	055040	047117	
9630	060170	020105	051117	200	
9631	060175	111	020124	047504	.ASCIZ /IT DOESN'T GET TO TMCB E84(13) OR E84 BAD/
9632	060202	051505	023516	020124	
9633	060210	042507	020124	047524	
9634	060216	052040	041515	020102	
9635	060224	034105	024064	031461	
9636	060232	020051	051117	042440	
9637	060240	032070	041040	042101	
9638	060246	000			
9639	060247	124	041515	020101	EM304: .ASCIZ /TMCA ABOVE BR7 DOESN'T GET TO TMCB E84(1) OR E84 BAD/
9640	060254	041101	053117	020105	
9641	060262	051102	020067	047504	
9642	060270	051505	023516	020124	
9643	060276	042507	020124	047524	
9644	060304	052040	041515	020102	
9645	060312	034105	024064	024461	
9646	060320	047440	020122	034105	
9647	060326	020064	040502	000104	
9648	060334	046524	040503	040440	EM305: .ASCIZ /TMCA ABOVE BR7 DOESN'T GET TO TMCB E77(13) OR E77 BAD/
9649	060342	047502	042526	041040	
9650	060350	033522	042040	042517	
9651	060356	047123	052047	043440	
9652	060364	052105	052040	020117	
9653	060372	046524	041103	042440	
9654	060400	033467	030450	024463	
9655	060406	047440	020122	033505	
9656	060414	020067	040502	000104	
9657	060422	046524	040503	041040	EM306: .ASCII /TMCA BLOCK BR4 DOESN'T GO LOW ON PIR4 OR/<CRLF>
9658	060430	047514	045503	041040	
9659	060436	032122	042040	042517	
9660	060444	047123	052047	043440	
9661	060452	020117	047514	020127	
9662	060460	047117	050040	051111	
9663	060466	020064	051117	200	
9664	060473	111	020124	047504	.ASCIZ /IT DOESN'T GET TO TMCB E77(5) OR E77 BAD/
9665	060500	051505	023516	020124	
9666	060506	042507	020124	047524	
9667	060514	052040	041515	020102	

9668	060522	033505	024067	024465	
9669	060530	047440	020122	033505	
9670	060536	020067	040502	000104	
9671	060544	046524	040503	044440	EM307: .ASCIZ /TMCA INH BELOW PIR4 DOESN'T GO LOW ON PIR5/
9672	060552	044116	041040	046105	
9673	060560	053517	050040	051111	
9674	060566	020064	047504	051505	
9675	060574	023516	020124	047507	
9676	060602	046040	053517	047440	
9677	060610	020116	044520	032522	
9678	060616	000			
9679	060617	124	041515	020101	EM310: .ASCIZ /TMCA INH BELOW PIR4 DOESN'T GO LOW ON BR5/
9680	060624	047111	020110	042502	
9681	060632	047514	020127	044520	
9682	060640	032122	042040	042517	
9683	060646	047123	052047	043440	
9684	060654	020117	047514	020127	
9685	060662	047117	041040	032522	
9686	060670	000			
9687	060671	124	041515	020101	EM311: .ASCIZ /TMCA BLOCK LEVEL 4 DIDN'T GO LOW ON PIR6/
9688	060676	046102	041517	020113	
9689	060704	042514	042526	020114	
9690	060712	020064	044504	047104	
9691	060720	052047	043440	020117	
9692	060726	047514	020127	047117	
9693	060734	050040	051111	000066	
9694	060742	046524	040503	040440	EM312: .ASCII /TMCA ABOVE BR7 DOESN'T GO LOW(ON PIR 7) OR/<CRLF>
9695	060750	047502	042526	041040	
9696	060756	033522	042040	042517	
9697	060764	047123	052047	043440	
9698	060772	020117	047514	024127	
9699	061000	047117	050040	051111	
9700	061006	033440	020051	051117	
9701	061014	200			
9702	061015	111	020124	047504	.ASCIZ /IT DOESN'T GET TO TMCB E77(1) OR E77 BAD/
9703	061022	051505	023516	020124	
9704	061030	042507	020124	047524	
9705	061036	052040	041515	020102	
9706	061044	033505	024067	024461	
9707	061052	047440	020122	033505	
9708	061060	020067	040502	000104	
9709	061066	041125	042103	042440	EM313: .ASCIZ /UBCD EXT BRQ DIDN'T GO LOW ON HONOR BR5/
9710	061074	052130	041040	050522	
9711	061102	042040	042111	023516	
9712	061110	020124	047507	046040	
9713	061116	053517	047440	020116	
9714	061124	047510	047516	020122	
9715	061132	051102	000065		
9716	061136	047111	020110	042502	EM314: .ASCII /INH BELOW BR6 DOESN'T GO LOW(ON BR6) OR IT DOES/<CRLF>
9717	061144	047514	020127	051102	
9718	061152	020066	047504	051505	
9719	061160	023516	020124	047507	
9720	061166	046040	053517	047450	
9721	061174	020116	051102	024466	
9722	061202	047440	020122	052111	
9723	061210	042040	042517	100123	

9724	061216	047516	020124	042507		.ASCIZ /NOT GET THRU TMCA E82(6)/
9725	061224	020124	044124	052522		
9726	061232	052040	041515	020101		
9727	061240	034105	024062	024466		
9728	061246	000				
9729	061247	124	041515	020101	EM315:	.ASCII /TMCA E67(8) DOESN'T GO LOW (ON SL YEL) OR IT IS NOT/<CRLF>
9730	061254	033105	024067	024470		
9731	061262	042040	042517	047123		
9732	061270	052047	043440	020117		
9733	061276	047514	020127	047450		
9734	061304	020116	046123	054440		
9735	061312	046105	020051	051117		
9736	061320	044440	020124	051511		
9737	061326	047040	052117	200		
9738	061333	107	052105	044524		.ASCIZ /GETTING TO E60(12) OR E60 BAD/
9739	061340	043516	052040	020117		
9740	061346	033105	024060	031061		
9741	061354	020051	051117	042440		
9742	061362	030066	041040	042101		
9743	061370	000				
9744	061371	124	041515	020101	EM316:	.ASCIZ /TMCA E81(12) DOESN'T GO LOW(ON PIR5) OR IT DOES/<CRLF>
9745	061376	034105	024061	031061		
9746	061404	020051	047504	051505		
9747	061412	023516	020124	047507		
9748	061420	046040	053517	047450		
9749	061426	020116	044520	032522		
9750	061434	020051	051117	044440		
9751	061442	020124	047504	051505		
9752	061450	000200				
9753	061452	042507	020124	047524		.ASCIZ /GET TO E83(10) OR E83 BAD/
9754	061460	042440	031470	030450		
9755	061466	024460	047440	020122		
9756	061474	034105	020063	040502		
9757	061502	000104				
9758	061504	046524	040503	042440	EM317:	.ASCIZ /TMCA E81(2) BAD/
9759	061512	030470	031050	020051		
9760	061520	040502	000104			
9761	061524	046524	040503	042440	EM320:	.ASCII /TMCA E67(8) DOESN'T GO LOW(ON PIR 7) OR IT DOES/<CRLF>
9762	061532	033466	034050	020051		
9763	061540	047504	051505	023516		
9764	061546	020124	047507	046040		
9765	061554	053517	047450	020116		
9766	061562	044520	020122	024467		
9767	061570	047440	020122	052111		
9768	061576	042040	042517	100123		
9769	061604	047516	020124	042507		.ASCIZ /NOT GET TO E83(13) OR E83 BAD/
9770	061612	020124	047524	042440		
9771	061620	031470	030450	024463		
9772	061626	047440	020122	034105		
9773	061634	020063	040502	000104		
9774	061642	041125	042103	042440	EM321:	.ASCIZ /UBCD EXT BRQ DIDN'T GO LOW ON HONOR BR6/
9775	061650	052130	041040	050522		
9776	061656	042040	042111	023516		
9777	061664	020124	047507	046040		
9778	061672	053517	047440	020116		
9779	061700	047510	047516	020122		



9780	061706	051102	000066		
9781	061712	046524	040503	042440	EM322: .ASCIZ /TMCA E67(8) DOESN'T GET TO E83(1) OR E83 BAD/
9782	061720	033466	034050	020051	
9783	061726	047504	051505	023516	
9784	061734	020124	042507	020124	
9785	061742	047524	042440	031470	
9786	061750	030450	020051	051117	
9787	061756	042440	031470	041040	
9788	061764	042101	000		
9789	061767	124	041515	020101	EM323: .ASCII /TMCA E81(8) IS NOT GOING LOW OR IT IS NOT/<CRLF>
9790	061774	034105	024061	024470	
9791	062002	044440	020123	047516	
9792	062010	020124	047507	047111	
9793	062016	020107	047514	020127	
9794	062024	051117	044440	020124	
9795	062032	051511	047040	052117	
9796	062040	200			
9797	062041	107	052105	044524	.ASCIZ /GETTING TO E76(12) OR E76 BAD/
9798	062046	043516	052040	020117	
9799	062054	033505	024066	031061	
9800	062062	020051	051117	042440	
9801	062070	033067	041040	042101	
9802	062076	000			
9803	062077	124	041515	020101	EM324: .ASCIZ /TMCA E67(8) NOT GETTING TO E76(13) OR E76 IS BAD/
9804	062104	033105	024067	024470	
9805	062112	047040	052117	043440	
9806	062120	052105	044524	043516	
9807	062126	052040	020117	033505	
9808	062134	024066	031461	020051	
9809	062142	051117	042440	033067	
9810	062150	044440	020123	040502	
9811	062156	000104			
9812	062160	046524	040503	042440	EM325: .ASCIZ /TMCA E67(8) DOESN'T GET TO E76(1) OR E76 BAD/
9813	062166	033466	034050	020051	
9814	062174	047504	051505	023516	
9815	062202	020124	042507	020124	
9816	062210	047524	042440	033067	
9817	062216	030450	020051	051117	
9818	062224	042440	033067	041040	
9819	062232	042101	000		
9820	062235	124	041515	020101	EM326: .ASCIZ /TMCA E67(6) NOT GETTING TO E69(12) OR E69 IS BAD/
9821	06224	033105	024067	024466	
9822	06225	047040	052117	043440	
9823	062256	052105	044524	043516	
9824	062264	052040	020117	033105	
9825	062272	024071	031061	020051	
9826	062300	051117	042440	034466	
9827	062306	044440	020123	040502	
9828	062314	000104			
9829	062316	051520	020127	044502	EM327: .ASCII /PSW BIT 11 DOESN'T SET OR TMCF CLK HI PS DOES/<CRLF>
9830	062324	020124	030461	042040	
9831	062332	042517	047123	052047	
9832	062340	051440	052105	047440	
9833	062346	020122	046524	043103	
9834	062354	041440	045514	044040	
9835	062362	020111	051520	042040	

9836	062370	042517	100123		
9837	062374	047516	020124	047507	.ASCIZ /NOT GO LOW OR BIT 11 DOESN'T GET TO OR THRU THE DMUX/
9838	062402	046640	053517	047440	
9839	062410	020122	044502	020124	
9840	062416	030461	042040	042517	
9841	062424	047123	052047	043440	
9842	062432	052105	052040	020117	
9843	062440	051117	052040	051110	
9844	062446	020125	044124	020105	
9845	062454	046504	054125	000	
9846	062461	107	040522	020103	EM330: .ASCII /GRAC GDREG SET 1 DOESN'T GO HIGH (ON PAD 5) OR/<CRLF>
9847	062466	042107	042522	020107	
9848	062474	042523	020124	020061	
9849	062502	047504	051505	023516	
9850	062510	020124	047507	044040	
9851	062516	043511	020110	047450	
9852	062524	020116	040520	020104	
9853	062532	024465	047440	100122	
9854	062540	047504	051505	023516	.ASCIZ /DOESN'T GET TO THE GD REG/
9855	062546	020124	042507	020124	
9856	062554	047524	052040	042510	
9857	062562	043440	020104	042522	
9858	062570	000107			
9859	062572	051107	041501	043440	EM331: .ASCII /GRAC GSREG SET 1 DOESN'T GO HIGH(ON PAD 5) OR/<CRLF>
9860	062600	051123	043505	051440	
9861	062606	052105	030440	042040	
9862	062614	042517	047123	052047	
9863	062622	043440	020117	044510	
9864	062630	044107	047450	020116	
9865	062636	040520	020104	024465	
9866	062644	047440	100122		
9867	062650	052111	042040	042517	.ASCIZ /IT DOESN'T GET TO THE GS REG/
9868	062656	047123	052047	043440	
9869	062664	052105	052040	020117	
9870	062672	044124	020105	051507	
9871	062700	051040	043505	000	
9872	062705	107	040522	020102	EM332: .ASCIZ /GRAB SRC SET 1 DOESN'T GO LOW OR GRAC E23(6) BAD/
9873	062712	051123	020103	042523	
9874	062720	020124	020061	047504	
9875	062726	051505	023516	020124	
9876	062734	047507	046040	053517	
9877	062742	047440	020122	051107	
9878	062750	041501	042440	031462	
9879	062756	033050	020051	040502	
9880	062764	000104			
9881	062766	051107	041501	042440	EM333: .ASCIZ /GRAC E24(6) DOESN'T GO LOW/
9882	062774	032062	033050	020051	
9883	063002	047504	051505	023516	
9884	063010	020124	047507	046040	
9885	063016	053517	000		
9886	063021	107	040522	020103	EM334: .ASCIZ /GRAC E23(4) DOESN'T GO LOW/
9887	063026	031105	024063	024464	
9888	063034	042040	042517	047123	
9889	063042	052047	043440	020117	
9890	063050	047514	000127		
9891	063054	051107	041501	042440	EM335: .ASCIZ /GRAC E23(4) IS NOT GOING LOW/

