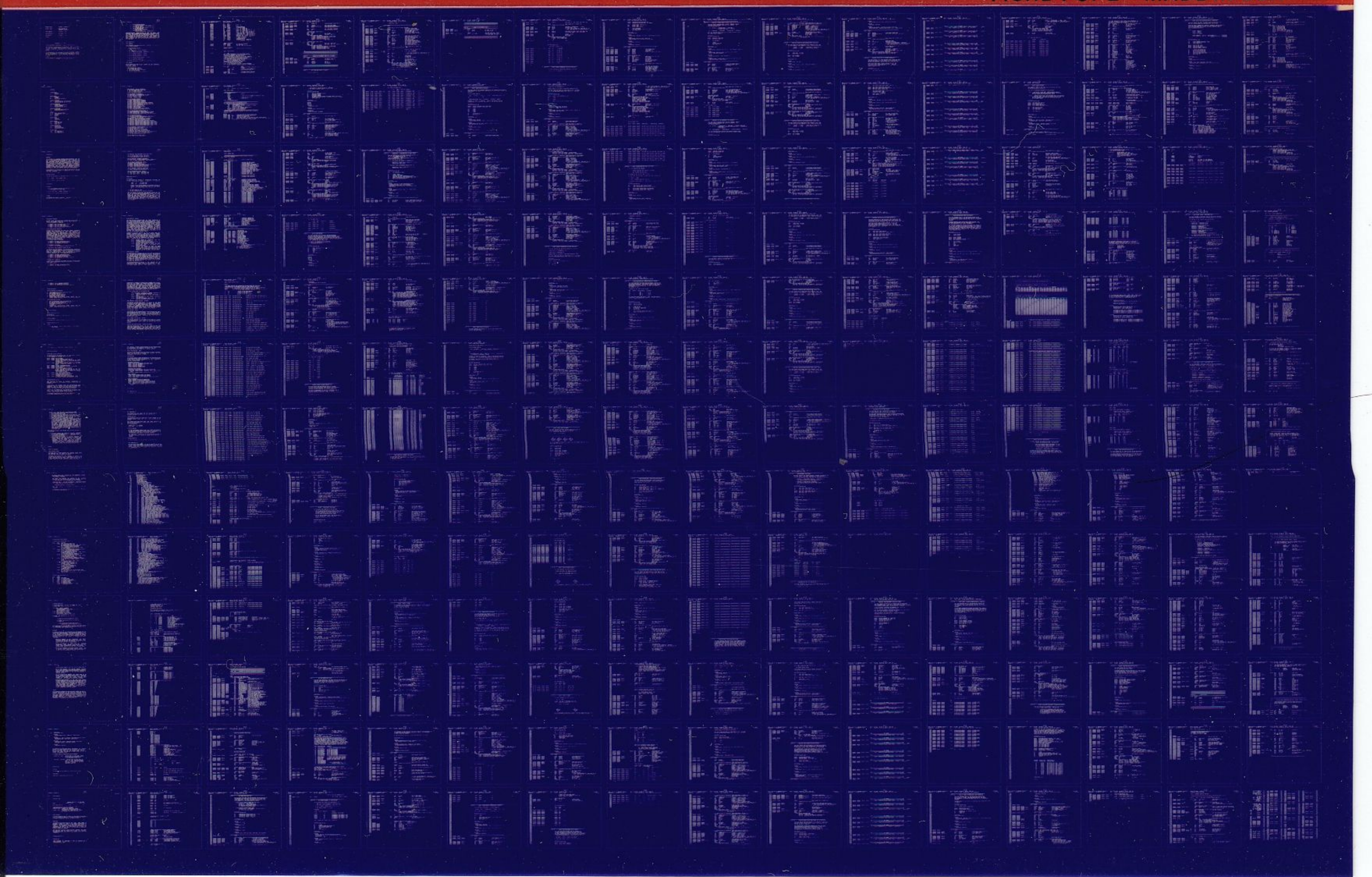
PDP-11/60

FP11-E HARDWARE DIAGNOSTIC
MD-11-DQFPE-A

EP-DQFPE-A-DL-A
COPYRIGHT © 1977
FICHE 1 OF 2

OCT 1977
digital
MADE IN USA

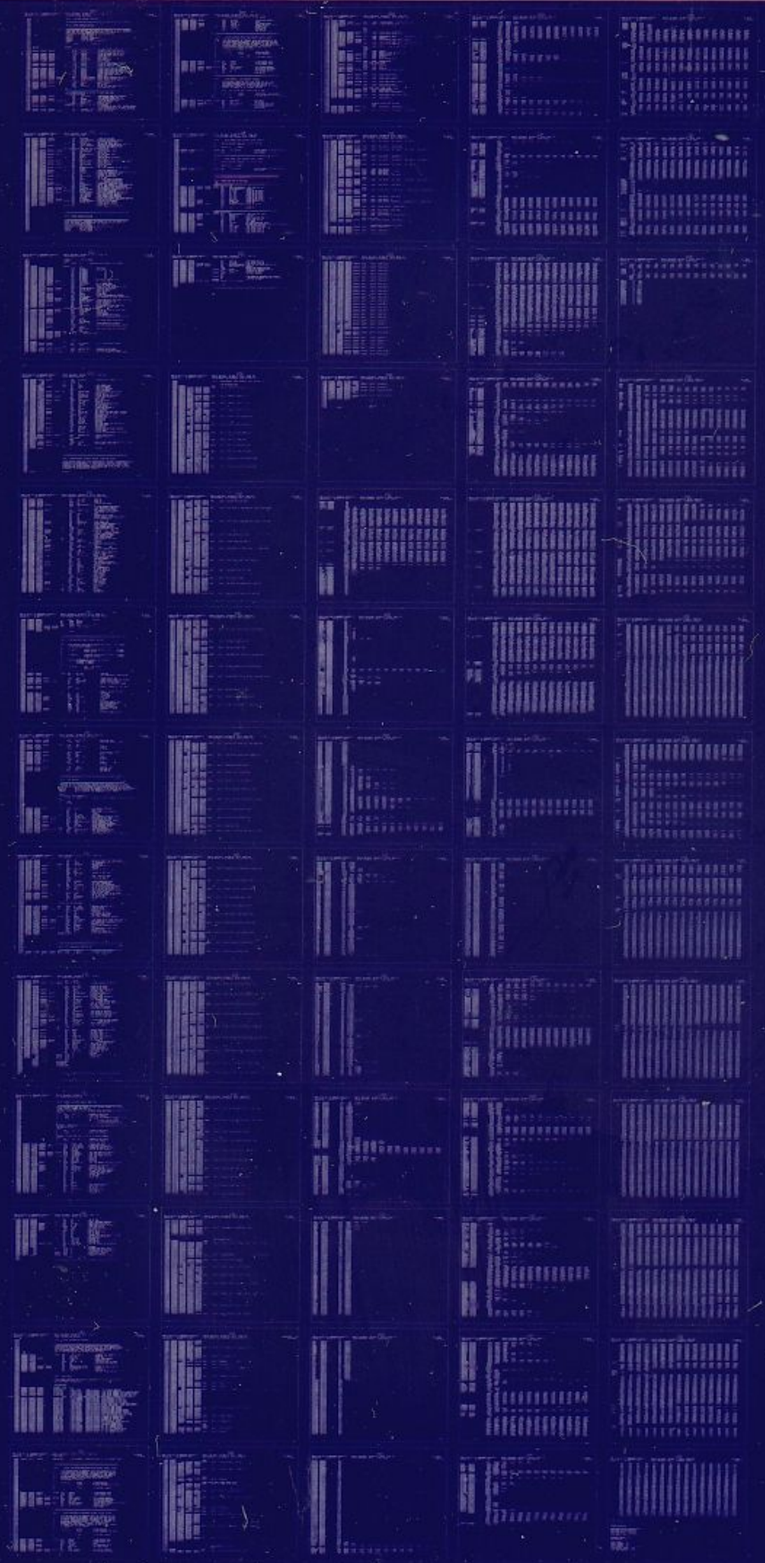


PDP-11/60

FP11-E HARDWARE DIAGNOSTIC
MD-11-DQFPE-A

EP-DQFPE-A-DL-A
COPYRIGHT © 1977
FICHE 2 OF 2

OCT 1977
digital
MADE IN USA



I D E N T I F I C A T I O N

PRODUCT CODE:	MAINDEC-11-DQFPE-A-D
PRODUCT NAME:	PDP-11/60 FP11-E HARDWARE DIAGNOSTIC
DATE CREATED:	September, 1977
LAST REVISION:	September, 1977
MAINTAINER:	Diagnostic Group
AUTHOR:	Don North

COPYRIGHT (C) 1977

DIGITAL EQUIPMENT CORPORATION, Maynard, Massachusetts

This software 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.

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 document.

DIGITAL assumes no responsibility for the use or reliability of its software on equipment not supplied by DIGITAL.