9892	063062	031462	032050	020051	
9893	063070	051511	047040	052117	
9894	063076	043440	044517	043516	
9895	063104	046040	053517	000	
9896	063111	120	051104	020104	EM336: .ASCIZ /PDRD PS11 DOESN'T GET TO GRAC AS A LOW/
9897	063116	051520	030461	042040	
9898	063124	042517	047123	052047	
9899	063132	043440	052105	052040	
9900	063140	020117	051107	041501	
9901	063146	040440	020123	020101	
9902	063154	047514	000127		
9903	063160	042522	044507	052123	EM337: .ASCIZ /REGISTER SET 1 SOURCE HAS STUCK BITS/
9904	063166	051105	051440	052105	
9905	063174	030440	051440	052517	
9906	063202	041522	020105	040510	
9907	063210	020123	052123	041525	
9908	063216	020113	044502	051524	
9909	063224	000			
9910	063225	105	051122	051117	DH337: .ASCIZ /ERRORPC PATTERN TEST NUMBER/
9911	063232	041520	050040	052101	
9912	063240	042524	047122	052040	
9913	063246	051505	020124	052516	
9914	063254	041115	051105	000	
9915		063262			
9916	063262	001116	001164	001210	DT337: .EVEN .WORD \$ERRPC,\$TMP1,\$\$TSTNM,0
9917	063270	000000			
9918	063272	042522	044507	052123	EM340: .ASCIZ /REGISTER SET 1 DESTINATION HAS STUCK BITS/
9919	063300	051105	051440	052105	
9920	063306	030440	042040	051505	
9921	063314	044524	040516	044524	
9922	063322	047117	044040	051501	
9923	063330	051440	052524	045503	
9924	063336	041040	052111	000123	
9925	063344	051107	041101	042040	EM341: .ASCIZ /GRAB DST SET 1 DOESN'T GO LOW ON R14/
9926	063352	052123	051440	052105	
9927	063360	030440	042040	042517	
9928	063366	047123	052047	043440	
9929	063374	020117	047514	020127	
9930	063402	047117	051040	032061	
9931	063410	000			
9932	063411	107	040522	020102	EM342: .ASCIZ /GRAB SRC SET 1 DOESN'T GO LOW ON R14/
9933	063416	051123	020103	042523	
9934	063424	020124	020061	047504	
9935	063432	051505	023516	020124	
9936	063440	047507	046040	053517	
9937	063446	047440	020116	030522	
9938	063454	000064			
9939	063456	030065	030060	020060	EM343: .ASCIZ /50000 PATTERN FAILED IN PSW/
9940	063464	040520	052124	051105	
9941	063472	020116	040506	046111	
9942	063500	042105	044440	020116	
9943	063506	051520	000127		
9944	063512	033061	030064	030060	EM344: .ASCIZ /164000 PATTERN FAILED IN PSW/
9945	063520	050040	052101	042524	
9946	063526	047122	043040	044501	
9947	063534	042514	020104	047111	

9948	063542	050040	053523	000	
9949	063547	120	053523	044040	EM345: .ASCIZ /PSW HIGH BYTE DIDN'T CLEAR/
9950	063554	043511	020110	054502	
9951	063562	042524	042040	042111	
9952	063570	023516	020124	046103	
9953	063576	040505	000122		
9954	063602	051107	041101	042040	EM346: .ASCIZ /GRAB DST SET 1 DOESN'T GO LOW ON DF6*SUPER MODE/
9955	063610	052123	051440	052105	
9956	063616	030440	042040	042517	
9957	063624	047123	052047	043440	
9958	063632	020117	047514	020127	
9959	063640	047117	042040	033106	
9960	063646	051452	050125	051105	
9961	063654	046440	042117	000105	
9962	063662	051107	041101	051440	EM347: .ASCIZ /GRAB SRC SET 1 DOESN'T GO LOW ON SF6*SUPER MODE/
9963	063670	041522	051440	052105	
9964	063676	030440	042040	042517	
9965	063704	047123	052047	043440	
9966	063712	020117	047514	020127	
9967	063720	047117	051440	033106	
9968	063726	051452	050125	051105	
9969	063734	046440	042117	000105	
9970	063742	051107	041501	042440	EM350: .ASCIZ /GRAC E35(8) DOESN'T GO HIGH ON DF6*USER MODE/
9971	063750	032463	034050	020051	
9972	063756	047504	051505	023516	
9973	063764	020124	047507	044040	
9974	063772	043511	020110	047117	
9975	064000	042040	033106	052452	
9976	064006	042523	020122	047515	
9977	064014	042504	000		
9978	064017	107	040522	020103	EM351: .ASCIZ /GRAC E15(6) DOESN'T GO HIGH ON SF6*USER MODE/
9979	064024	030505	024065	024466	
9980	064032	042040	042517	047123	
9981	064040	052047	043440	020117	
9982	064046	044510	044107	047440	
9983	064054	020116	043123	025066	
9984	064062	051525	051105	046440	
9985	064070	042117	000105		
9986	064074	051525	051105	047440	EM352: .ASCIZ /USER OR SUPER SP DST FAILED BIT TEST/
9987	064102	020122	052523	042520	
9988	064110	020122	050123	042040	
9989	064116	052123	043040	044501	
9990	064124	042514	020104	044502	
9991	064132	020124	042524	052123	
9992	064140	000			
9993	064141	125	042523	020122	EM353: .ASCIZ /USER OR SUPER SP SRC FAILED BIT TEST/
9994	064146	051117	051440	050125	
9995	064154	051105	051440	020120	
9996	064162	051123	020103	040506	
9997	064170	046111	042105	041040	
9998	064176	052111	052040	051505	
9999	064204	000124			
10000	064206	051107	041501	042440	EM354: .ASCIZ /GRAC E20(4) NOT GOING LOW/
10001	064214	030062	032050	020051	
10002	064222	047516	020124	047507	
10003	064230	047111	020107	047514	

10004	064236	000127							
10005	064240	051107	041501	042440	EM355:	.ASCIZ	/GRAC E20(12) NOT GOING LOW ON MTP*DMO*DF6*PS12/		
10006	064246	030062	030450	024462					
10007	064254	047040	052117	043440					
10008	064262	044517	043516	046040					
10009	064270	053517	047440	020116					
10010	064276	052115	025120	046504					
10011	064304	025060	043104	025066					
10012	064312	051520	031061	000					
10013	064317	107	040522	020103	EM356:	.ASCIZ	/GRAC E20(11) NOT GOING LOW/		
10014	064324	031105	024060	030461					
10015	064332	020051	047516	020124					
10016	064340	047507	047111	020107					
10017	064346	047514	000127						
10018	064352	051523	020120	040502	EM357:	.ASCII	/SSP BAD ON MTP EITHER GRAC GRWE/<CRLF>		
10019	064360	020104	047117	046440					
10020	064366	050124	042440	052111					
10021	064374	042510	020122	051107					
10022	064402	041501	043440	053522					
10023	064410	100105							
10024	064412	044510	020102	051117		.ASCIZ	/HIB OR LOB NOT GOING LOW/		
10025	064420	046040	041117	047040					
10026	064426	052117	043440	044517					
10027	064434	043516	046040	053517					
10028	064442	000							
10029	064443	105	051122	051117	DH357:	.ASCII	/ERRORPC SSP TST NUM/<CRLF>		
10030	064450	041520	020040	020040					
10031	064456	020040	051523	004520					
10032	064464	051524	020124	052516					
10033	064472	100115							
10034	064474	020011	054105	042520		.ASCIZ	/ EXPECT ACTUAL/		
10035	064502	052103	020040	041501					
10036	064510	052524	046101	000					
10037	064515	107	040522	020103	EM360:	.ASCIZ	/GRAC E35(8) DIDN'T GO HIGH ON MTP*DMO*DF6*PS13/		
10038	064522	031505	024065	024470					
10039	064530	042040	042111	023516					
10040	064536	020124	047507	044040					
10041	064544	043511	020110	047117					
10042	064552	046440	050124	042052					
10043	064560	030115	042052	033106					
10044	064566	050052	030523	000063					
10045	064574	051111	041503	042040	EM361:	.ASCIZ	/IRCC DMO(MFP+MTP) NOT GOING HIGH ON MFP/		
10046	064602	030115	046450	050106					
10047	064610	046453	050124	020051					
10048	064616	047516	020124	047507					
10049	064624	047111	020107	044510					
10050	064632	044107	047440	020116					
10051	064640	043115	000120						
10052	064644	051107	041501	042440	EM362:	.ASCIZ	/GRAC E24(13) DOESN'T GO HIGH/		
10053	064652	032062	030450	024463					
10054	064660	042040	042517	047123					
10055	064666	052047	043440	020117					
10056	064674	044510	044107	000					
10057	064701	120	051104	020104	EM363:	.ASCIZ	/PDRD PS15 DOESN'T GET TO GRAC AS A HIGH/		
10058	064706	051520	032461	042040					
10059	064714	042517	047123	052047					

10060	064722	043440	052105	052040	
10061	064730	020117	051107	041501	
10062	064736	040440	020123	020101	
10063	064744	044510	044107	000	
10064	064751	123	050123	053440	EM364: .ASCIZ /SSP WAS READ BUT NEITHER USP NOR SSP WERE WRITTEN/
10065	064756	051501	051040	040505	
10066	064764	020104	052502	020124	
10067	064772	042516	052111	042510	
10068	065000	020122	051525	020120	
10069	065006	047516	020122	051523	
10070	065014	020120	042527	042522	
10071	065022	053440	044522	052124	
10072	065030	047105	000		
10073	065033	107	040522	020103	EM365: .ASCIZ /GRAC E23(13) DOESN'T GO HIGH/
10074	065040	031105	024063	031461	
10075	065046	020051	047504	051505	
10076	065054	023516	020124	047507	
10077	065062	044040	043511	000110	
10078	065070	051107	041501	042440	EM366: .ASCIZ /GRAC E23(3) DOESN'T GO HIGH/
10079	065076	031462	031450	020051	
10080	065104	047504	051505	023516	
10081	065112	020124	047507	044040	
10082	065120	043511	000110		
10083	065124	042120	042122	050040	EM367: .ASCII /PDRD PS14(0) DOESN'T GET TO PDRD E73(13)/<CRLF>
10084	065132	030523	024064	024460	
10085	065140	042040	042517	047123	
10086	065146	052047	043440	052105	
10087	065154	052040	020117	042120	
10088	065162	042122	042440	031467	
10089	065170	030450	024463	200	
10090	065175	101	020123	020101	.ASCIZ /AS A HIGH OR E73(12) NOT GOING LOW/
10091	065202	044510	044107	047440	
10092	065210	020122	033505	024063	
10093	065216	031061	020051	047516	
10094	065224	020124	047507	047111	
10095	065232	020107	047514	000127	
10096	065240	020101	051520	020127	EM370: .ASCII /A PSW HIGH BYTE BIT(S) DIDN'T SET ON LOAD PS 8/<CRLF>
10097	065246	044510	044107	041040	
10098	065254	052131	020105	044502	
10099	065262	024124	024523	042040	
10100	065270	042111	023516	020124	
10101	065276	042523	020124	047117	
10102	065304	046040	040517	020104	
10103	065312	051520	023040	200	
10104	065317	113	051105	042516	.ASCIZ /KERNEL MODE WITH A BR VALUE OF 174000/
10105	065324	020114	047515	042504	
10106	065332	053440	052111	020110	
10107	065340	020101	051102	053040	
10108	065346	046101	042525	047440	
10109	065354	020106	033461	030064	
10110	065362	030060	000		
10111	065365	101	041040	052111	EM371: .ASCII /A BIT(S) IN THE PSW CLEARED ON AN RTI IN USER MODE/<CRLF>
10112	065372	051450	020051	047111	
10113	065400	052040	042510	050040	
10114	065406	053523	041440	042514	
10115	065414	051101	042105	047440	

10116	065422	020116	047101	051040	
10117	065430	044524	044440	020116	
10118	065436	051525	051105	046440	
10119	065444	042117	100105		
10120	065450	020046	020101	051102	.ASCIZ /& A BR VALUE OF 0/
10121	065456	053040	046101	042525	
10122	065464	047440	020106	000060	
10123	065472	020101	044502	024124	EM372: .ASCII /A BIT(S) OF PSW <15,13:11> DIDN'T PRESET ON AN RTI/<CRLF>
10124	065500	024523	047440	020106	
10125	065506	051520	020127	030474	
10126	065514	026065	031461	030472	
10127	065522	037061	042040	042111	
10128	065530	023516	020124	051120	
10129	065536	051505	052105	047440	
10130	065544	020116	047101	051040	
10131	065552	044524	200		
10132	065555	111	020116	052523	.ASCIZ /IN SUPER MODE & A BR VALUE OF 134000/
10133	065562	042520	020122	047515	
10134	065570	042504	023040	040440	
10135	065576	041040	020122	040526	
10136	065604	052514	020105	043117	
10137	065612	030440	032063	030060	
10138	065620	000060			
10139	065622	020101	044502	024124	EM373: .ASCII /A BIT(S) IN PSW HIGH BYTE FAILED ON AN IOT IN USER/<CRLF>
10140	065630	024523	044440	020116	
10141	065636	051520	020127	044510	
10142	065644	044107	041040	052131	
10143	065652	020105	040506	046111	
10144	065660	042105	047440	020116	
10145	065666	047101	044440	052117	
10146	065674	044440	020116	051525	
10147	065702	051105	200		
10148	065705	115	042117	020105	.ASCIZ /MODE & A BR VALUE OF 0/
10149	065712	020046	020101	051102	
10150	065720	053040	046101	042525	
10151	065726	047440	020106	000060	
10152	065734	020101	051120	053105	EM374: .ASCIZ /A PREVIOUS MODE BIT(S) DIDN'T CLEAR ON AN IOT IN KERNEL/
10153	065742	047511	051525	046440	
10154	065750	042117	020105	044502	
10155	065756	024124	024523	042040	
10156	065764	042111	023516	020124	
10157	065772	046103	040505	020122	
10158	066000	047117	040440	020116	
10159	066006	047511	020124	047111	
10160	066014	045440	051105	042516	
10161	066022	000114			
10162	066024	051523	041522	045440	EM375: .ASCII /SSRC KT ABORT FLG DOESN'T GET TO TMCC E34(10) OR/<CRLF>
10163	066032	020124	041101	051117	
10164	066040	020124	046106	020107	
10165	066046	047504	051505	023516	
10166	066054	020124	042507	020124	
10167	066062	047524	052040	041515	
10168	066070	020103	031505	024064	
10169	066076	030061	020051	051117	
10170	066104	200			
10171	066105	105	032063	030450	.ASCIZ /E34(10) BAD OR TMCC AERF(1) DOESN'T GO HIGH ON A KT ABORT/

10172	066112	024460	041040	042101	
10173	066120	047440	020122	046524	
10174	066126	041503	040440	051105	
10175	066134	024106	024461	042040	
10176	066142	042517	047123	052047	
10177	066150	043440	020117	044510	
10178	066156	044107	047440	020116	
10179	066164	020101	052113	040440	
10180	066172	047502	052122	000	
10181	066177	120	053523	041040	EM376: .ASCIZ /PSW BIT 8 FAILED TO SET/
10182	066204	052111	034040	043040	
10183	066212	044501	042514	020104	
10184	066220	047524	051440	052105	
10185	066226	000			
10186	066227	124	041515	020102	EM400: .ASCII /TMCB SEGT DOESN'T GO LOW ON A KT ABORT OR/<CRLF>
10187	066234	042523	052107	042040	
10188	066242	042517	047123	052047	
10189	066250	043440	020117	047514	
10190	066256	020127	047117	040440	
10191	066264	045440	020124	041101	
10192	066272	051117	020124	051117	
10193	066300	200			
10194	066301	111	020124	047504	.ASCIZ /IT DOESN'T GET TO DAPE/
10195	066306	051505	023516	020124	
10196	066314	042507	020124	047524	
10197	066322	042040	050101	000105	
10198	066330	040504	042520	052040	EM401: .ASCIZ /DAPE TV05*07 DOESN'T GO HIGH ON SEGT/
10199	066336	030126	025065	033460	
10200	066344	042040	042517	047123	
10201	066352	052047	043440	020117	
10202	066360	044510	044107	047440	
10203	066366	020116	042523	052107	
10204	066374	000			
10205	066375	104	050101	020105	EM402: .ASCIZ /DAPE TV03 DOESN'T GO HIGH ON SEGT/
10206	066402	053124	031460	042040	
10207	066410	042517	047123	052047	
10208	066416	043440	020117	044510	
10209	066424	044107	047440	020116	
10210	066432	042523	052107	000	
10211	066437	124	041515	020101	EM403: .ASCII /TMCA SEG+CON+PAR DOESN'T GO LOW OR IT DOES/<CRLF>
10212	066444	042523	025507	047503	
10213	066452	025516	040520	020122	
10214	066460	047504	051505	023516	
10215	066466	020124	047507	046040	
10216	066474	053517	047440	020122	
10217	066502	052111	042040	042517	
10218	066510	100123			
10219	066512	047516	020124	042507	.ASCIZ /NOT GET THRU TMCB E70(2)/
10220	066520	020124	044124	052522	
10221	066526	052040	041515	020102	
10222	066534	033505	024060	024462	
10223	066542	000			
10224	066543	124	041515	020101	EM405: .ASCII /TMCA SEGTF DOESN'T GET TO E44 OR TMCE PAUSES H/<15><12>
10225	066550	042523	052107	020106	
10226	066556	047504	051505	023516	
10227	066564	020124	042507	020124	



10228	066572	047524	042440	032064	
10229	066600	047440	020122	046524	
10230	066606	042503	050040	052501	
10231	066614	042523	020123	006510	
10232	066622	012			
10233	066623	104	042517	047123	.ASCIZ /DOESN'T GET TO E44 OR E44 BAD/
10234	066630	052047	043440	052105	
10235	066636	052040	020117	032105	
10236	066644	020064	051117	042440	
10237	066652	032064	041040	042101	
10238	066660	000			
10239	066661	124	041515	020103	EM406: .ASCII /TMCC NEXM DOESN'T GO LOW OR IT DOESN'T GET THRU E34/<CRLF>
10240	066666	042516	046530	042040	
10241	066674	042517	047123	052047	
10242	066702	043440	020117	047514	
10243	066710	020127	051117	044440	
10244	066716	020124	047504	051505	
10245	066724	023516	020124	042507	
10246	066732	020124	044124	052522	
10247	066740	042440	032063	200	
10248	066745	117	020122	052111	.ASCIZ /OR IT DOESN'T GET TO E14 OR E40 BAD/
10249	066752	042040	042517	047123	
10250	066760	052047	043440	052105	
10251	066766	052040	020117	030505	
10252	066774	020064	051117	042440	
10253	067002	030064	041040	042101	
10254	067010	000			
10255	067011	116	054105	020115	EM410: .ASCII /NEXM BIT DIDN'T SET IN CPU ERROR REG/<CRLF>
10256	067016	044502	020124	044504	
10257	067024	047104	052047	051440	
10258	067032	052105	044440	020116	
10259	067040	050103	020125	051105	
10260	067046	047522	020122	042522	
10261	067054	100107			
10262	067056	051117	052040	041515	.ASCIZ /OR TMCC ABORT DOESN'T GO HIGH/
10263	067064	020103	041101	051117	
10264	067072	020124	047504	051505	
10265	067100	023516	020124	047507	
10266	067106	044040	043511	000110	
10267	067114	042516	046530	041040	EM411: .ASCIZ /NEXM BIT DIDN'T CLEAR IN CPU ERROR REG/
10268	067122	052111	042040	042111	
10269	067130	023516	020124	046103	
10270	067136	040505	020122	047111	
10271	067144	041440	052520	042440	
10272	067152	051122	051117	051040	
10273	067160	043505	000		
10274	067163	124	041515	020105	EM412: .ASCIZ /TMCE KT BEND DOESN'T GO LOW ON TMCC NEXM LOW/
10275	067170	052113	041040	047105	
10276	067176	020104	047504	051505	
10277	067204	023516	020124	047507	
10278	067212	046040	053517	047440	
10279	067220	020116	046524	041503	
10280	067226	047040	054105	020115	
10281	067234	047514	000127		
10282	067240	046524	042503	045440	EM413: .ASCIZ /TMCE KT BEND DOESN'T GO LOW ON TMCD SL RED/
10283	067246	020124	042502	042116	

10284	067254	042040	042517	047123	
10285	067262	052047	043440	020117	
10286	067270	047514	020127	047117	
10287	067276	052040	041515	020104	
10288	067304	046123	051040	042105	
10289	067312	000			
10290	067313	124	041515	020105	EM414: .ASCIZ /TMCE KT BEND DOESN'T GO LOW ON TMCC ODD ADRS ERR/
10291	067320	052113	041040	047105	
10292	067326	020104	047504	051505	
10293	067334	023516	020124	047507	
10294	067342	046040	053517	047440	
10295	067350	020116	046524	041503	
10296	067356	047440	042104	040440	
10297	067364	051104	020123	051105	
10298	067372	000122			
10299	067374	052113	040440	047502	EM415: .ASCIZ ?KT ABORT FAILED TO OVER-RIDE NEXM TRAP (KB11-E/EM)?
10300	067402	052122	043040	044501	
10301	067410	042514	020104	047524	
10302	067416	047440	042526	026522	
10303	067424	044522	042504	047040	
10304	067432	054105	020115	051124	
10305	067440	050101	024040	041113	
10306	067446	030461	042455	042457	
10307	067454	024515	000		
10308	067457	124	041515	020105	EM416: .ASCIZ /TMCE CACHE BEND DIDN'T GO HIGH ON KT ABORT/
10309	067464	040503	044103	020105	
10310	067472	042502	042116	042040	
10311	067500	042111	023516	020124	
10312	067506	047507	044040	043511	
10313	067514	020110	047117	045440	
10314	067522	020124	041101	051117	
10315	067530	000124			
10316	067532	040504	040520	041040	EM417: .ASCII /DAPA BR14 DOESN'T GET TO RACK E49 AS A LOW/<CRLF>
10317	067540	030522	020064	047504	
10318	067546	051505	023516	020124	
10319	067554	042507	020124	047524	
10320	067562	051040	041501	020113	
10321	067570	032105	020071	051501	
10322	067576	040440	046040	053517	
10323	067604	200			
10324	067605	117	020122	032105	.ASCIZ /OR E49(5) BAD/
10325	067612	024071	024465	041040	
10326	067620	042101	000		
10327	067623	111	046114	043505	EM420: .ASCIZ /ILLEGAL HALT BIT IN CPU ERROR REG DIDN'T SET/
10328	067630	046101	044040	046101	
10329	067636	020124	044502	020124	
10330	067644	047111	041440	052520	
10331	067652	042440	051122	051117	
10332	067660	051040	043505	042040	
10333	067666	042111	023516	020124	
10334	067674	042523	000124		
10335	067700	051523	040522	050040	EM421: .ASCII /SSRA PS RESTORE(1) DOESN'T GET TO RACK E63/<CRLF>
10336	067706	020123	042522	052123	
10337	067714	051117	024105	024461	
10338	067722	042040	042517	047123	
10339	067730	052047	043440	052105	