CONTENTS

1.0 INTRODUCTION
1.1 ABSTRACT
1.2 REVISION HISTORY

2.0 REQUIREMENTS
2.1 EQUIPMENT
2.2 STORAGE
2.3 PRELIMINARY PROGRAMS

3.0 LOADING PROCEDURE
3.1 LOADING/STARTING VIA PAPERTAPE
3.2 LOADING/STARTING VIA XXDP MEDIA

4.0 STARTING PROCEDURE
4.1 CONTROL SWITCH SETTINGS
4.2 STARTING ADDRESS
4.3 PROGRAM/OPERATOR ACTION

5.0 OPERATING PROCEDURE
5.1 OPERATIONAL SWITCH SETTINGS
5.2 PROGRAM/OPERATOR ACTION

6.0 ERRORS
6.1.1 ERROR MESSAGE FORMAT
6.1.2 FLOATING POINT DATA FORMAT
6.2 RECOVERY
6.3 CAUSES

7.0 RESTRICTIONS
7.1 STARTING
7.2 OPERATIONAL

8.0 MISCELLANEOUS
8.1 EXECUTION TIME
8.2 STACK POINTER
8.3 POWER FAIL

9.0 PROGRAM DESCRIPTION
9.1 ORGANIZATION
9.2 TEST DESCRIPTION
9.3 SUBROUTINE ABSTRACTS

10.0 ACT/APT/XXDP
10.1 ACT COMPATIBILITY
10.2 APT COMPATIBILITY
10.3 XXDP COMPATIBILITY

1.0 INTRODUCTION

1.1 ABSTRACT

THIS PROGRAM IS A HARDWARE ORIENTED MACRO DIAGNOSTIC FOR THE FP11-E "HOT" FLOATING POINT PROCESSOR OPTION OF THE PDP-11/60 CPU. THE SEQUENTIAL TEST STRUCTURE OF THIS DIAGNOSTIC HAS BEEN OPTIMIZED TOWARDS THE SPECIFIC FLOATING POINT PROCESSOR HARDWARE OF THE FP11-E, AND ITS INTERFACE WITH THE PDP-11/60 CPU. SPECIFIC ATTENTION HAS BEEN DIRECTED AT THE EXPONENT / FRACTION DATAPATH PARTITIONING, "ADD-/SUB-" INSTRUCTION IMPLEMENTATION, AND THE "MUL-" ROM MULTIPLIER NETWORK. DIAGNOSTIC ERROR PRINTOUTS, AND SPECIFIC HARDWARE INFORMATION PROVIDED AT EACH TEST HEADER, FACILITATE MODULE LEVEL FAULT RESOLUTION TO THE FP11-E UNIT OR HOST PDP-11/60 PROCESSOR.

THIS DIAGNOSTIC IS INTENDED TO BE USED IN CONJUNCTION WITH THE EXISTING FLOATING POINT INSTRUCTION TEST PROGRAMS "MD-11-DQFP[A/B/C/D]-*".

1.2 REVISION HISTORY

THIS SECTION DOCUMENTS ALL REVISIONS MADE TO THIS DIAGNOSTIC:

REV.	DATE	WHY / WHERE / WHO
AO	01-SEP-77	INITIAL RELEASE

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP-11/60 STANDARD COMPUTER WITH MINIMUM 16K WORDS OF ANY MEMORY TYPE (MOS, CORE),

2. DL11-W LINE CLOCK / CONSOLE INTERFACE, AND

3. FP11-E "HOT" FLOATING POINT PROCESSOR.

2.2 STORAGE

THE PROGRAM USES MEMORY 0-45520(8). THE UPPER 2.0K WORDS ARE RESERVED FOR THE XXDP MONITOR, IF EMPLOYED.

2.3 PRELIMINARY PROGRAMS

THE CPU, CACHE, AND MEMORY TEST PROGRAMS MUST BE RUN FIRST TO VERIFY THE CORRECT OPERATION OF THE BASE MACHINE. THE FOLLOWING SEQUENCE IS SUGGESTED:

(1) DQKDA-* PDP-11/60 BASIC LOGIC TESTS
(2) DQKDB-* PDP-11/60 TRAPS TEST
(3) DQKKA-* PDP-11/60 CACHE DIAGNOSTIC
(4) DZQMC-* PDP-11 0-124K MEMORY EXERCISER

2.3.1 "FAULT RESOLUTION" OPERATION

FOR BEST FAULT RESOLUTION, THE PDP-11/60 "WARM" (MICROCODE) FLOATING POINT INSTRUCTION SET TESTS MUST NOW BE RUN, IN "WARM"-ONLY MODE [IE, SWR=(xxxxx3)]. THIS VERIFIES THE CORRECT OPERATION OF THE BASE PROCESSOR FLOATING POINT SUPPORT MICROCODE; THIS MUST BE DONE PRIOR TO RUNNING ANY "HOT" FLOATING POINT TESTS, AS THE FP11-E UNIT RELIES HEAVILY ON THE BASE PROCESSOR FOR SUPPORT FUNCTIONS (OPERAND FETCH/STORE, ETC.). THE "DQFPE-*" DIAGNOSTIC ASSUMES THAT THE "WARM" FLOATING POINT PORTION OF THE BASE PROCESSOR (HARDWARE AND MICROCODE) IS FULLY OPERATIVE.
THE FOLLOWING ORDER IS REQUIRED:

(1) DQFPA-* FPU BASIC INSTRUCTION TESTS
(2) DQFPB-* FPU ADVANCED INSTRUCTION TESTS
(3) DQFPC-* FPU INSTRUCTION EXERCISER

THE FOLLOWING IS OPTIONAL:
(4) DQFPD-* FPU ADD/SUB/MUL/DIV RANDOM EXERCISER

AT THIS POINT, "MD-11-DQFPE-*" SHOULD BE RUN.

TO COMPLETE THE TEST SEQUENCE, THE FLOATING POINT INSTRUCTION SET TEST PROGRAMS SHOULD NOW BE RUN IN BOTH "WARM" AND "HOT" MODES [IE, SWR=(xxxxx0)]. THIS IS NECESSARY TO PROVIDE COMPLETE TESTING OF THE FP11-E UNIT INSTRUCTION EXECUTION AND EXCEPTION HANDLING LOGIC. THE SUGGESTED SEQUENCE IS:

(1) DQFPA-* FPU BASIC INSTRUCTION TESTS
(2) DQFPB-* FPU ADVANCED INSTRUCTION TESTS
(3) DQFPC-* FPU INSTRUCTION EXERCISER
(4) DQFPD-* FPU ADD/SUB/MUL/DIV RANDOM EXERCISER

2.3.2 "COVERAGE ONLY" OPERATION

TO OBTAIN FULL COVERAGE OF THE "WARM" AND "HOT" FLOATING POINT UNITS, ONLY THE FOLLOWING SHORTER TEST SEQUENCE IS NECESSARY, USING SWR=(xxxxx0):

(1) DQFPA-* FPU BASIC INSTRUCTION TESTS
(2) DQFPB-* FPU ADVANCED INSTRUCTION TESTS

(3) DQFPE-* FP11-E HARDWARE DIAGNOSTIC
(4) DQFPC-* FPU INSTRUCTION EXERCISER
(5) DQFPD-* FPU ADD/SUB/MUL/DIV RANDOM EXERCISER

3.0 LOADING PROCEDURE

3.1 LOADING/STARTING VIA PAPERTAPE

(1) LOAD PROGRAM INTO MEMORY USING ABS LOADER.
(2) LOAD ADDRESS 200(8).
(3) SET SWITCHES (SEE SECTION 5.1)
SR=(000000) IS WORST CASE TEST.
(4) PRESS "CNTRL/START" TO BEGIN.
(5) PROGRAM TYPES IDENTIFICATION HEADER (VERIFY THAT THE CORRECT PROGRAM HAS BEEN LOADED!), AND EXECUTION BEGINS.

3.2 LOADING/STARTING VIA XXDP

(1) BOOT THE APPLICABLE XXDP LOAD DEVICE (RK, TC, DP, ETC)
(2) SET THE SWITCHES AS DESIRED (SEE SECTION 5.1)
SR=(000000) IS WORST CASE TEST.
(3) FROM XXDP MONITOR MODE ("."), TYPE:
.R QFPEA0
(4) PROGRAM TYPES IDENTIFICATION HEADER (VERIFY THAT THE CORRECT PROGRAM HAS BEEN LOADED!), AND EXECUTION BEGINS.

4.0 STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1
SWITCH REGISTER (000000) IS WORST CASE TEST.

4.2 STARTING ADDRESS

THE PROGRAM MUST ALWAYS BE STARTED AT LOCATION 200(8).

4.3 PROGRAM/OPERATOR ACTION

EITHER:

(1) AT THE CONSOLE: "HALT", ENTER (000200), "LOAD.ADDRESS", "CNTRL/START"

(2) '.R name' TO XXDP MONITOR

(3) 'LOAD name', 'START 200' TO UPDATE PROGRAM (1 OR 2)

5.0 OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

THE DEFINITION OF THE SPECIFIC BITS IN THE SWITCH REGISTER (EITHER HARDWARE OR SOFTWARE) ARE AS FOLLOWS:

SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON CURRENTLY EXECUTING TEST
SW13=1	020000	INHIBIT ERROR TYPEOUTS (WHICH IS AN "ERROR MESSAGE" RESULTING FROM AN ERROR DETECTED IN THE HARDWARE)
SW12=1	010000	INHIBIT STATUS TYPEOUTS (WHICH IS A NON-ERROR RELATED INFORMATIVE MESSAGE, SUCH AS "PASS #XX")
SW11=1	004000	INHIBIT ITERATIONS PER TEST
SW10	002000	SET=BELL ON ERROR/CLEAR=BELL ON PASS END
SW09=1	001000	LOOP ON ERROR
SW08=1	000400	LOOP ON TEST NUMBER IN "$LPTST" IF SET, THEN THE TEST SPECIFIED BY THE TEST NUMBER CONTAINED IN THE MEMORY WORD "$LPTST" (SEE PROGRAM LISTING) WILL SPECIFY THE DESIRED TEST ON WHICH TO LOOP.
SW07	000200	1=16. BIT FP DATA TYPEOUTS 0=SIGN/EXP/FRAC FP DATA TYPEOUTS
SW06	000100	0=DETAILED ERROR PRINTOUTS 1=SUMMARY ONLY ERROR PRINTOUTS
SW05=1	000040	IF ERROR OCCURS AND LOOP-ON-ERROR (SW09) IS SET, FORCE A TIGHT-LOOP-ON-ERROR TO OCCUR.

5.2 PROGRAM/OPERATOR ACTION

ONCE EXECUTION HAS BEGUN, NO OPERATOR INTERVENTION IS REQUIRED, UNLESS IT IS DESIRED TO ALTER A SWITCH REGISTER OPTION, ETC.

IF ALL IS WELL, THE PROGRAM TYPES ITS IDENTIFICATION UPON BEGINNING; AND AT THE START OF EACH PASS, THE CURRENT PASS NUMBER (IN OCTAL) IS ECHOED. NOTE THAT SETTING SW<12>=1 WILL INHIBIT THE TYPEOUT OF THE BEGIN AND END PASS MESSAGES.

IF SW<10>=0, THE CONSOLE BELL WILL BE RUNG AT THE END OF EACH PASS. NOTE THAT ONLY SW<10> AFFECTS THE BELL RINGING AT END OF PASS - SW<12> HAS NO EFFECT ON THIS FUNCTION.

IF AN ERROR OCCURS DURING EXECUTION, MANY VARIATIONS IN ACTION ARE POSSIBLE DEPENDING UPON THE SWITCH SETTINGS:

SW<15>=1 WILL CAUSE THE CPU TO HALT AFTER AN ERROR.
SW<13>=1 WILL ALSO INHIBIT ANY ERROR MESSAGE TYPEOUT THAT WOULD OCCUR AT THIS TIME.
SW<10>=1 WILL CAUSE THE CONSOLE BELL TO BE RUNG ONLY WHEN AN ERROR IS DETECTED (AND NOT AT THE END OF A PASS).
SW<9>=1 CAUSES THE PROGRAM TO LOOP ON THE MOST RECENT ERROR, AS LONG AS IT CONTINUES TO OCCUR.
SW<5>=1 AND SW<9>=1 WILL CAUSE THE DIAGNOSTIC TO ENTER A "TIGHT LOOP ON ERROR" CONDITION. THIS GENERALLY HANGS UP THE PROCESSOR IN A FORCED LOOP ABOUT THE LAST 1-3 FLOATING POINT INSTRUCTIONS EXECUTED. THE DIAGNOSTIC MUST BE HALTED AND RESTARTED AT 200(8) TO BREAK OUT OF THIS LOOP. FLOATING POINT PROCESSOR TRAPS TO 244(8) ARE DISABLED, THE LINE CLOCK IS TURNED "OFF", AND THE CONTENTS OF MEMORY LOCATION "$FPBRK" ARE LOADED INTO THE FP11-E MICROBREAK REGISTER PRIOR TO ENTERING THE LOOP. AT THIS POINT, THE PROCESSOR CAN ALSO BE PUT IN "MAINTENANCE CLOCK" MODE AND SINGLE MICRO CYCLED IN THIS TIGHT LOOP.

THERE ARE ALSO SEVERAL OTHER GENERAL USE FUNCTIONS DEFINED BY THE SWITCHES:

SW<11>=1 WILL INHIBIT THE ITERATIONS (=400(10)) PERFORMED OF EACH TEST ON PASSES 2,3,4, ... THRU THE PROGRAM.
SW<14>=1 CAUSES THE PROGRAM TO LOOP INDEFINATELY ON THE CURRENTLY EXECUTING TEST.
SW<8>=1 CAUSES THE PROGRAM TO CONTINUE EXECUTION AS NORMAL, EXCEPT WHEN THE CONTENTS OF MEMORY WORD "$LPTST" MATCHES THE NUMBER OF THE TEST CURRENTLY EXECUTING. AT THIS POINT, THE TEST IS LOOPED ON INDEFINATELY, UNTIL EITHER SW<8>=0 OR "$LPTST" IS CHANGED. NOTE THAT IF "$LPTST" DOES NOT MATCH THE TEST NUMBER OF ANY TEST, THE CONTENTS OF "$LPTST" ARE EFFECTIVELY IGNORED, AND EXECUTION PROCEEDS NORMALLY.

6.0 ERRORS

6.1 FORMAT OF MESSAGES

6.1.1 ERROR MESSAGE FORMAT

THE FIRST LINE IS A BRIEF MESSAGE THAT EXPLAINS WHICH ERROR WAS DETECTED (EG, AN ERROR IN THE EXPECTED CONTENTS OF A "MULTIPLY ROM" ON THE "MULNET/K10" MODULE).

THE SECOND LINE CONSISTS OF DATA HEADERS TO IDENTIFY THE VALUES TYPED OUT ON LINE THREE. THESE HEADERS WILL EITHER BE OF THE FORM "EXPECTED" AND "RECEIVED" DATA, OR WILL BE A MNEMONIC NAME OF A WORD LOCATION IN MEMORY OR REGISTERS. AT

THE ACTUAL "ERROR CALL" LOCATION IN THE DIAGNOSTIC LISTING (FROM "$ERRPC" LOCATION), EACH HEADER IS DOCUMENTED AS TO WHAT IT SPECIFICALLY CONTAINS.

THE THIRD LINE DISPLAYS THE CONTENTS OF THE LOCATIONS SPECIFIED BY LINE TWO AS SIX DIGIT OCTAL NUMBERS. NOTE THAT ALL DATA DISPLAYED IN ANY MESSAGES ARE OCTAL NUMBERS.

LINES FOUR THRU --- (AS MANY AS NECESSARY) CONTAINING FLOATING POINT FORMAT DATA MAY ALSO BE PRINTED. THEY ARE OF THE FORM:

"REGISTER/LOCATION = [2W/4W FP DATA]"

SW07 (SEE SECTION 5.1) GOVERNS THE FORMAT OF THE DATA TYPEOUTS.

AS EXPLAINED IN SECTION 5.2, SETTING SW<13>=1 WILL SUPPRESS THE TYPING OF THESE MESSAGES.

6.1.2 FLOATING POINT DATA FORMATS

FLOATING POINT STATUS WORD (FPS):

BIT**	OCTAL	FUNCTION
15	100000	FER - FLOATING ERROR FLAG SET WHEN EITHER FIUV, FIU, FIV, FIC ENABLED AND APPROPRIATE EXCEPTION OCCURRED.
14	040000	FID - FLOATING DISABLE INTERRUPTS NO FP INTERRUPTS TO VECTOR 244(8) IF SET.
13:12		NOT USED (ZEROES)
11	004000	FIUV - FLOATING UNDEFINED VARIABLE INTERRUPT IF SET, (-0) MEMORY DATA IS ERROR
10	002000	FIU - FLOATING INTR UNDERFLOW IF SET AND UNDERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY +400(8) IF CLEAR AND UNDERFLOW, ANSWER <-- ZERO
9	001000	FIV - FLOATING OVERFLOW INTERRUPT IF SET AND OVERFLOW, SET FER, STORE ANSWER, EXPONENT WRONG BY -400(8) IF CLEAR AND OVERFLOW, ANSWER <-- ZERO
8	000400	FIC - FLOATING INTEGER CONVERSION INTERRUPT IF SET AND "STCFI" ERROR, ANSWER <-- ZERO, SET ERROR IF CLEAR AND "STCFI" ERROR, ANSWER <-- ZERO
7	000200	FD - FLOATING MODE 1=DOUBLE, 64 BIT OPERANDS (4W) 0=SINGLE, 32 BIT OPERANDS (2W)
6	000100	FL - INTEGER MODE 1=LONG, 32 BIT INTEGERS (2W) 0=SHORT, 16 BIT INTEGERS (1W)
5	000040	FT - ROUND/TRUNCATE MODE 1=TRUNCATE RESULTS 0=ROUND RESULTS
4	000020	FMM - PUT FP11-E ONLY IN MAINTENANCE MODE ALL THIS DOES IS TO ALLOW A HFP MICROBREAK TRAP TO OCCUR.
3:0	000017	FN-FZ-FV-FC - FLOATING CONDITION CODES