10340	067736	052040	020117	040522	
10341	067744	045503	042440	031466	
10342	067752	200			
10343	067753	117	020122	033105	.ASCIZ /OR E63(5) BAD/
10344	067760	024063	024465	041040	
10345	067766	042101	000		
10346	067771	125	041502	020102	EM422: .ASCII /UBCB PE ABORT DOESN'T GO LOW OR/<CRLF>
10347	067776	042520	040440	047502	
10348	070004	052122	042040	042517	
10349	070012	047123	052047	043440	
10350	070020	020117	047514	020127	
10351	070026	051117	200		
10352	070031	111	020124	047504	.ASCII /IT DOESN'T GET TO TMCC E33 OR E33 BAD/<CRLF>
10353	070036	051505	023516	020124	
10354	070044	042507	020124	047524	
10355	070052	052040	041515	020103	
10356	070060	031505	020063	051117	
10357	070066	042440	031463	041040	
10358	070074	042101	200		
10359	070077	117	020122	041125	.ASCII /OR UBCB PARITY ERR DOESN'T GET TO TMCB E53/<CRLF>
10360	070104	041103	050040	051101	
10361	070112	052111	020131	051105	
10362	070120	020122	047504	051505	
10363	070126	023516	020124	042507	
10364	070134	020124	047524	052040	
10365	070142	041515	020102	032505	
10366	070150	100063			
10367	070152	051501	040440	046040	.ASCIZ /AS A LOW OR E53(5) BAD/
10368	070160	053517	047440	020122	
10369	070166	032505	024063	024465	
10370	070174	041040	042101	000	
10371	070201	125	041502	020102	EM423: .ASCII /UBCB PARITY ERR DOESN'T GO LOW OR IT DOES/<CRLF>
10372	070206	040520	044522	054524	
10373	070214	042440	051122	042040	
10374	070222	042517	047123	052047	
10375	070230	043440	020117	047514	
10376	070236	020127	051117	044440	
10377	070244	020124	047504	051505	
10378	070252	200			
10379	070253	116	052117	043440	.ASCIZ /NOT GET TO DAPE/
10380	070260	052105	052040	020117	
10381	070266	040504	042520	000	
10382	070273	104	050101	020105	EM424: .ASCIZ /DAPE E11(4) BAD OR TV06 DOESN'T GET TO THE ALU/
10383	070300	030505	024061	024464	
10384	070306	041040	042101	047440	
10385	070314	020122	053124	033060	
10386	070322	042040	042517	047123	
10387	070330	052047	043440	052105	
10388	070336	052040	020117	044124	
10389	070344	020105	046101	000125	
10390	070352	040504	042520	042440	EM425: .ASCIZ /DAPE E7(1) BAD/
10391	070360	024067	024461	041040	
10392	070366	042101	000		
10393	070371	124	041515	020101	EM426: .ASCIZ /TMCA SEG+CON+PAR DOESN'T GO LOW ON CCBJ PARITY RAP/
10394	070376	042523	025507	047503	
10395	070404	025516	040520	020122	

10396	070412	047504	051505	023516	
10397	070420	020124	047507	046040	
10398	070426	053517	047440	020116	
10399	070434	041503	045102	050040	
10400	070442	051101	052111	020131	
10401	070450	051124	050101	000	
10402	070455	124	041515	020102	EM427: .ASCII /TMCB PART DOESN'T GO LOW OR DOES/<CRLF>
10403	070462	040520	052122	042040	
10404	070470	042117	047123	052047	
10405	070476	043440	020117	047514	
10406	070504	020127	051117	042040	
10407	070512	042517	100123		
10408	070516	047516	020124	042507	.ASCIZ /NOT GET TO UBCB OR UBCB E18(1) BAD/
10409	070524	020124	047524	052440	
10410	070532	041502	020102	051117	
10411	070540	052440	041502	020102	
10412	070546	030505	024070	024461	
10413	070554	041040	042101	000	
10414	070561	124	041515	020101	EM430: .ASCIZ /TMCA E67(8) DOESN'T GO LOW ON MGMT/
10415	070566	033105	024067	024470	
10416	070574	042040	042517	047123	
10417	070602	052047	043440	020117	
10418	070610	047514	020127	047117	
10419	070616	046440	046507	000124	
10420	070624	046524	040503	042440	EM431: .ASCIZ /TMCA E67(12) DOESN'T GO LOW ON MGMT/
10421	070632	033466	030450	024462	
10422	070640	042040	042517	047123	
10423	070646	052047	043440	020117	
10424	070654	047514	020127	047117	
10425	070662	046440	046507	000124	
10426	070670	046524	040503	042440	EM432: .ASCII /TMCA E68(6) DOESN'T GO LOW ON PAR TRP/<CRLF>
10427	070676	034066	033050	020051	
10428	070704	047504	051505	023516	
10429	070712	020124	047507	046040	
10430	070720	053517	047440	020116	
10431	070726	040520	020122	051124	
10432	070734	100120			
10433	070736	051117	042440	032464	.ASCIZ /OR E45(4) BAD/
10434	070744	032050	020051	040502	
10435	070752	000104			
10436	070754	046524	040503	042440	EM433: .ASCIZ /TMCA E68(8) DOESN'T GO LOW ON PAR TRP/
10437	070762	034066	034050	020051	
10438	070770	047504	051505	023516	
10439	070776	020124	047507	046040	
10440	071004	053517	047440	020116	
10441	071012	040520	020122	051124	
10442	071020	000120			
10443	071022	046524	041503	050040	EM435: .ASCII /TMCC PRIORITY CLEAR DIDN'T GO LOW OR DIDN'T/<CRLF>
10444	071030	044522	051117	052111	
10445	071036	020131	046103	040505	
10446	071044	020122	044504	047104	
10447	071052	052047	043440	020117	
10448	071060	047514	020127	051117	
10449	071066	042040	042111	023516	
10450	071074	100124			
10451	071076	042507	020124	044124	.ASCIZ /GET THRU TMCA E43(2) ON ABORT CLEAR/

10452	071104	052522	052040	041515	
10453	071112	020101	032105	024063	
10454	071120	024462	047440	020116	
10455	071126	041101	051117	020124	
10456	071134	046103	040505	000122	
10457	071142	052502	020123	041120	EM436: .ASCIZ /BUS PB DIDN'T GET TO UBCB PE ABORT/
10458	071150	042040	042111	023516	
10459	071156	020124	042507	020124	
10460	071164	047524	052440	041502	
10461	071172	020102	042520	040440	
10462	071200	047502	052122	000	
10463	071205	125	041502	020102	EM437: .ASCIZ /UBCB PARITY ERR DIDN'T GO LOW ON BUS PB/
10464	071212	040520	044522	054524	
10465	071220	042440	051122	042040	
10466	071226	042111	023516	020124	
10467	071234	047507	046040	053517	
10468	071242	047440	020116	052502	
10469	071250	020123	041120	000	
10470	071255	127	044501	020124	EM440: .ASCIZ /WAIT INSTRUCTION DIDN'T WAIT FOR INTERRUPT/
10471	071262	047111	052123	052522	
10472	071270	052103	047511	020116	
10473	071276	044504	047104	052047	
10474	071304	053440	044501	020124	
10475	071312	047506	020122	047111	
10476	071320	042524	051122	050125	
10477	071326	000124			
10478	071330	040527	052111	044440	EM441: .ASCIZ /WAIT INSTRUCTION FELL THRU ON T BIT TRAP/
10479	071336	051516	051124	041525	
10480	071344	044524	047117	043040	
10481	071352	046105	020114	044124	
10482	071360	052522	047440	020116	
10483	071366	020124	044502	020124	
10484	071374	051124	050101	000	
10485	071401	125	041502	020103	EM442: .ASCIZ /UBCC (PWRP+INTR) DOESN'T GO LOW ON TMCB PIRQ/
10486	071406	050050	051127	025506	
10487	071414	047111	051124	020051	
10488	071422	047504	051505	023516	
10489	071430	020124	047507	046040	
10490	071436	053517	047440	020116	
10491	071444	046524	041103	050040	
10492	071452	051111	000121		
10493	071456	030523	027063	030060	EM443: .ASCIZ /S13.00 FAILED/
10494	071464	043040	044501	042514	
10495	071472	000104			
10496	071474	032123	027065	030060	EM444: .ASCIZ /S45.00 FAILED/
10497	071502	043040	044501	042514	
10498	071510	000104			
10499	071512	052115	027120	030061	EM445: .ASCIZ /MTP.10 FAILED/
10500	071520	043040	044501	042514	
10501	071526	000104			
10502	071530	042516	027107	030071	EM446: .ASCIZ /NEG.90 FAILED/
10503	071536	043040	044501	042514	
10504	071544	000104			
10505	071546	032104	027065	030070	EM447: .ASCIZ /D45.80 FAILED/
10506	071554	043040	044501	042514	
10507	071562	000104			

10508	071564	032104	027065	030071	EM450: .ASCIZ /D45.90 FAILED/
10509	071572	043040	044501	042514	
10510	071600	000104			
10511	071602	032104	027065	030060	EM451: .ASCIZ /D45.00 FAILED/
10512	071610	043040	044501	042514	
10513	071616	000104			
10514	071620	032104	027065	030460	EM452: .ASCIZ /D45.01 FAILED/
10515	071626	043040	044501	042514	
10516	071634	000104			
10517	071636	047101	044440	046114	EM453: .ASCIZ /AN ILLEGAL INSTRUCTION FAILED TO TRAP/
10518	071644	043505	046101	044440	
10519	071652	051516	051124	041525	
10520	071660	044524	047117	043040	
10521	071666	044501	042514	020104	
10522	071674	047524	052040	040522	
10523	071702	000120			
10524	071704	046524	041103	042440	EM454: .ASCIZ /TMCB E53 DOESN'T GO HIGH OR TMCB PIRQ DOESN'T GO LOW/
10525	071712	031465	042040	042517	
10526	071720	047123	052047	043440	
10527	071726	020117	044510	044107	
10528	071734	047440	020122	046524	
10529	071742	041103	050040	051111	
10530	071750	020121	047504	051505	
10531	071756	023516	020124	047507	
10532	071764	046040	053517	000	
10533	071771	123	051123	020101	EM455: .ASCII /SSRA PS RESTORE IS STUCK HIGH OR IT IS NOT/<CRLF>
10534	071776	051520	051040	051505	
10535	072004	047524	042522	044440	
10536	072012	020123	052123	041525	
10537	072020	020113	044510	044107	
10538	072026	047440	020122	052111	
10539	072034	044440	020123	047516	
10540	072042	100124			
10541	072044	042507	052124	047111	.ASCIZ /GETTING THRU RACK E63(B0) AS A LOW/
10542	072052	020107	044124	052522	
10543	072060	051040	041501	020113	
10544	072066	033105	024063	030102	
10545	072074	020051	051501	040440	
10546	072102	046040	053517	000	
10547	072107	124	041515	020103	EM456: .ASCIZ /TMCC E5(11) DOESN'T GO HIGH ON DATI OR DATO/
10548	072114	032505	030450	024461	
10549	072122	042040	042517	047123	
10550	072130	052047	043440	020117	
10551	072136	044510	044107	047440	
10552	072144	020116	040504	044524	
10553	072152	047440	020122	040504	
10554	072160	047524	000		
10555	072163	120	053523	041040	EM457: .ASCIZ /PSW BIT 11 DOESN'T CLEAR/
10556	072170	052111	030440	020061	
10557	072176	047504	051505	023516	
10558	072204	020124	046103	040505	
10559	072212	000122			
10560	072214	051520	020127	044103	EM460: .ASCIZ /PSW CHANGED ON RESET IN KERNEL MODE/
10561	072222	047101	042507	020104	
10562	072230	047117	051040	051505	
10563	072236	052105	044440	020116	

10564	072244	042513	047122	046105	
10565	072252	046440	042117	000105	
10566	072260	051520	020127	044103	EM461: .ASCIZ /PSW CHANGES ON RESET IN SUPER MODE/
10567	072266	047101	042507	020123	
10568	072274	047117	051040	051505	
10569	072302	052105	044440	020116	
10570	072310	052523	042520	020122	
10571	072316	047515	042504	000	
10572	072323	115	046505	046440	EM703: .ASCIZ /MEM MGT TESTS DISABLED/<CRLF>
10573	072330	052107	052040	051505	
10574	072336	051524	042040	051511	
10575	072344	041101	042514	100104	
10576	072352	000			
10577	072353	103	041501	042510	EM704: .ASCIZ /CACHE TESTS DISABLED/<CRLF>
10578	072360	052040	051505	051524	
10579	072366	042040	051511	041101	
10580	072374	042514	100104	000	
10581	072401	125	044516	052502	EM706: .ASCIZ /UNIBUS PE TEST DISABLED/<CRLF>
10582	072406	020123	042520	052040	
10583	072414	051505	020124	044504	
10584	072422	040523	046102	042105	
10585	072430	000200			
10586	072432	047516	041040	020122	EM710: .ASCIZ /NO BR 5 DEVICE/<CRLF>
10587	072440	020065	042504	044526	
10588	072446	042503	000200		
10589	072452	047516	041040	020122	EM711: .ASCIZ /NO BR 6 DEVICE/<CRLF>
10590	072460	020066	042504	044526	
10591	072466	042503	000200		
10592	072472	051102	032040	052040	EM712: .ASCIZ /BR 4 TESTS DISABLED/<CRLF>
10593	072500	051505	051524	042040	
10594	072506	051511	041101	042514	
10595	072514	100104	000		
10596	072517	102	020122	020065	EM713: .ASCIZ /BR 5 TESTS DISABLED/<CRLF>
10597	072524	042524	052123	020123	
10598	072532	044504	040523	046102	
10599	072540	042105	000200		
10600	072544	051102	033040	052040	EM715: .ASCIZ /BR 6 TESTS DISABLED/<CRLF>
10601	072552	051505	051524	042040	
10602	072560	051511	041101	042514	
10603	072566	100104	000		
10604	072571	117	051120	052040	EM717: .ASCIZ /OPR TEST DISABLED/<CRLF>
10605	072576	051505	020124	044504	
10606	072604	040523	046102	042105	
10607	072612	000200			
10608	072614	047514	045517	040440	EM720: .ASCII /LOOK AT THE CONSOLE LIGHTS/<CRLF>
10609	072622	020124	044124	020105	
10610	072630	047503	051516	046117	
10611	072636	020105	044514	044107	
10612	072644	051524	200		
10613	072647	124	042510	042040	.ASCII /THE DATA LIGHTS SHOULD READ 166667/<CRLF>
10614	072654	052101	020101	044514	
10615	072662	044107	051524	051440	
10616	072670	047510	046125	020104	
10617	072676	042522	042101	030440	
10618	072704	033066	033066	100067	
10619	072712	044124	020105	042101	.ASCIZ /THE ADDRESS LIGHTS SHOULD READ /

10620	072720	051104	051505	020123	
10621	072726	044514	044107	051524	
10622	072734	051440	047510	046125	
10623	072742	020104	042522	042101	
10624	072750	020040	000		
10625	072753	200	044103	047101	EM721: .ASCIZ <CRLF>/CHANGE SWITCH 7 TO CONTINUE/<CRLF>
10626	072760	042507	051440	044527	
10627	072766	041524	020110	020067	
10628	072774	047524	041440	047117	
10629	073002	044524	052516	100105	
10630	073010	000			
10631	073011	200	054524	042520	EM722: .ASCIZ <CRLF>/TYPE A CHARACTER TO CONTINUE/<CRLF>
10632	073016	040440	041440	040510	
10633	073024	040522	052103	051105	
10634	073032	052040	020117	047503	
10635	073040	052116	047111	042525	
10636	073046	000200			
10637	073050	047200	020117	040515	EM724: .ASCIZ <CRLF>/NO MAP REGISTERS AVAILABLE FOR TEST 75/<CRLF>
10638	073056	020120	042522	044507	
10639	073064	052123	051105	020123	
10640	073072	053101	044501	040514	
10641	073100	046102	020105	047506	
10642	073106	020122	042524	052123	
10643	073114	033440	100065	000	
10644	073121	116	020117	020114	EM725: .ASCIZ /NO L CLOCK/<CRLF>
10645	073126	046103	041517	100113	
10646	073134	000			
10647		073136			
10648	073136	073136			ADRTAB: .EVEN
10649	073140	005476			.WORD
10650	073142	005660			TST1
10651	073144	006154			TST2
10652	073146	006436			TST3
10653	073150	007200			TST4
10654	073152	007314			TST5
10655	073154	007374			TST6
10656	073156	007460			TST7
10657	073160	010302			TST10
10658	073162	010442			TST11
10659	073164	011004			TST12
10660	073166	011120			TST13
10661	073170	012600			TST14
10662	073172	012746			TST15
10663	073174	013120			TST16
10664	073176	013306			TST17
10665	073200	013446			TST20
10666	073202	013562			TST21
10667	073204	013766			TST22
10668	073206	014336			TST23
10669	073210	014474			TST24
10670	073212	014632			TST25
10671	073214	015002			TST26
10672	073216	015156			TST27
10673	073220	015356			TST30
10674	073222	015530			TST31
10675	073224	016650			TST32
					TST33



10676	073226	017144	TST34
10677	073230	017274	TST35
10678	073232	017424	TST36
10679	073234	020016	TST37
10680	073236	020432	TST40
10681	073240	021166	TST41
10682	073242	021432	TST42
10683	073244	022042	TST43
10684	073246	022476	TST44
10685	073250	023234	TST45
10686	073252	023504	TST46
10687	073254	023556	TST47
10688	073256	026400	TST50
10689	073260	027030	TST51
10690	073262	027244	TST52
10691	073264	027414	TST53
10692	073266	027664	TST54
10693	073270	030030	TST55
10694	073272	030264	TST56
10695	073274	030412	TST57
10696	073276	030622	TST60
10697	073300	030700	TST61
10698	073302	031444	TST62
10699	073304	031560	TST63
10700	073306	032054	TST64
10701	073310	032366	
10702		000001	

SUBTAB: \$EOP  
.END

ADRTAB 073136	7620	7630	10648#											
BINARY 034606	7550*	7553*	7555	7561#										
BIT0 = 000001	1842#	3669	3699	4101	7640									
BIT00 = 000001	1832#	1842												
BIT01 = 000002	1831#	1841												
BIT02 = 000004	1830#	1840												
BIT03 = 000010	1829#	1839												
BIT04 = 000020	1828#	1838												
BIT05 = 000040	1827#	1837												
BIT06 = 000100	1826#	1836												
BIT07 = 000200	1825#	1835												
BIT08 = 000400	1824#	1834	7327											
BIT09 = 001000	1823#	1833	7335	7401										
BIT1 = 000002	1841#	5287	5298	5300	5564									
BIT10 = 002000	1822#	4621	4665	6043	7384									
BIT11 = 004000	1821#	4654	4699	6059	6429	6430	6432	6437	6446	6448	6460	6462	6474	
	6485	6491	6495	6501	6503	6505	6512	6515	6529	6566	6568	7342		
BIT12 = 010000	1820#	4569	4687	4734	6077	6174	6205	6222	6718	6789	7220			
BIT13 = 020000	1819#	4721	4776	6092	6243	6292	6309	7247	7275	7391				
BIT14 = 040000	1818#	4577	4758	4811	6134	6259	6327	6357	6620	6622	6631	6634	6642	
	6654	6677	6737	6742	6748	6784	6807	6853	6855	6993				
BIT15 = 100000	1817#	3405	3406	3465	3473	3516	3531	3541	3561	3564	3590	3697	3726	
	3740	3804	3875	3922	4183	4190	4311	4319	4321	4349	4397	4434	4586	
	4761	4798	4875	6149	6274	6342	6372	6644	6646	6656	6659	6692	6699	
	6811	6822	6834	7057										
BIT2 = 000004	1840#	5422	5425	5558	5577									
BIT3 = 000010	1839#	5223	5226	5528	5549	5567								
BIT4 = 000020	1838#	3361	3396	3455	4503	4551	4833	4844	4869	4885	4888	5034	5063	
	5099	5132	5166	5245	5309	5346	5719	5887	5975	6112	6168	6199	6779	
	6972													
BIT5 = 000040	1837#													
BIT6 = 000100	1836#	3254	3347	3353	3354	3372	3374	3392	4988	5316	5319	5322	5652	
	5666	5705	6386	6397	6403	7027	7041	7154	7159	7605				
BIT7 = 000200	1835#	6999	7582											
BIT8 = 000400	1834#	5910												
BIT9 = 001000	1833#	4558	4562	4633	5289	5581	6011	6027						
BPTVEC = 000014	1849#	4467*												
BUFFDP 034212	7485#	7491												
CACHSP 033072	3257	7269#												
CACHVE = 000114	1856#	3257*	3258*											
CHAINQ 034714	7209	7590#												
CHGE1 023566	5969#													
CHGE2 023574	5970#													
CONTRL = 177746	1866#	3190*	3263*											
CPUERR = 177766	1879#	3259*	4601*	4844	4885	4887	4890*	4891	4950*	4960*	4964*	5223	5225	
	5230*	5231	5422	5424	5428*	5429	5558	5567	5593*	5645*	5652	5655*	5666	
	5705	5708*	6384*	6999	7001	7005*	7006	7253	7266*					
CPUSPU 032666	3255	3359	4480	4593	4599	4626	4659	4688	4692	4696	4723	4727	4730	
	4759	4768	4773	4800	4804	4807	4883	4949	4961	4963	5191	5200	5212	
	5359	5363	5369	5383	5394	5407	5417	5421	5569	5607	5636	5640	5661	
	5665	5681	5698	5704	6032	6048	6064	6227	6234	6314	6383	7009	7241#	
CR = 000015	1754#	7734	7744											
CRLF = 000200	1755#	3236	7707	7744	7960	7966	7974	7985	8064	8080	8115	8146	8194	
	8237	8253	8269	8301	8328	8387	8481	8548	8705	8729	8749	8774	8798	
	8812	8819	8827	8967	8993	9011	9041	9050	9059	9080	9092	9105	9123	
	9134	9146	9159	9183	9205	9216	9230	9248	9266	9316	9356	9364	9378	

		9382	9398	9439	9446	9455	9462	9469	9476	9499	9511	9543	9551	9560
		9569	9607	9623	9657	9694	9716	9729	9744	9761	9789	9829	9846	9859
		10018	10029	10083	10096	10111	10123	10139	10162	10186	10211	10239	10255	10316
		10335	10346	10352	10359	10371	10402	10426	10443	10533	10572	10577	10581	10586
		10589	10592	10596	10600	10604	10608	10613	10625	10631	10637	10644		
DH1	036626	2263	2267	2270	7985#									
DH14	037457	2297	2562	8064#										
DH15	037602	2301	8080#											
DH163	046655	2613	8749#											
DH201	050157	2647	2656	2665	2674	2686	2695	8877#						
DH243	053405	2758	2762	9183#										
DH25	040357	2325	2445	8146#										
DH250	054136	2774	9248#											
DH255	054726	2789	2795	2801	3063	3087	9316#							
DH263	055463	2807	7461	9378#										
DH263A	055510	7478	9382#											
DH263B	055674	9403#	9428											
DH263C	055720	9407#	9429											
DH263D	055746	9411#	9430											
DH263E	055775	9415#	9431											
DH263F	056023	9419#	9432											
DH263G	056046	9423#	9433											
DH337	063225	2939	2942	2972	2975	9910#								
DH357	064443	2987	10029#											
DH4	037075	2273	2276	2279	2282	2285	2288	2291	2294	2304	2307	2310	2313	2316
		2319	2322	2329	2336	2342	2345	2348	2351	2354	2357	2360	2403	2430
		2433	2439	2451	2460	2463	2469	2472	2478	2484	2487	2490	2493	2496
		2508	2514	2517	2550	2556	2559	2565	2568	2571	2574	2580	2583	2589
		2592	2595	2598	2601	2604	2607	2610	2617	2620	2623	2626	2629	2632
		2635	2641	2644	2650	2653	2659	2662	2668	2671	2677	2680	2683	2689
		2692	2698	2701	2704	2707	2710	2713	2716	2719	2722	2725	2728	2731
		2734	2737	2740	2743	2746	2749	2752	2755	2765	2768	2771	2777	2780
		2783	2786	2792	2798	2804	2813	2816	2819	2822	2825	2828	2831	2834
		2837	2840	2843	2846	2849	2852	2855	2858	2861	2864	2867	2870	2873
		2876	2879	2882	2885	2888	2891	2894	2897	2900	2903	2906	2909	2912
		2915	2918	2921	2924	2927	2930	2933	2936	2945	2948	2960	2963	2966
		2969	2978	2981	2984	2991	2994	2997	3000	3003	3006	3009	3030	3033
		3039	3042	3045	3048	3051	3054	3057	3060	3066	3069	3072	3075	3078
		3081	3084	3090	3093	3096	3099	3102	3105	3108	3111	3114	3117	3120
		3123	3126	3129	3133	3137	3140	3143	3146	3149	3152	3155	3158	3161
		3164	3167	3170	3173	3176	3179	3182	8016#					
DH41	041347	2363	2547	8237#										
DH42	041470	2332	2339	2367	2378	2384	2418	2427	2442	2448	2454	2457	2505	2511
		2520	2544	2553	2577	2586	2951	2954	2957	3012	3015	3018	3021	3024
		3027	3185	3188	8253#									
DH43	041612	2371	2375	2381	2387	2400	2406	2409	2412	2415	2421	2424	2436	2466
		2475	2481	2499	2502	2523	2526	2529	2532	2535	2538	2541	8269#	
DH46	042066	2638	8301#											
DH51	042301	2390	2394	2397	8328#									
DISPLA=	177570	1749#	7356*	7380*										
DOMFPT	032076	7087	7089#											
DONE7	032126	7088	7094	7098#										
DT1	036746	2265	2268	2271	8000#									
DT14	037540	2299	2302	2563	2989	8074#								
DT163	046734	2615	8758#											
DT201	050210	2648	2657	2666	2675	2687	2696	8883#						