FLOATING EXCEPTION CODES (FEC):

OCTAL	ENABLE	
00	(NONE)	(NOT USED)
02	(NONE)	FP OPCODE ERROR
04	(NONE)	FP DIVIDE-BY-ZERO ERROR
06	W/FIC	FP INTEGER CONVERSION ERROR
10	W/FIV	FP OVERFLOW ERROR
12	W/FIU	FP UNDERFLOW ERROR
14	W/FIUV	FP UNDEFINED-VARIABLE/(-0) ERROR
16	W/FMM	FP MAINTENANCE TRAP

```
        FLOATING POINT DATA:

        IN FLOAT MODE (FD=0), IS 2-16.  BIT WORDS, 32.  BITS
        IN DOUBLE MODE (FD=1), IS 4-16.  BIT WORDS, 64.  BITS

        FIRST WORD:  (BOTH F, D MODES)
                B15=SIGN OF NUMBER (1/-, 0/+)
                B14:07=EXPONENT, 8.BITS, FROM -128./+127.
                B06:00=FRACTION, 7 BITS
        SECOND WORD:  (BOTH F, D MODES)
                B15:00=FRACTION, 16.BITS
        THIRD, FOURTH WORDS:  (ONLY D MODE)
                B15:00, B15:00=FRACTION, 32. BITS

        IN F MODE, THE COMPOSITE 24. BIT FRACTION
        IS FORMED BY:

                .1*[WORD1-BIT<06:00>]*[WORD2-BIT<15:00>]

                IN D MODE, THE COMPOSITE 56. BIT FRACTION
                IS FORMED BY:

                .1*[WORD1-BIT<06:00>]*[WORD2-BIT<15:00>]
                        *[WORD3-BIT<15:00>]*[WORD4-BIT<15:00>]

        FOR A MORE DETAILED EXPLANATION OF FLOATING POINT DATA FORMATS
        AND  OPERATIONS,  SEE THE PDP-11/60 PROCESSOR HANDBOOK SECTION
        ON THE FLOATING POINT INSTRUCTION SET.
```

6.2 RECOVERY

ALL ERRORS DETECTED BY THE DIAGNOSTIC WILL BE HANDLED THRU THE "ERROR XXX" TRAP MECHANISM. THUS APPROPRIATE MESSAGES / DATA WILL BE PRINTED ON THE CONSOLE TELETYPE TO INFORM THE USER AS TO THE SOURCE OF THE ERROR CONDITION. BESIDES THE "NORMAL" ERROR CALLS PRESENT AS PART OF EACH INDIVIDUAL TEST, THE FOLLOWING CALLS ARE ALSO PRESENT TO HANDLE MISCELLANEOUS CONDITIONS:

1. TRAP-TO-"4" HANDLER - IF AN "UNEXPECTED CPU ERROR" CONDITION OCCURS, THIS ROUTINE WILL GAIN CONTROL, PRINT AN APPROPRIATE MESSAGE, AND "ATTEMPT" TO RETURN CONTROL WHERE IT LEFT OFF (TRY TO IGNORE ERROR).

2. TRAP-TO-"10" HANDLER - IF SOME TYPE OF "RESERVED INSTRUCTION" TRAP OCCURS, THIS ROUTINE WILL GAIN CONTROL, PRINT AN APPROPRIATE MESSAGE, AND "ATTEMPT" TO RETURN CONTROL WHERE IT LEFT OFF (TRY TO IGNORE ERROR).

3. TRAP-TO-"114" HANDLER - IF EITHER A MEMORY / CACHE / WCS (IF PRESENT) PARITY ERROR OCCURS, THIS ROUTINE WILL GAIN CONTROL, PRINT AN APPROPRIATE MESSAGE, AND "ATTEMPT" TO

RETURN CONTROL WHERE IT LEFT OFF (TRY TO IGNORE ERROR).

4. TRAP-TO-"244" HANDLER - THIS ROUTINE HANDLES "FLOATING POINT EXCEPTION TRAPS". WHETHER UNEXPECTED OR EXPECTED. UNEXPECTED TRAPS GENERATE AN ERROR MESSAGE IMMEDIATELY; EXPECTED TRAPS RETURN TO THE TEST IN PROGRESS, TO COMPLETE THE TEST.

5. TRAP-TO-"OTHER.VECTOR" HANDLER - ANY OTHER TRAP TO AN UNUSED VECTOR IN THE RANGE 000(8)-776(8) IS HANDLED BY THE "SCOPE/ERROR" UNEXPECTED I/O TRAP CODE. AN ERROR MESSAGE IS PRINTED, AND CONTROL RETURNS WHERE INTERRUPTED. THE INTERRUPT IS EFFECTIVELY IGNORED.

6. "PROCESSOR HUNG" RECOVERY - IF THE LINE CLOCK SERVICE ROUTINE "OBSERVES" THAT THE PROCESSOR HAS BEEN EXECUTING A SINGLE INSTRUCTION (POINTED TO BY THE RETURN PC) FOR THE LAST 6 CLOCK TICKS (.1 SECOND), THE "PROCESSOR HUNG: LINE CLOCK TIMEOUT" ERROR IS GENERATED. THIS SHOULD / COULD CONCEIVABLY ONLY HAPPEN IN A FLOATING POINT INSTRUCTION, HUNG IN THE PROCESSOR / FP11-E "FLP.GO / FP.ACK" HANDSHAKE SEQUENCE. CONTROL RETURNS TO THE "HUNG" INSTRUCTION AFTER THE MESSAGE.

6.3 CAUSES

THIS DIAGNOSTIC PROGRAM HAS BEEN ORIENTED TOWARDS THE SPECIFIC HARDWARE ARCHITECTURE OF THE PDP-11/60 PROCESSOR AND THE FP11-E. TO THIS END, HARDWARE INFORMATION IS INCLUDED WITHIN EACH TEST HEADER THAT ATTEMPTS TO DETAIL WHAT LOGIC IS TO BE CONSIDERED "UNDER TEST" DURING EACH TEST. THIS INFORMATION HAS BEEN ORGANIZED ON A MODULE BY MODULE BASIS (IE, K2-K7 PROCESSOR, K8-K11 FP11-E) TO FACILITATE MODULE LEVEL FAULT RESOLUTION. AN EXAMPLE FOLLOWS (NEXT PAGE):

MODULE/ERROR INFO:

FNUA/K8
  MNETSUM-ENABLE-LOGIC, 'MPP'-EXEC, CROM/LATCHES

FEXP/K9
  MNETREG-CLK, MNET-ALU-CONTROL, MIER/MAND-FUNCTION-CONTROL,
  MIER/MAND-CLOCKS, 'MPP'-EXEC, CROM/LATCHES

FMUL/K10
  MIER-REG/MUX-(BYTE4), MAND-REG-(LOW28), MULXX-ROMS,
  CNTR-ROMS, SUM-REG, CARRY-REG, MNET-ALU

FALU/K11
  [PREVIOUSLY VERIFIED]

THE COMMENTS FOLLOWING EACH MODULE DESIGNATOR (IE, FEXP/K9) SPECIFY THE LOGIC "UNDER TEST" ON THAT MODULE (BY THIS TEST). NOT LISTED IS THAT LOGIC THAT HAS ALREADY BEEN VERIFIED / TESTED, AND LOGIC THAT HAS NOT YET BEEN TESTED, BUT WILL BE CHECKED IN SUBSEQUENT TESTS.

TWO OTHER TYPES OF ENTRIES MAY ALSO BE PRESENT:

[ESSENTIALLY NONE] - IMPLIES THAT, AT THIS TIME, THERE IS NO LOGIC ON THIS MODULE THAT IS TO BE CONSIDERED "UNDER TEST".

[PREVIOUSLY VERIFIED] - IMPLIES THAT, AT THIS TIME, ALL LOGIC ON THIS PARTICULAR MODULE THAT IS BEING EMPLOYED HAS BEEN PREVIOUSLY VERIFIED TO BE IN AN OPERATING CONDITION.

7.0 RESTRICTIONS

7.1 STARTING

THE PROGRAM MUST BE STARTED AT LOCATION 200(8) ALWAYS.

7.2 OPERATIONAL

THERE ARE NO OPERATIONAL RESTRICTIONS.