CEKBEO 11/70-74MP CPU #2  
CEKBBE.P11 10-MAR-80 09:31

MACY11 30A(1052) 10-MAR-80 09:32 PAGE 205  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0203

EM127	045423	2528	8625#				
EM13	037377	2293	8054#				
EM130	045475	2531	2537	8633#			
EM134	045536	2543	8639#				
EM135	045605	2546	8646#				
EM136	045614	2549	8648#				
EM14	037427	2296	8059#				
EM140	045647	2555	2579	8653#			
EM141	045704	2558	8658#				
EM142	045743	2561	8664#				
EM143	045771	2564	8668#				
EM144	046025	2567	8673#				
EM145	046054	2570	8677#				
EM146	046113	2573	8683#				
EM147	046152	2576	8689#				
EM15	037552	2300	8076#				
EM151	046203	2582	8694#				
EM152	046216	2585	8696#				
EM155	046245	2594	8700#				
EM156	046276	2597	8705#				
EM157	046416	2600	8719#				
EM16	037663	2303	2588	8089#			
EM160	046447	2603	8724#				
EM161	046500	2606	8729#				
EM162	046575	2609	8740#				
EM163	046626	2612	8745#				
EM164	046746	2616	8760#				
EM165	046777	2619	8765#				
EM166	047061	2622	8774#				
EM167	047166	2625	8786#				
EM17	037717	2306	8094#				
EM170	047230	2628	8792#				
EM171	047272	2631	8798#				
EM172	047413	2634	8812#				
EM174	047666	2640	8843#				
EM175	047750	2643	2652	2661	2670	2691	8852#
EM176	047765	2646	8855#				
EM177	050027	2649	8861#				
EM2	036764	2266	8003#				
EM20	037774	2309	8102#				
EM201	050111	2655	8870#				
EM202	050220	2658	8885#				
EM204	050302	2664	8894#				
EM205	050350	2667	8901#				
EM207	050433	2673	8910#				
EM21	040065	2312	8112#				
EM210	050501	2676	8917#				
EM211	050547	2679	8924#				
EM212	050630	2682	8933#				
EM213	050674	2685	8939#				
EM214	050742	2688	8946#				
EM216	051024	2694	8955#				
EM217	051072	2697	8962#				
EM22	040105	2315	8115#				
EM220	051126	2700	8967#				
EM221	051207	2703	8976#				

CEKBEO 11/70-74MP CPU #2  
CEKBBE.P11 10-MAR-80 09:31

MACY11 30A(1052) 10-MAR-80 09:32 PAGE 206  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0204

EM222	051243	2706	8981#	
EM223	051323	2709	8990#	
EM224	051344	2712	8993#	
EM225	051513	2715	9011#	
EM226	051624	2718	9024#	
EM227	051704	2721	9032#	
EM23	040202	2318	8126#	
EM230	051765	2724	9041#	
EM232	052252	2730	9073#	
EM233	052324	2733	9080#	
EM234	052421	2736	9092#	
EM235	052526	2739	9105#	
EM237	052673	2745	9123#	
EM24	040272	2321	8136#	
EM240	052766	2748	9134#	
EM241	053070	2751	9146#	
EM242	053201	2754	9159#	
EM243	053335	2757	9176#	
EM244	053502	2761	9195#	
EM245	053564	2764	9205#	
EM246	053657	2767	9216#	
EM247	054000	2770	9230#	
EM25	040325	2324	2380	8141#
EM250	054111	2773	9244#	
EM251	054272	2776	9266#	
EM252	054437	2779	9284#	
EM253	054507	2782	9291#	
EM254	054557	2785	9298#	
EM255	054647	2788	9308#	
EM256	055007	2791	9325#	
EM257	055044	2794	9330#	
EM26	040444	2328	8157#	
EM260	055115	2797	9337#	
EM261	055160	2800	9343#	
EM262	055232	2803	9350#	
EM263	055274	2806	7459	9356#
EM264	056106	2809	9434#	
EM265	056131	2812	9439#	
EM266	056447	2815	9476#	
EM267	056565	2818	9490#	
EM27	040504	2331	8163#	
EM270	056645	2821	9499#	
EM271	056743	2824	9511#	
EM272	057073	2827	9527#	
EM273	057132	2830	9533#	
EM274	057200	2833	9540#	
EM275	057220	2836	9543#	
EM276	057570	2839	9584#	
EM277	057621	2842	9589#	
EM3	037005	2269	8006#	
EM30	040546	2335	8169#	
EM300	057675	2845	9597#	
EM301	057764	2848	9607#	
EM302	060051	2851	9616#	
EM303	060116	2854	9623#	
EM304	060247	2857	9639#	

EM305	060334	2860	9648#
EM306	060422	2863	9657#
EM307	060544	2866	9671#
EM31	040604	2338	8174#
EM310	060617	2869	9679#
EM311	060671	2872	9687#
EM312	060742	2875	9694#
EM313	061066	2878	9709#
EM314	061136	2881	9716#
EM315	061247	2884	9729#
EM316	061371	2887	9744#
EM317	061504	2890	9758#
EM32	040646	2341	8180#
EM320	061524	2893	9761#
EM321	061642	2896	9774#
EM322	061712	2899	9781#
EM323	061767	2902	9789#
EM324	062077	2905	9803#
EM325	062160	2908	9812#
EM326	062235	2911	9820#
EM327	062316	2914	9829#
EM33	040712	2344	8186#
EM330	062461	2917	9846#
EM331	062572	2920	9859#
EM332	062705	2923	9872#
EM333	062766	2926	9881#
EM334	063021	2929	9886#
EM335	063054	2932	9891#
EM336	063111	2935	9896#
EM337	063160	2938	9903#
EM34	040730	2347	8189#
EM340	063272	2941	9918#
EM341	063344	2944	9925#
EM342	063411	2947	9932#
EM343	063456	2950	9939#
EM344	063512	2953	9944#
EM345	063547	2956	9949#
EM346	063602	2959	9954#
EM347	063662	2962	9962#
EM35	040763	2350	8194#
EM350	063742	2965	9970#
EM351	064017	2968	9978#
EM352	064074	2971	9986#
EM353	064141	2974	9993#
EM354	064206	2977	10000#
EM355	064240	2980	10005#
EM356	064317	2983	10013#
EM357	064352	2986	10018#
EM36	041101	2353	8208#
EM360	064515	2990	10037#
EM361	064574	2993	10045#
EM362	064644	2996	10052#
EM363	064701	2999	10057#
EM364	064751	3002	10064#
EM365	065033	3005	10073#
EM366	065070	3008	10078#

EM367	065124	3011	10083#
EM37	041164	2356	8217#
EM370	065240	3014	10096#
EM371	065365	3017	10111#
EM372	065472	3020	10123#
EM373	065622	3023	10139#
EM374	065734	3026	10152#
EM375	066024	3029	10162#
EM376	066177	3032	10181#
EM4	037032	2272	8010#
EM40	041234	2359	8224#
EM400	066227	3038	10186#
EM401	066330	3041	10198#
EM402	066375	3044	10205#
EM403	066437	3047	10211#
EM405	066543	3053	10224#
EM406	066661	3056	10239#
EM41	041304	2362	8231#
EM410	067011	3062	10255#
EM411	067114	3065	10267#
EM412	067163	3068	10274#
EM413	067240	3071	10282#
EM414	067313	3074	10290#
EM415	067374	3077	10299#
EM416	067457	3080	10308#
EM417	067532	3083	10316#
EM42	041440	2366	8249#
EM420	067623	3086	10327#
EM421	067700	3089	10335#
EM422	067771	3092	10346#
EM423	070201	3095	10371#
EM424	070273	3098	10382#
EM425	070352	3101	10390#
EM426	070371	3104	10393#
EM427	070455	3107	10402#
EM43	041560	2370	8264#
EM430	070561	3110	10414#
EM431	070624	3113	10420#
EM432	070670	3116	10426#
EM433	070754	3119	10436#
EM435	071022	3125	10443#
EM436	071142	3128	10457#
EM437	071205	3132	10463#
EM44	041740	2374	8286#
EM440	071255	3136	10470#
EM441	071330	3139	10478#
EM442	071401	3142	10485#
EM443	071456	3145	10493#
EM444	071474	3148	10496#
EM445	071512	3151	10499#
EM446	071530	3154	10502#
EM447	071546	3157	10505#
EM45	042004	2377	8292#
EM450	071564	3160	10508#
EM451	071602	3163	10511#
EM452	071620	3166	10514#



EM453	071636	3169	10517#																
EM454	071704	3172	10524#																
EM455	071771	3175	10533#																
EM456	072107	3178	10547#																
EM457	072163	3181	10555#																
EM46	042033	2637	8296#																
EM460	072214	3184	10560#																
EM461	072260	3187	10566#																
EM47	042162	2383	8313#																
EM5	037130	2275	8022#																
EM50	042207	2386	8317#																
EM51	042241	2389	8322#																
EM52	042462	2393	8349#																
EM53	042535	2396	8357#																
EM54	042570	2399	8362#																
EM55	042606	2402	2438	2591			8365#												
EM56	042664	2405	8373#																
EM57	042735	2408	8380#																
EM6	037216	2278	8031#																
EM60	043005	2411	8387#																
EM61	043106	2414	8399#																
EM62	043164	2417	8407#																
EM63	043226	2420	8413#																
EM64	043266	2423	8419#																
EM65	043326	2426	8425#																
EM66	043363	2429	8430#																
EM67	043414	2432	8435#																
EM7	037251	2281	8036#																
EM70	043446	2435	8440#																
EM703	072323	10572#																	
EM704	072353	10577#																	
EM706	07240	10581#																	
EM710	072432	4941	10586#																
EM711	072452	4967	10589#																
EM712	072472	5029	10592#																
EM713	072517	5091	10596#																
EM715	072544	5124	10600#																
EM717	072571	7119	10604#																
EM72	043464	2441	8443#																
EM720	072614	7124	7146	10608#															
EM721	072753	7131	7140	10625#															
EM722	073011	7151	10631#																
EM724	073050	10637#																	
EM725	073121	4952	10644#																
EM73	043526	2444	8449#																
EM74	043544	2447	8452#																
EM75	043606	2450	8458#																
EM76	043673	2453	8467#																
EM77	043716	2456	8471#																
ENDBR5	016520	4973	4980#																
ENDGET	026374	6410	6412	6416#															
ENDKB	023202	5838	5841	5843#															
ERRVEC	000004	1845#	3255*	3256*	3339*	3359*	4469*	4480*	4593*	4599*	4626*	4659*	4688*	4692*					
		4696*	4723*	4727*	4730*	4759*	4768*	4773*	4800*	4804*	4807*	4832*	4835*	4837*					
		4856*	4867*	4873*	4883*	4915*	4918*	4921*	4924*	4944*	4949*	4955*	4961*	4963*					
		5171*	5172*	5186*	5191*	5195*	5200*	5206*	5212*	5219*	5351*	5354*	5359*	5363*					