8.0 MISCELLANEOUS

8.1 EXECUTION TIME

MODEL	AVERAGE EXECUTION TIME PER PASS SHORTEST PASS	LONGEST PASS
PDP-11/60 W/FP11-E	0:10	1:45

TIMES SPECIFIED AS (MINUTES):(SECONDS)
SHORTEST PASS ::= PASS=1, NO ITERATIONS, USING:
SWR=(004000) FOR PDP-11/60 W/FP11-E
LONGEST PASS ::= PASS>=2, 400. ITERATIONS/TEST, USING:
SWR=(000000) FOR PDP-11/60 W/FP11-E

8.2 STACK POINTER

THE STACK POINTER IS SET TO 1200(8) AT THE START OF EACH PASS. IF ALL IS OPERATING CORRECTLY, IT SHOULD ALSO BE THIS VALUE AT THE START OF EACH TEST, AND AT THE END OF A PASS.

8.3 POWER FAIL

THE TESTS MAY BE POWER FAILED AT ANY TIME. WHEN POWER IS RESTORED, "POWER" IS TYPED ON THE CONSOLE AND EXECUTION IS RESUMED AT THE ENTRY POINT FOR A NEW PASS, JUST PRIOR TO "TEST 1". THE DIAGNOSTIC CANNOT CONTINUE FROM WHERE IT WAS INTERRUPTED, AS THERE IS NOT SUFFICIENT TIME TO SAVE THE COMPLETE STATE OF THE FP11-E [IE, ALL VOLATILE REGISTERS] DURING A POWER FAIL SEQUENCE.

NOTE THAT THE "VOLATILE" SWITCH REGISTER CONTENTS ARE SAVED AND RESTORED FROM THE STACK IN A POWER FAIL SEQUENCE; THEREFORE THE SWITCH REGISTER SETTINGS SHOULD NOT BE LOST OVER A POWER FAIL.

9.0 PROGRAM DESCRIPTION

9.1 ORGANIZATION

THESE PROGRAMS ARE ORGANIZED AS MUCH AS POSSIBLE IN A STRAIGHTFORWARD, LINEAR MANNER. THE MAIN BODY OF CODE IS STRUCTURED AS FOLLOWS:

(1) INITIALIZATION ROUTINE
- SETS UP VECTORS, TYPES HEADER, ETC.
(2) MAIN BODY OF TESTS
- INLINE TEST CODE, INLINE TEST CALLS
(3) END OF PASS ROUTINE
- END OF PASS PROCESSING
(4) OVERHEAD ROUTINES
- SERVICE SUBROUTINES (TYPEOUT,ETC.)

WHEREVER FEASIBLE, COMMON SECIONS OF CODE FOR WIDELY USED FUNCTIONS ARE CONDENSED INTO SUBROUTINES TO CONSERVE MEMORY. THE "TRAP" INSTRUCTION, AND THE TRAP DECODER ROUTINE, IS THE USUAL METHOD OF INVOKING THESE ROUTINES. THIS INCLUDES NOT ONLY STANDARD SERVICE ROUTINES (SUCH AS SCOPE, ERROR, AND ASCII TYPEOUT), BUT ALSO SOME SPECIALLY DEVELOPED MULTIPLE WORD ARITHMETIC (ADD, SUB, CMP, ASH) AND SERVICE (ERROR LOOP, ETC) ROUTINES.

9.2 TEST DESCRIPTION

9.2.1 TEST SEQUENCE

THIS DIAGNOSTIC HAS BEEN STRUCTURED TO PERFORM ITS TESTS IN THE FOLLOWING SEQUENCE:

1. BASE PROCESSOR SPECIFIC
2. BASE PROCESSOR / FP11-E INTERFACE
3. FP11-E INSTRUCTION DECODE, SEQUENCING / CONTROL
4. FP11-E EXPONENT DATAPATH / CONTROL
5. FP11-E FRACTION DATAPATH / CONTROL
6. FP11-E ROM MULTIPLIER DATAPATH / CONTROL
7. FP11-E EXCEPTION CONDITIONS
8. FP11-E MAINTENANCE INSTRUCTIONS, FUNCTIONAL TESTS

9.2.2 TEST SUMMARY

THE FOLLOWING IS A TEST BY TEST SUMMARY OF THE DIAGNOSTIC. EACH TEST NUMBER AND TITLE IS AS IT APPEARS IN THE ACTUAL DIAGNOSTIC LISTING.

---BASE MACHINE ONLY TESTS---
T1 BM/ WHAMI AND FLAGS INIT
T3 BM/ FLAGS AND INSTR1 FP DECODE
T4 BM/ FP CNST RESTORE
T5 BM/WFP ILLEGAL INTERNAL ADDRESS TEST

```
        ---BASE MACHINE / FP11-E INTERFACE---
          T6  BM/ HFP FLPGO-FPACK; SRVC-GRANT
          T7  BM/ HFP UBREAK SRVC CODE
          T10 BM/ HFP ENABLE/DISABLE, FP INSTR DECODE

        ---FP11-E INSTRUCTION DECODE---
          T11 IFORK(LEFT), -[(ADD+SUB)*MO] FP INSTR DECODE
          T12 FIR8 IMMEDIATE-H ADDRESS MODE DECODE
          T13 UFLOW - FPINIT, F-MODE
          T14 UFLOW - FPINIT, D-MODE

        ---FP11-E EXPONENT DATAPATH---
          T16 EXPNT, ESPAD.B[AC0/3] DATAPATH
          T17 EXPNT, ESPAD.A[AC0/3] DATAPATH
          T20 EXPNT, ESPAD.B[AC4/5] DATAPATH
          T21 EXPNT, "LDEXP/STEXP" FPINMUX(DOUT) DATAPATH
          T22 EXPNT, ESPAD.B ADDRESSING VIA R[DF] AND R[SF]
          T23 EXPNT, ESPAD.A ADDRESSING VIA R[DF] AND R[SF]
          T24 EXPNT, (EA=0, EB=0) W/"CMPF"
          T25 EXPNT, (EA=0.OR.EB=0) W/"MULF"
          T26 EXPNT, (ER=0) W/"ABSF"
          T27 EXPNT, EALU ADD/CARRY LOGIC W/"MULF"
          T30 EXPNT, COUNTER/PRE-SHFT-QUOT WITH "MAS"

        ---FP11-E ADD/SUB-MODE0 DECODE/FLOWS---
          T31 IFORK[(ADD+SUB)*MO], SUMPATH/MO*R(6+7) DECODE
          T32 IFORK[(ADD+SUB)*MO], EXPNT(A+B)=ZERO DECODE
          T33 IFORK[(ADD+SUB)*MO], EXPNT.RANGE.CODE ROM CONTENTS

        ---FP11-E FRACTION DATAPATH---
          T34 FRACTION, FPINMUX-INBUF-FSPADMUX-FSPAD-FPOUTMUX
                DATAPATH, VIA "LDF/STF"
          T35 FRACTION, 60. BIT DATAPATH, VIA "LDD/STD"
          T36 FRACTION, FSPAD DATA PATTERNS, AC0-AC5
          T37 FPINIT/FP.EMIT.(E/F)/FSPAD EXACT.ZERO
          T40 FRACTION, FSPAD ADDRESSING VIA R[DF] AND R[SF]
          T41 FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDF"
          T42 FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCFD"
          T43 FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCDF"
          T44 FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCDF"
          T45 FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCFD"

        ---FP11-E FRACTION, SHIFTER DATAPATH AND CONTROL---
          T46 SHIFTER/NORMK, WITH "MNS"
          T47 SHIFTER, LEFT(2+4) OF (200,0,0,0)
          T50 SHIFTER, RITE(1.-11.) OF (0,0,0,0)
          T51 FRACTION, FALU FSPAD.IN.MUX BIT(59:58)
          T52 FPINIT/FP.EMIT.F/CLRX EXACT.ZERO "01" IN BIT(59:58)
          T53 SHIFTER, RITE(4,5,6,7/1,9)[MAS] RIPPLE-A-1
          T54 SHIFTER, LEFT(2+(1,2,3,4))[MNS] RIPPLE-A-1

        ---FP11-E FRACTION ALU TESTS (ADD)---
          T55 FALU/FEXP, F/D-R/T MODE SELECT
```

T56 FRACTION, FALU ADD/CARRY LOGIC WITH "ADDD"

---FP11-E ADD/SUB MODE-0 INSTR. EXECUTION---
T57 IFORK/(ADD+SUB)*M0 "ADDD"*-M0 EXECUTE

---FP11-E FRACTION, DATAPATH LEFTOVERS---
T60 "DIVF" EXEC, DIVIDE W/INBUF-AR.SHIFT, FSPAD.SELECT
T61 "LDC.I.F" EXEC, FPINMUX/DOUT, SHIFT/NORMALIZE

---FP11-E MULNET DATAPATH AND CONTROL---
T62 MULNET, BASIC DATAPATH
T63 MULNET, MULTIPLY ROM CONTENTS
T64 MULNET, SUM/CARRY REGISTERS, COUNTER ROM CONTENTS
T65 MULNET, MIER REGISTER DATA/SHIFTING
T66 MULNET, MAND REGISTER DATA/SHIFTING

---FP11-E EXPONENT, EXCEPTION CONDITIONS---
T67 EXPNT, ECR AND FCCR EXCEPTION/HFP(CC) CONDITIONS

---FP11-E MAINTENANCE INSTR. TESTS---
T70 'MPP' MAINT. INSTR - FUNCTIONAL TEST
T71 'MNS' MAINT. INSTR - FUNCTIONAL TEST
T72 'MAS' MAINT. INSTR - FUNCTIONAL TEST

9.3 SUBROUTINE ABSTRACTS

9.3.1 TRAPCATCHER

THE TRAPCATCHER IS A SERIES OF INSTRUCTIONS OCCUPYING THE INTERRUPT VECTOR AREA OF MEMORY. IT CONSISTS OF THE SEQUENCE:

```
.WORD   .+2     ;PC AFTER TRAP
.WORD   IOT     ;PS AFTER TRAP
```

PLACED AT EACH VECTOR ADDRESS IN LOCATIONS 4-776(8) OF MEMORY. THE FIRST WORD OF EACH PAIR ("PC AFTER TRAP") POINTS TO THE SECOND WORD, WHICH SERVES A DUAL PURPOSE AS

(1) THE NEW LOADED PS, AND
(2) THE NEXT INSTRUCTION TO EXECUTE (IOT=SCOPE).

WHEN THE PROGRAM IS EXECUTING, ANY REQUIRED VECTORS ARE SET UP IN THE VECTOR AREA WITH APPROPRIATE VALUES; THE OTHERS BEING LEFT IN THE "TRAPCATCHER" STATE. THUS, IF AN UNEXPECTED TRAP EVER OCCURS IN THE MACHINE, IT WILL BE CAUGHT, AND THE MACHINE WILL ENTER THE SCOPE ROUTINE. THE SCOPE ROUTINE WILL THEN DETECT THAT THE RETURN PC IS FROM THE TRAP CATCHER AREA, AND EXIT TO THE ERROR ROUTINE TO PRINT AN APPROPRIATE ERROR MESSAGE, INDICATING AN UNEXPECTED TRAP OCCURRED.

9.3.2 SCOPE ROUTINE - $SCOPE

THE SCOPE ROUTINE IS ENTERED FROM THE FIRST INSTRUCTION OF EACH TEST IN THE PROGRAM. (NOTE THAT BY DEFINITION, A "TEST" WILL BE DESIGNATED AS THE SECTION OF CODE BETWEEN TWO "SCOPE" STATEMENTS.) THIS ROUTINE PROVIDES THE OVERHEAD CODE NECESSARY TO IMPLEMENT SEVERAL OF THE SWITCH REGISTER CONTROL OPTIONS. UPON ENTRANCE TO A TEST, THE SCOPE STATEMENT AT THE BEGINNING SETS UP CERTAIN LOCATIONS (SEE BELOW) TO SPECIFY THE CURRENT TEST NUMBER AND LOOPING ADDRESS (FOR ITERATIONS). CONTROL IS THEN PASSED TO THE ACTUAL TEST CODE, PERFORMING THE DESIRED TEST. UPON EXIT, THE SCOPE STATEMENT OF THE NEXT TEST IS ENTERED, WHICH DETERMINES WHETHER TO (1) LOOP BACK TO THE PREVIOUS TEST (EG, FOR ITERATIONS) OR (2) INITIALIZE FOR THE NEXT TEST (AS DESCRIBED EARLIER, ABOVE).

ENTRANCE TO THE SCOPE ROUTINE IS VIA AN "IOT" TRAP CALL THROUGH LOCATION 20(8). (FROM THE SCOPE=IOT EQUATE). DEPENDING UPON THE SWITCH SETTINGS (SEE 5.2), CODE IS PRESENT TO: LOAD THE FP11 MICRO BREAK REGISTER, LOOP ON THE CURRENTLY EXECUTING TEST, LOOP ON A SPECIFIC TEST, PERFORM ITERATIONS OF EACH TEST, AND SET UP ADDRESSES FOR POSSIBLE LOOPING ON ERRORS. IMPORTANT VALUES USED IN THIS ROUTINE ARE:

- $MXCNT - MAXIMUM NUMBER OF ITERATIONS PER TEST (GENERALLY WILL BE 400(10))
- $TSTNM - A COUNTER INDICATING THE NUMBER (1-377(8)) OF THE TEST CURRENTLY BEING EXECUTED
- $LPADR - CONTAINS THE ADDRESS TO WHICH THE SCOPE ROUTINE WILL LOOP, IF THE CURRENT TEST IS BEING LOOPED UPON
- $LPERR - CONTAINS THE ADDRESS TO WHICH THE ERROR ROUTINE (SEE 9.3.3) WILL LOOP, IF AN ERROR OCCURS AND THE LOOPING ON AN ERROR OPTION IS SPECIFIED IN THE SWITCHES. SET UP BY SCOPE. GENERALLY WILL BE THE SAME AS $LPADR, ABOVE.

9.3.3 ERROR ROUTINE - $ERROR

THE ERROR ROUTINE IS ENTERED WHEN THE TEST CODE HAS DETERMINED THAT AN ERROR HAS OCCURRED AS PART OF A TEST. THROUGH USE OF THIS ROUTINE, THE TEST HAS A MEANS OF SIGNALING AN ERROR TO THE OPERATOR/MONITOR; AND IMPLEMENTING THE CONTROL FUNCTIONS FOR HALTING ON ERROR, BELL ON ERROR, AND LOOPING ON ERROR. IN ADDITION, THE ERROR ROUTINE HAS THE PROVISION TO TYPE OUT ON THE OPERATOR'S CONSOLE A MESSAGE BRIEFLY EXPLAINING THE ERROR, AND SOME OF THE MOST PERTINENT DATA VALUES TO HELP DIAGNOSE THE CAUSE (SEE SECTION 6.2).

THE CALLING MECHANISM IS SIMILAR TO THAT EMPLOYED FOR THE SCOPE ROUTINE (VIA A TRAP), EXCEPT IN THIS INSTANCE, THE "EMT" INSTRUCTION IS USED, TRAPPING THROUGH LOCATION 30(8). (NOTE THE EQUATE ERROR N=EMT N). THE LOWER BYTE OF THE EMT

INSTRUCTION IS CAPABLE OF TRANSMITTING A NUMBER FROM 0-377(8), WHICH WILL BE TERMED THE "ERROR ITEM NUMBER." THIS NUMBER DETERMINES WHICH ERROR MESSAGE, AND ASSOCIATED DATA VALUES WILL BE TYPED OUT WHEN A PARTICULAR ERROR IS SIGNALED. IF THIS NUMBER IS ZERO, JUST THE PC OF THE CALLING "ERROR" INSTRUCTION WILL BE TYPED, OTHERWISE, THE NUMBER IS USED AS AN INDEX THROUGH THE ERROR TABLE ($ERRTB) TO FIND THE APPROPRIATE VALUES TO TYPE (SEE PROGRAM LISTING FOR FURTHER DETAILS).

IMPORTANT VALUES USED IN THIS ROUTINE ARE:

- EREG0 THRU EREG7 - CONTENTS OF GENERAL REGISTERS R0 THRU R7 JUST BEFORE ERROR CALL
- $ERTTL - CUMULATIVE NUMBER OF ERRORS ENCOUNTERED TO DATE
- $ERRPC - CONTAINS THE PC OF THE "ERROR" INSTRUCTION JUST EXECUTED
- $LPERR - CONTAINS THE ADDRESS WHICH WILL BE LOOPED UPON FOR THE ERROR LOOPING FACILITY

9.3.4 ERROR MESSAGE TYPEOUT ROUTINE - $TYPERR

THIS ROUTINE ($TYPERR ENTRY POINT) IS CALLED BY THE ERROR PROCESSING ROUTINE DESCRIBED IN 9.3.3 ABOVE. ITS PURPOSE IS TO IMPLEMENT THE ERROR MESSAGE/DATA VALUE ERROR TYPEOUT FACILITY. THE SUBROUTINE WILL, GIVEN THE INDEXING BYTE FROM THE ERROR CALL INSTRUCTION, PICK UP THE CORRECT ERROR MESSAGE VECTOR FROM $ERRTB (ERROR TABLE), AND TYPE OUT THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES ON THE CONSOLE.

9.3.5 TYPE ROUTINE - $TYPE

THIS ROUTINE IS THE STANDARD SYSTEM TYPEOUT ROUTINE FOR ASCII SINGLE-CHARACTER-PER-BYTE STRINGS. IT IS CALLED THROUGH A TRAP INSTRUCTION WITH THE NEXT WORD CONTAINING THE ADDRESS OF THE FIRST CHARACTER IN THE STRING. TYPING TERMINATES WHEN AN ALL-ZERO BYTE IS FOUND. HORIZONTAL TAB STOPS ARE ALSO AUTOMATICALLY PLACED.