ITEM43	001610	2370#
ITEM44	001616	2374#
ITEM45	001624	2377#
ITEM46	001632	2380#
ITEM47	001640	2383#
ITEM5	001324	2275#
ITEM50	001646	2386#
ITEM51	001654	2389#
ITEM52	001662	2393#
ITEM53	001670	2396#
ITEM54	001676	2399#
ITEM55	001704	2402#
ITEM56	001712	2405#
ITEM57	001720	2408#
ITEM6	001332	2278#
ITEM60	001726	2411#
ITEM61	001734	2414#
ITEM62	001742	2417#
ITEM63	001750	2420#
ITEM64	001756	2423#
ITEM65	001764	2426#
ITEM66	001772	2429#
ITEM67	002000	2432#
ITEM7	001340	2281#
ITEM70	002006	2435#
ITEM71	002014	2438#
ITEM72	002022	2441#
ITEM73	002030	2444#
ITEM74	002036	2447#
ITEM75	002044	2450#
ITEM76	002052	2453#
ITEM77	002060	2456#
ITE100	002066	2459#
ITE101	002074	2462#
ITE102	002102	2465#
ITE103	002110	2468#
ITE104	002116	2471#
ITE105	002124	2474#
ITE106	002132	2477#
ITE107	002140	2480#
ITE110	002146	2483#
ITE111	002154	2486#
ITE112	002162	2489#
ITE113	002170	2492#
ITE114	002176	2495#
ITE115	002204	2498#
ITE116	002212	2501#
ITE117	002220	2504#
ITE120	002226	2507#
ITE121	002234	2510#
ITE122	002242	2513#
ITE123	002250	2516#
ITE124	002256	2519#
ITE125	002264	2522#
ITE126	002272	2525#
ITE127	002300	2528#

ITE130	002306	2531#
ITE131	002314	2534#
ITE132	002322	2537#
ITE133	002330	2540#
ITE134	002336	2543#
ITE135	002344	2546#
ITE136	002352	2549#
ITE137	002360	2552#
ITE140	002366	2555#
ITE141	002374	2558#
ITE142	002402	2561#
ITE143	002410	2564#
ITE144	002416	2567#
ITE145	002424	2570#
ITE146	002432	2573#
ITE147	002440	2576#
ITE150	002446	2579#
ITE151	002454	2582#
ITE152	002462	2585#
ITE153	002470	2588#
ITE154	002476	2591#
ITE155	002504	2594#
ITE156	002512	2597#
ITE157	002520	2600#
ITE160	002526	2603#
ITE161	002534	2606#
ITE162	002542	2609#
ITE163	002550	2612#
ITE164	002556	2616#
ITE165	002564	2619#
ITE166	002572	2622#
ITE167	002600	2625#
ITE170	002606	2628#
ITE171	002614	2631#
ITE172	002622	2634#
ITE173	002630	2637#
ITE174	002636	2640#
ITE175	002644	2643#
ITE176	002652	2646#
ITE177	002660	2649#
ITE200	002666	2652#
ITE201	002674	2655#
ITE202	002702	2658#
ITE203	002710	2661#
ITE204	002716	2664#
ITE205	002724	2667#
ITE206	002732	2670#
ITE207	002740	2673#
ITE210	002746	2676#
ITE211	002754	2679#
ITE212	002762	2682#
ITE213	002770	2685#
ITE214	002776	2688#
ITE215	003004	2691#
ITE216	003012	2694#
ITE217	003020	2697#

ITE220	003026	2700#
ITE221	003034	2705#
ITE222	003042	2706#
ITE223	003050	2709#
ITE224	003056	2712#
ITE225	003064	2715#
ITE226	003072	2718#
ITE227	003100	2721#
ITE230	003106	2724#
ITE231	003114	2727#
ITE232	003122	2730#
ITE233	003130	2733#
ITE234	003136	2736#
ITE235	003144	2739#
ITE236	003152	2742#
ITE237	003160	2745#
ITE240	003166	2748#
ITE241	003174	2751#
ITE242	003202	2754#
ITE243	003210	2757#
ITE244	003216	2761#
ITE245	003224	2764#
ITE246	003232	2767#
ITE247	003240	2770#
ITE250	003246	2773#
ITE251	003254	2776#
ITE252	003262	2779#
ITE253	003270	2782#
ITE254	003276	2785#
ITE255	003304	2788#
ITE256	003312	2791#
ITE257	003320	2794#
ITE260	003326	2797#
ITE261	003334	2800#
ITE262	003342	2803#
ITE263	003350	2806#
ITE264	003356	2809#
ITE265	003364	2812#
ITE266	003372	2815#
ITE267	003400	2818#
ITE270	003406	2821#
ITE271	003414	2824#
ITE272	003422	2827#
ITE273	003430	2830#
ITE274	003436	2833#
ITE275	003444	2836#
ITE276	003452	2839#
ITE277	003460	2842#
ITE300	003466	2845#
ITE301	003474	2848#
ITE302	003502	2851#
ITE303	003510	2854#
ITE304	003516	2857#
ITE305	003524	2860#
ITE306	003532	2863#
ITE307	003540	2866#

ITE310	003546	2869#
ITE311	003554	2872#
ITE312	003562	2875#
ITE313	003570	2878#
ITE314	003576	2881#
ITE315	003604	2884#
ITE316	003612	2887#
ITE317	003620	2890#
ITE320	003626	2893#
ITE321	003634	2896#
ITE322	003642	2899#
ITE323	003650	2902#
ITE324	003656	2905#
ITE325	003664	2908#
ITE326	003672	2911#
ITE327	003700	2914#
ITE330	003706	2917#
ITE331	003714	2920#
ITE332	003722	2923#
ITE333	003730	2926#
ITE334	003736	2929#
ITE335	003744	2932#
ITE336	003752	2935#
ITE337	003760	2938#
ITE340	003766	2941#
ITE341	003774	2944#
ITE342	004002	2947#
ITE343	004010	2950#
ITE344	004016	2953#
ITE346	004032	2959#
ITE347	004040	2962#
ITE350	004046	2965#
ITE351	004054	2968#
ITE352	004062	2971#
ITE353	004070	2974#
ITE354	004076	2977#
ITE355	004104	2980#
ITE356	004112	2983#
ITE357	004120	2986#
ITE360	004126	2990#
ITE361	004134	2993#
ITE362	004142	2996#
ITE363	004150	2999#
ITE364	004156	3002#
ITE365	004164	3005#
ITE366	004172	3008#
ITE367	004200	3011#
ITE370	004206	3014#
ITE371	004214	3017#
ITE372	004222	3020#
ITE373	004230	3023#
ITE374	004236	3026#
ITE375	004244	3029#
ITE376	004252	3032#
ITE377	004260	3035#
ITE400	004266	3038#

ITE401	004274	3041#							
ITE402	004302	3044#							
ITE403	004310	3047#							
ITE404	004316	3050#							
ITE405	004324	3053#							
ITE406	004332	3056#							
ITE407	004340	3059#							
ITE410	004346	3062#							
ITE411	004354	3065#							
ITE412	004362	3068#							
ITE413	004370	3071#							
ITE414	004376	3074#							
ITE415	004404	3077#							
ITE416	004412	3080#							
ITE417	004420	3083#							
ITE420	004426	3086#							
ITE421	004434	3089#							
ITE422	004442	3092#							
ITE423	004450	3095#							
ITE424	004456	3098#							
ITE425	004464	3101#							
ITE426	004472	3104#							
ITE427	004500	3107#							
ITE430	004506	3110#							
ITE431	004514	3113#							
ITE432	004522	3116#							
ITE433	004530	3119#							
ITE434	004536	3122#							
ITE435	004544	3125#							
ITE436	004552	3128#							
ITE437	004560	3132#							
ITE440	004566	3136#							
ITE441	004574	3139#							
ITE442	004602	3142#							
ITE443	004610	3145#							
ITE444	004616	3148#							
ITE445	004624	3151#							
ITE446	004632	3154#							
ITE447	004640	3157#							
ITE450	004646	3160#							
ITE451	004654	3163#							
ITE452	004662	3166#							
ITE453	004670	3169#							
ITE454	004676	3172#							
ITE455	004704	3175#							
ITE456	004712	3178#							
ITE457	004720	3181#							
ITE460	004726	3184#							
ITE461	004734	3187#							
KBDONE	023234	5832	5849	5851#					
KBTST	023142	5717	5831#						
KB11E	001272	2239#	5833*	5837*	5846	5892	6884	6897	7086
KB11EM	001273	2240#							
KDPAR0=	172360	2022#							
KDPAR1=	172362	2023#							
KDPAR2=	172364	2024#							





MAPH15= 170266	2068#	
MAPH16= 170272	2070#	
MAPH17= 170276	2072#	
MAPH2 = 170212	2110#	
MAPH20= 170302	2074#	
MAPH21= 170306	2076#	
MAPH22= 170312	2078#	
MAPH23= 170316	2080#	
MAPH24= 170320	2082#	
MAPH25= 170326	2084#	
MAPH26= 170332	2086#	
MAPH27= 170336	2088#	
MAPH3 = 170216	2112#	
MAPH30= 170342	2090#	
MAPH31= 170346	2092#	
MAPH32= 170352	2094#	
MAPH33= 170356	2096#	
MAPH34= 170362	2098#	
MAPH35= 170366	2100#	
MAPH36= 170372	2102#	
MAPH37= 170376	2104#	
MAPH4 = 170222	2114#	
MAPH5 = 170226	2116#	
MAPH6 = 170232	2118#	
MAPH7 = 170236	2120#	
MAPL0 = 170200	2105#	3191*
MAPL00= 170200	2041#	2105
MAPL01= 170204	2043#	2107
MAPL02= 170210	2045#	2109
MAPL03= 170214	2047#	2111
MAPL04= 170220	2049#	2113
MAPL05= 170224	2051#	2115
MAPL06= 170230	2053#	2117
MAPL07= 170234	2055#	2119
MAPL1 = 170204	2107#	3193*
MAPL10= 170240	2057#	
MAPL11= 170244	2059#	
MAPL12= 170250	2061#	
MAPL13= 170254	2063#	
MAPL14= 170260	2065#	
MAPL15= 170264	2067#	
MAPL16= 170270	2069#	
MAPL17= 170274	2071#	
MAPL2 = 170210	2109#	
MAPL20= 170300	2073#	
MAPL21= 170304	2075#	
MAPL22= 170310	2077#	
MAPL23= 170314	2079#	
MAPL24= 170320	2081#	
MAPL25= 170324	2083#	
MAPL26= 170330	2085#	
MAPL27= 170334	2087#	
MAPL3 = 170214	2111#	
MAPL30= 170340	2089#	
MAPL31= 170344	2091#	
MAPL32= 170350	2093#	

CEKBEO 11/70-74MP CPU #2  
CEKBEE.P11 10-MAR-80 09:31

MACY11 30A(1052) 10-MAR-80 09:32 K 1 PAGE 218  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0216

MAPL33=	170354	2095#													
MAPL34=	170360	2097#													
MAPL35=	170364	2099#													
MAPL36=	170370	2101#													
MAPL37=	170374	2103#													
MAPL4 =	170220	2113#													
MAPL5 =	170224	2115#													
MAPL6 =	170230	2117#													
MAPL7 =	170234	2119#													
MEMERR=	177744	1865#	3260*	7281	7294*										
MFPT =	000007	2243#	5835	7091											
MFPTTR	023174	5834	5840#												
MMRO =	177572	1890#	1894												
MMR1 =	177574	1891#	1895												
MMR2 =	177576	1892#	1896												
MMR3 =	172516	1893#	1897												
MMSET	032126	7085	7099#												
MMVEC =	000250	1858#													
MSG1	036440	5845	7960#												
MSG3	036477	5848	7966#												
MSG5	036544	5850	7973#												
MSG6	036550	7974#													
NEXTEN	034224	7489#	7537												
NEXTTS	001270	2238#	3289*	3334*	3402*	3452*	3585*	3618*	3639*	3665*	3796*	3846*	3940*	4021*	
		4307*	4346*	4386*	4428*	4466*	4499*	4545*	4616*	4649*	4682*	4716*	4753*	4793*	
		4831*	4908*	4913	5020*	5050*	5084*	5117*	5343*	5447*	5525*	5625*	5700*	5717*	
		5868*	5923*	5968*	6428*	6527*	6586*	6616*	6675*	6712*	6773*	6805*	6888*	6975	
		6990*	7020*	7085*	7114*	7267	7295	7610							
		4892	4906#												
PDINTE	016056	1747#	4510*	4511	4516*	4517	4519	4526*	4528	4558*	4559*	4562*	4563*	4565*	
PIRQ	177772	4569*	4570*	4572*	4577*	4578*	4580*	4586*	4587*	4589*	4592*	4596*	4598*	4603*	
		4621*	4622*	4625*	4630*	4633*	4634*	4636	4637*	4654*	4655*	4658*	4662*	4665*	
		4666*	4668	4669*	4687*	4691*	4695*	4699*	4700*	4703	4704*	4721*	4722*	4726*	
		4731*	4734*	4735*	4738	4739*	4758*	4761*	4762*	4765*	4769*	4772*	4776*	4777*	
		4780	4781*	4798*	4799*	4803*	4809*	4811*	4812*	4814	4815*	4875*	4878*	4880*	
		6011*	6016*	6025*	6027*	6033*	6038*	6043*	6049*	6054*	6059*	6065*	6077*	6083*	
		6092*	6098*	6104*	6134*	6139*	6149*	6155*	6161*	6174*	6180*	6185*	6205*	6211*	
		6217*	6222*	6228*	6235*	6243*	6250*	6259*	6265*	6274*	6280*	6292*	6299*	6309*	
		6313*	6327*	6333*	6342*	6348*	6357*	6361*	6372*	6376*	6382*	6404*	7057*	7063*	
		7068*													
PIRQVE=	000240	1857#	4550*	4553*	4555*	4561*	4568*	4576*	4585*	4619*	4631*	4651*	4663*	4684*	
		4697*	4718*	4732*	4755*	4760*	4774*	4795*	4808*	4868*	4871*	5978*	5986*	6005*	
		6024*	6041*	6057*	6074*	6090*	6132*	6147*	6165*	6196*	6219*	6240*	6257*	6272*	
		6289*	6306*	6325*	6340*	6354*	6369*	7055*							
		2125#	5000*	6399*											
PLKC =	172544	2236#	4956	4959											
PLKSTA	001264	2237#	4958												
PLKVEC	001266	1779#													
PR0 =	000000	1780#													
PR1 =	000040	1781#	2215	3305											
PR2 =	000100	1782#													
PR3 =	000140	1783#													
PR4 =	000200	1784#	2216	3297											
PR5 =	000240	1785#													
PR6 =	000300	1786#	3252	3256	3258	3398	4390	4548	4555	4837	4873	5038	5067	5103	
PR7 =	000340	5136	5168	5171	5313	5348	5351	5724	5820	5871	5883	5894	5906	5984	