9.3.6 OCTAL NUMBER TYPE ROUTINE - $TYPOC

THIS ROUTINE CONVERTS THE TOP NUMBER ON THE STACK TO A 6-DIGIT OCTAL REPRESENTATION, AND TYPES IT ON THE CONSOLE USING THE TYPE ROUTINE $TYPE. SEE LISTING FOR OPTIONS AND FURTHER DETAILS.

9.3.7 POWER UP AND DOWN ROUTINES - $PWRUP AND $PWRDN

THESE TWO ROUTINES ARE ENTERED FOR THE POWER UP AND DOWN CONDITIONS, RESPECTIVELY. THE POWER DOWN ROUTINE ($PWRDN) SAVES THE GENERAL REGISTERS AND STACK POINTER. THE POWER UP ROUTINE ($PWRUP) CORRESPONDINGLY RESTORES THE REGISTERS, STACK POINTER, AND TYPES THE MESSAGE "POWER" WHEN POWER IS RESTORED.

THE VOLATILE INTERNAL SWITCH REGISTER IS ALSO SAVED/RESTORED BY THIS ROUTINE. THE DIAGNOSTIC RESTARTS AT THE ENTRY POINT FOR A NEW PASS AFTER A POWER FAIL / RESTART SEQUENCE.

9.3.8 END OF PASS ROUTINE - $EOP

THE END OF PASS ROUTINE COUNTS THE NUMBER OF PASSES PERFORMED, DINGS THE BELL/TYPES A MESSAGE (IF ENABLED), AND ALSO INTERFACES TO THE MONITOR, IF PRESENT.

9.3.9 USER ROUTINES

THIS SECTION GIVES A SHORT DESCRIPTION OF THE FUNCTION OF EACH OF THE USER DEFINED TRAP-CALL SUBROUTINES. SEE THE DIAGNOSTIC LISTING FOR A COMPLETE DESCRIPTION OF EACH SUBROUTINE'S FUNCTION, OPERAND FORMAT, ETC.

--ARITHMETIC ROUTINES--
ASH64M -4W/64.BIT ARITHMETIC LEFT/RIGHT SHIFT
ASH64I -4W/64.BIT ARITHMETIC LEFT/RIGHT SHIFT
SUB64M -4W/64.BIT SUBTRACT
CMP64M -4W/64.BIT COMPARE EQ/NE
CMP32M -2W/32.BIT COMPARE EQ/NE

--PROCESSOR TRAP CATCHERS--
SETDW/CLRDW -ENABLE/DISABLE LINE CLOCK ESCAPE TRAP
SETUB/CLRUB -ENABLE/DISABLE PROCESSOR MICRO BREAK ESCAPE TRAP
SETFP/CLRFP -ENABLE/DISABLE FLOATING POINT ESCAPE TRAP

--MISC. SERVICE--
ERRPNT -SETUP "$LPERR" ERROR LOOP ADDRESS
LOOPNT -SETUP "$LPADR" SCOPE LOOP ADDRESS
CND$ES -CONDITIONALLY SETUP "$ESCAPE" ERROR LOOP ADDRESS

--MISC. FP SERVICE--
SGLDAT -GENERATE 2W/32.BITS RANDOM DATA
DBLDAT -GENERATE 4W/64.BITS RANDOM DATA
ZAPHFP -INIT FP11-E/WFP-STATUS, ENABLE HFP EXECUTE
ZAPWFP -INIT FP11-E/WFP-STATUS, ENABLE WFP EXECUTE
EADJ -GENERATE FP11-E "EADJ" EXPNT/"NORMK" VALUE
FIXFRA -USING "EADJ", GENERATE FRACTION "HIDDEN BITS" AND INSERT

10.0 ACT/APT/XXDP

10.1 ACT COMPATIBILITY

THIS PROGRAM SHOULD RUN UNDER THE ACT SYSTEM MONITOR.

10.2 APT COMPATIBILITY

THIS PROGRAM WILL RUN UNDER THE APT SYSTEM MONITOR. ALL NECESSARY SOFTWARE COMMUNICATION HOOKS ARE PRESENT.

10.2.1 LOADING ONTO APT

THIS DIAGNOSTIC SHOULD BE LOADED ONTO THE APT SYSTEM IN THE STANDARD MANNER, USING THE "TSP - TEST SOFTWARE PACKAGE" UTILITY.

THE "LONGEST TEST" AND "FIRST PASS" RUN TIMES SPECIFIED AS DEFAULTS IN THE DIAGNOSTIC LISTING SHOULD BE KEPT AS IS. THEY ARE, FOR REFERENCE:

LONGEST TEST = 15. SECONDS
FIRST PASS = 10. SECONDS

NOTES ON LOADING:

SETTING "ENVIRONMENT [$ENV]"=(001) WILL FORCE THE DIAGNOSTIC TO USE THE "APT SWITCH REGISTER" [SWITCH 1] IN PLACE OF THE HARDWARE SWITCH REGISTER. THIS ACTION IS INDEPENDENT OF THE "ENVIRONMENTAL MODE [$ENVM]" INDICATOR CONTENTS.

ANY "MEANINGFUL" COMBINATIONS OF SWITCH REGISTER OPTIONS IN "SWITCH 1" IS VALID. "SWITCH 2" [$USWR] IS NOT USED BY THIS PROGRAM.

10.2.2 RUNNING UNDER APT

NO SPECIAL PRECAUTIONS ARE NECESSARY TO RUN UNDER APT.

10.3 XXDP COMPATIBILITY

FOR XXDP MEDIA COMPATIBILITY, THE TOP 2K WORDS OF THE 16K WORD MINIMUM MEMORY AREA ARE NOT DISTURBED DURING EXECUTION. THIS LEAVES (1) THE "XXDP" MONITOR INTACT FOR CHAIN MODE EXECUTION OF DIAGNOSTICS, AND (2) SPACE FOR "UPD1/2" TO PERFORM DIAGNOSTIC UPDATES.

```
    14          OPERATIONAL SWITCH SETTINGS
    32          BASIC DEFINITIONS
   256          TRAP CATCHER
   281          STARTING ADDRES(ES)
   284          ACT11 HOOKS
   295          APT PARAMETER BLOCK
   318          COMMON TAGS
   384          APT MAILBOX-ETABLE
   411          ERROR POINTER TABLE
   592          PROGRAM DEFINED COMMON TAGS
   729          START OF PASS ROUTINE
   740          INITIALIZE THE COMMON TAGS
   840  T1      BM/ WHAMI AND FLAGS INIT
   932  T2      ...ENABLE BM MICROBREAK TRAP-TO-4, IN WHAMI...
   949  T3      BM/ FLAGS AND INSTR1 FP DECODE
  1084  T4      BM/ FP CNST RESTORE
  1222  T5      BM/WFP ILLEGAL INTERNAL ADDRESS TEST
  1327  T6      HFP/BM:  FLPGO-FPACK; SRVC-GRANT
  1472  T7      BM/ HFP UBREAK SRVC CODE
  1578  T10     HFP ENABLE/DISABLE, FP INSTR DECODE
  1714  T11     IFORK(LEFT), -[(ADD+SUB)*MD] FP INSTR DECODE
  1986  T12     FIRB IMMEDIATE-H ADDRESS MODE DECODE
  2096  T13     UFLOW - FPINIT, F-MODE
  2207  T14     UFLOW - FPINIT, D-MODE
  2313  T15     ...ENABLE HFP MICROBREAK LOAD...
  2328  T16     EXPNT, ESPAD.B[AC0/3] DATAPATH
  2547  T17     EXPNT, ESPAD.A[AC0/3] DATAPATH
  2788  T20     EXPNT, ESPAD.B[AC4/5] DATAPATH
  2958  T21     EXPNT, "LDEXP/STEXP" FPINMUX(DOUT) DATAPATH
  3053  T22     EXPNT, ESPAD.B ADDRESSING VIA R[DF] AND R[SF]
  3206  T23     EXPNT, ESPAD.A ADDRESSING VIA R[DF] AND R[SF]
  3361  T24     EXPNT, (EA=0, EB=0) W/"CMPF"
  3487  T25     EXPNT, (EA=0.OR.EB=0) W/"MULF"
  3600  T26     EXPNT, (ER=0) W/"ABSF"
  3719  T27     EXPNT, EALU ADD/CARRY LOGIC W/"MULF"
  3866  T30     EXPNT, COUNTER/PRE-SHFT-QUOT WITH "MAS"
  3951  T31     IFORK[(ADD+SUB)*MD], SUMPATH/MD*R(6+7) DECODE
  4125  T32     IFORK[(ADD+SUB)*MD], EXPNT(A+B)=ZERO DECODE
  4212  T33     IFORK[(ADD+SUB)*MD], EXPNT.RANGE.CODE ROM CONTENTS
  4417  T34     FRACTION, FPINMUX-INBUF-FSPADMUX-FSPAD-FPOUTMUX DATAPATH, VIA "LDF/STF"
  4511  T35     FRACTION, 60. BIT DATAPATH, VIA "LDD/STD"
  4632  T36     FRACTION, FSPAD DATA PATTERNS, AC0-AC5
  4965  T37     FPINIT/FP.EMIT.(E&F)/FSPAD EXACT.ZERO
  5028  T40     FRACTION, FSPAD ADDRESSING VIA R[DF] AND R[SF]
  5173  T41     FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDF"
  5245  T42     FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCFD"
  5317  T43     FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCDF"
  5387  T44     FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCDF"
  5459  T45     FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCFD"
  5529  T46     SHIFTER/NORMK, WITH "MNS"
  5635  T47     SHIFTER, LEFT(2+4) OF (200,0,0,0)
  5695  T50     SHIFTER, RITE(1.-11.) OF (0,0,0,0)
  5770  T51     FRACTION, FALU FSPAD.IN.MUX BIT(59:58)
  5841  T52     FPINIT/FP.EMIT.F/CLRX EXACT.ZERO "01" IN BIT(59:58)
  5910  T53     SHIFTER, RITE(4,5,6,7/1,9)[MAS] RIPPLE-A-1
  6041  T54     SHIFTER, LEFT(2+(1,2,3,4))[MNS] RIPPLE-A-1
```