\$LPERR 001110	2180#	3226*	3337*	3394*	3454*	3490*	3515*	3530*	3552*	3696*	3723*	3739*	3754*
	3769*	3888*	3915*	4041*	4108*	4118*	4151*	4171*	4182*	4202*	4220*	4239*	4258*
	4277*	4362*	4388*	4502*	4629*	4661*	4694*	4729*	4771*	4806*	4838*	4857*	4866*
	5164*	5173*	5187*	5196*	5207*	5221*	5229*	5251*	5258*	5266*	5277*	5283*	5296*
	5307*	5352*	5378*	5387*	5398*	5410*	5427*	5647*	5674*	5692*	5725*	5733*	5739*
	5745*	5754*	5762*	5773*	5782*	5788*	5797*	5806*	5813*	5876*	5898*	6003*	6022*
	6039*	6055*	6073*	6088*	6108*	6130*	6145*	6164*	6195*	6218*	6238*	6255*	6270*
	6288*	6305*	6323*	6338*	6353*	6368*	6444*	6458*	6472*	6500*	6511*	6618*	6629*
	6641*	6653*	6749*	6777*	6892*	6910*	6924*	6937*	6950*	7024*	7035*	7053*	7337
	7353*	7359	7403										
\$MXCNT 033546	7350	7359#											
\$NULL 001146	2194#	7715	7744										
\$NWTST- 000001	3265#	3267	3319#	3321	3377#	3379	3438#	3440	3571#	3573	3607#	3609	3628#
	3630	3650#	3652	3783#	3785	3820#	3822	3927#	3929	3959#	3961	4290#	4292
	4333#	4335	4376#	4378	4416#	4418	4453#	4455	4487#	4489	4530#	4532	4606#
	4608	4639#	4641	4672#	4674	4706#	4708	4741#	4743	4783#	4785	4818#	4820
	5012#	5014	5076#	5078	5109#	5111	5142#	5144	5234#	5236	5325#	5327	5433#
	5435	5489#	5491	5612#	5614	5710#	5712	5853#	5855	5915#	5917	5934#	5936
	6418#	6420	6520#	6522	6579#	6581	6609#	6611	6667#	6669	6705#	6707	6763#
	6765	6795#	6797	6846#	6848	6862#	6864	6978#	6980	7011#	7013	7071#	7073
	7100#	7102											
\$SOCNT 036016	7777*	7806*	7819#										
\$SOMODE 036020	7772*	7776*	7781	7784*	7795*	7821#							
\$SOVER 033532	7314	7330	7338	7348	7356#								
\$SPASS 001100	2175#	3261	4909	5027	5089	5122	7115	7177*	7178*	7189	7231	7344	7360
\$SPOWER 036430	7952	7957#											
\$SPR2 001212	2215#	8000											
\$SPR5 001214	2216#	8000											
\$SPWRDN 036300	3207	4594	5448	7925#	7949								
\$SPWRMG 036416	7952#												
\$SPWRUP 036346	7934	7939#											
\$SQUES 001202	2210#	7409											
\$RDCHR= ***** U	7920												
\$RDDEC= ***** U	7920												
\$RDLIN= ***** U	7920												
\$RDOCT= ***** U	7920												
\$REGAD 001152	2198#												
\$REGO 001154	2200#	3420*	3427*	3683*	3691*	3705*	3749*	3766*	3778*	3810*	3815*	3858*	3896*
	3953*	4067*	4087*	4097*	4133*	4140*	4214*	4228*	4247*	4266*	4318*	4359*	4887*
	5225*	5424*	5610*	6740*	7001*	7489	7538	7624*	7661	8074	8247	8283	8546
\$REG1 001156	2201#	3421*	3428*	3506*	3563*	3692*	3706*	3750*	3767*	3779*	3811*	3816*	3859*
	3897*	3954*	4068*	4098*	4098*	4134*	4141*	4216*	4229*	4248*	4267*	4360*	6743*
	7656*	7667	8074	R15C	8283	8346							
\$REG2 001160	2202#												
\$RTRN 032652	3213	3215*	3220*	4481	5971	7048	7052	7205	7226#				
\$SAVRE= ***** U	7920												
\$SAVR6 036426	7933*	7939	7940*	7941*	7956#								
\$SCOPE 033302	3201	6971	7312#										
\$SETUP= 000037	2127#	3201	3203	3205	3207	3209	3210	3211	3213	3225	3231	7175	7395
\$STUP = 177777	2127#												
\$SVLAD 033504	7322	7351#											
\$SVPIC 000204	2159#	2164											
\$SWR - 177400	1709	1710#	1718	1719	1720	1721	1722	1723	1724	1725	2207	2208	2209
	3210	3211	3213	3225	3226	3288	3332	3392	3452	3584	3617	3638	3665
	3795	3846	3939	4021	4306	4346	4385	4427	4465	4498	4544	4616	4649
	4682	4716	4753	4793	4831	5020	5083	5116	5163	5244	5343	5446	5516





CEKBBE0 11/70-74MP (PU #2)  
CFKBBE.P11 10-MAR-80 09:31

MACY11 30A(1052) 10-MAR-80 09:32 PAGE 226  
CROSS REFERENCE TABLE -- USER SYMBOLS

F 2

SFO 0224

\$XTSTR 033310	7316#												
\$\$DONE 034434	7496	7538#											
\$\$GET4- 000001	7203#	7210#											
\$\$TRP - 000002	7905#	7916	7917	7918	7919	7920							
\$\$TSTN 001210	2214#	7254*	7259	7282*	7287	7377*	7415	7417	7469	8000	8021	8074	8155
	8247	8262	8283	8311	8346	8758	8883	9193	9203	9263	9916		
\$OFILL 036017	7771*	7775*	7785	7820#									
- 073312	2130#	2134	2136#	2159	2160#	2162#	2164#	2172#	2213	3199	3225	3226	3236#
	7188#	7231	7233#	7246#	7252#	7274#	7280#	7359	7360	7409	7543#	7572#	7589#
	7602#	7660#	7744	7889#	7936	7955	7976#	7999#	8020#	8246#	8261#	8310#	8757#
	9192#	9202#	9427#	9915#	10647#	10648							



MSG136	5109#	5111													
MSG137	5142#	5144													
MSG140	5325#	5327													
MSG141	5433#	5435													
MSG142	5489#	5491													
MSG143	5612#	5614													
MSG144	5853#	5855													
MSG145	5915#	5917													
MSG146	5934#	5936													
MSG147	6418#	6420													
MSG150	6520#	6522													
MSG151	6579#	6581													
MSG152	6609#	6611													
MSG153	6667#	6669													
MSG154	6705#	6707													
MSG155	6763#	6765													
MSG156	6795#	6797													
MSG157	6846#	6848													
MSG160	6862#	6864													
MSG173	7100#	7102													
MS137A	5234#	5236													
MS145A	5710#	5712													
MS160A	6978#	6980													
MS160B	7011#	7013													
MULT	1#	2124#													
MYTAGS	1694#	2213													
NEWST	1#	1697#	2124#	3265	3319	3377	3438	3571	3607	3628	3650	3783	3820	3927	3959
	4290	4333	4376	4416	4453	4487	4530	4606	4639	4672	4706	4741	4783	4818	5012
	5076	5109	5142	5234	5325	5433	5489	5612	5710	5853	5915	5934	6418	6520	6579
	6609	6667	6705	6763	6795	6846	6862	6978	7011	7071	7100				
POP	1#	2124#	7876	7943											
PUSH	1#	2124#	7855	7927											
SAVEAD	1694#	3288	3333	3401	3584	3617	3638	3795	3939	4306	4385	4427	4465	4498	4544
	4906	5048	5083	5116	5446	5524	5699	5867	6427	6526	6674	6772			
SCOPE	1743#	3287	3331	3391	3451	3583	3616	3637	3664	3794	3845	3938	4020	4305	4345
	4384	4426	4464	4497	4543	4615	4648	4681	4715	4752	4792	4830	5019	5082	5115
	5162	5243	5342	5445	5515	5624	5716	5866	5922	5967	6426	6525	6585	6615	6673
	6711	6771	6804	6851	6881	6989	7019	7084	7113	7174					
SETTRA	7906#	7916	7917	7918	7919										
SETUP	1#	2124#	3195												
SKIP	1#	2124#	3262	3312	3367	3426	3568	3593	3625	3647	3777	3814	3925	3952	4287
	4328	4373	4411	4445	4527	4595	4635	4667	4702	4737	4779	4813	4892	4951	4962
	4968	5030	5092	5095	5106	5125	5128	5139	5232	5430	5609	5932	6388	6478	6481
	6517	6571	6603	6792	6859	6973									
SLASH	1#	2124#													
SPACE	2124#														
STARS	1#	1697#	2124#	2139	2165	2244	3228	3240	3265	3286	3319	3330	3377	3390	3438
	3450	3488	3513	3528	3550	3571	3582	3607	3615	3628	3636	3650	3663	3694	3721
	3737	3752	3783	3793	3820	3844	3886	3913	3927	3937	3959	4019	4039	4106	4116
	4149	4169	4180	4200	4218	4237	4256	4275	4290	4304	4333	4344	4376	4383	4416
	4425	4453	4463	4483	4485	4487	4496	4530	4542	4606	4614	4639	4647	4672	4680
	4706	4714	4741	4751	4783	4791	4818	4829	4894	4905	5012	5018	5076	5081	5109
	5114	5142	5161	5234	5242	5325	5341	5433	5444	5489	5514	5516	5523	5612	5623
	5710	5715	5752	5771	5780	5795	5804	5823	5852	5853	5865	5869	5875	5915	5921
	5934	5966	5995	6019	6036	6052	6067	6086	6102	6127	6143	6158	6183	6214	6231
	6253	6268	6283	6302	6318	6336	6351	6365	6418	6425	6441	6456	6470	6497	6509



.\$SUPR	1#			
.\$TRAP	1#	1697#	7297#	7890
.\$TYPB	1#			
.\$TYPD	1#	1697#	7297#	7822
.\$TYPE	1#	1697#	7671	
.\$TYPO	1#	1697#	7297#	7744
.1170	1#	1697#	1734	

. ABS. 073312 000

ERRORS DETECTED: 0

CEKBBE.BIN,CEKBBE.LST/CRF/SOL/NL:TOC CEKBBE.SML,CEKBBE.P11  
RUN-TIME: 83 97 10 SECONDS  
RUN-TIME RATIO: 397/191-2.0  
CORE USED: 35K (69 PAGES)