6158 T55 FALU/FEXP, F/D-R/T MODE SELECT
6273 T56 FRACTION, FALU ADD/CARRY LOGIC WITH "ADDD"
6664 T57 IFORK/(ADD+SUB)*MD "ADDD"*-MD EXECUTE
6966 T60 "DIVF" EXEC, DIVIDE W/INBUF-AR.SHIFT, FSPAD.SELECT
7105 T61 "LDC.I.F" EXEC, FPINMUX/DOUT, SHIFT/NORMALIZE
7205 T62 MULNET, BASIC DATAPATH
7488 T63 MULNET, MULTIPLY ROM CONTENTS
7721 T64 MULNET, SUM/CARRY REGISTERS, COUNTER ROM CONTENTS
8228 T65 MULNET, MIER REGISTER DATA/SHIFTING
8408 T66 MULNET, MAND REGISTER DATA/SHIFTING
8530 T67 EXPNT, ECR & FCCR EXCEPTION/HFP(CC) CONDITIONS
8687 T70 'MPP' MAINT. INSTR - FUNCTIONAL TEST
8821 T71 'MNS' MAINT. INSTR - FUNCTIONAL TEST
8966 T72 'MAS' MAINT. INSTR - FUNCTIONAL TEST
9114 T73 ...EXIT TO EOP...
9132 END OF PASS ROUTINE
9166 INTERRUPT SERVICE ROUTINES
9170 ...DW11-L LINE CLOCK INTERRUPT SERVICE ROUTINE
9213 ...FPP INTERRUPT SERVICE ROUTINE
9282 ...TRAP-TO-4 INTERRUPT SERVICE ROUTINE
9327 ...TRAP-TO-10 INTERRUPT SERVICE ROUTINE
9345 ...MEMORY/CACHE PARITY ERROR INTERRUPT SERVICE ROUTINE
9361 MISC. SUPPORT SUBROUTINES
9365 ...RANDOM "FPS" SUBROUTINE (RANFPS)
9402 ...RANDOM FLOATING POINT DATA SUBROUTINES (DBLDAT, SGLDAT)
9433 RANDOM NUMBER GENERATOR ROUTINE
9473 ...INITIALIZE HFP/WFP, FPS/FEC/FEA (ZAPHFP, ZAPWFP)
9519 ...NORMALIZATION COUNT GENERATION SUBROUTINE (EADJ)
9556 ...FRACTION ADJUSTMENT ROUTINE (FIXFRA)
9590 64. BIT ARITHMETIC/LOGICAL FUNCTION SUBROUTINES
9594 ...64. BIT "ASHC" ROUTINE (ASH64I, ASH64M)
9700 ...64. BIT "SUB" ROUTINE (SUB64M)
9741 ...MULTIPLE WORD COMPARISON ROUTINES (CMP64M, CMP32M, CMPXXM)
9789 --SYSMAC SUPPORT ROUTINES--
9793 SCOPE HANDLER ROUTINE
9892 ERROR HANDLER ROUTINE
10005 ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)
10092 FLOATING POINT DATA TYPEOUT ROUTINE
10149 TYPE ROUTINE
10233 APT COMMUNICATIONS ROUTINE
10295 BINARY TO OCTAL (ASCII) AND TYPE
10375 "TRAP" INSTRUCTION DECODER
10397 TRAP TABLE
10436 ...SETUP LINE-CLOCK/PROCESSOR HUNG ESCAPE (SETDW, CLRDW)
10463 ...SETUP PROCESSOR MICROBREAK ESCAPE (SETUB, CLRUB)
10497 ...SETUP FLOATING POINT TRAP ESCAPE (SETFP, CLRFP)
10522 ...CONDITIONALLY LOAD $ESCAPE (CND$ES)
10543 ...SETUP ERROR LOOP ($LPERR) POINT (ERRPNT)
10554 ...SETUP SCOPE LOOP ($LPADR) POINT (LOOPNT)
10567 POWER DOWN AND UP ROUTINES
10616 ERR MESSAGES, DATA HEADERS, DATA VECTORS, ETC

```
                        .TITLE  PDP-11/60 FP11-E HARDWARE DIAGNOSTIC
                        ;*COPYRIGHT (C) 1977
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DON NORTH
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USIN  HE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                        ;*

                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH  OCTAL           USE
                        ;*      ------  ------  ------------------
                        ;*        15    100000  HALT ON ERROR
                        ;*        14    040000  LOOP ON CURRENTLY EXECUTING TEST
                        ;*        13    020000  INHIBIT ERROR TYPEOUTS
                        ;*        12    010000  INHIBIT STATUS TYPEOUTS
                        ;*        11    004000  INHIBIT ITERATIONS
                        ;*        10    002000  1=BELL ON ERROR
                        ;*         9    001000  LOOP ON ERROR
                        ;*         8    000400  LOOP ON TEST NUMBER IN "$LPTST"
                        ;*         7    000200  0=SIGN/EXP/FRAC FP DATA TYPEOUTS
                        ;*                      1=16. BIT WORD   "    "      "
                        ;*         6    000100  0=DETAILED PRINTOUT OF ERRORS
                        ;*                      1=SUMMARY ONLY
                        ;*         5    000040  TIGHT LOOP ON ERROR

                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
001200                  STACK=  1200
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL

                        ;*MISCELLANEOUS DEFINITIONS
000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
000012                  LF=     12              ;;CODE FOR LINE FEED
000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                        .EQUIV  PS,PSW
177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER

                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000                  R0=     %0              ;;GENERAL REGISTER
000001                  R1=     %1              ;;GENERAL REGISTER
000002                  R2=     %2              ;;GENERAL REGISTER
000003                  R3=     %3              ;;GENERAL REGISTER
000004                  R4=     %4              ;;GENERAL REGISTER
000005                  R5=     %5              ;;GENERAL REGISTER
```

```
                                R6=     %6              ;;GENERAL REGISTER
                                R7=     %7              ;;GENERAL REGISTER
                                SP=     %6              ;;STACK POINTER
                                PC=     %7              ;;PROGRAM COUNTER

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                          SW15=   100000
040000                          SW14=   40000
020000                          SW13=   20000
010000                          SW12=   10000
004000                          SW11=   4000
002000                          SW10=   2000
001000                          SW09=   1000
000400                          SW08=   400
000200                          SW07=   200
000100                          SW06=   100
000040                          SW05=   40
000020                          SW04=   20
000010                          SW03=   10
000004                          SW02=   4
000002                          SW01=   2
000001                          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)
100000                          BIT15=  100000
040000                          BIT14=  40000
020000                          BIT13=  20000
010000                          BIT12=  10000
004000                          BIT11=  4000
002000                          BIT10=  2000
001000                          BIT09=  1000
000400                          BIT08=  400
000200                          BIT07=  200
000100                          BIT06=  100
000040                          BIT05=  40
000020                          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
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR

                        ;*MED CODES
                        ;*
                        ;*      USE IS AS FOLLOWS:
                        ;*
                        ;*      READING:                        WRITING:
                        ;*
                        ;*      ...                             MOV     #DATA,R0
                        ;*      MED     ,RXXX                   MED     ,WXXX
                        ;*      MOV     R0,RESULT               ...
                        ;*

076600                  MED=    076600          ;OPCODE

000352                  WINIT=  352             ;BM "INIT" SUBR, R0=FLAGS FOR FUNCTION

000346                  WSR=    346             ;BM "SR" (SHIFT REGISTER, WRITE ONLY)

000350                  WNUA=   350             ;BM "NUA" (NEXT MICROADDRESS, WRITE ONLY BIT<14:03>)

000022                  RWHAMI= 022             ;BM "WHAMI"
000222                  WWHAMI= 222             ;

                                                ;THE FOLLOWING ARE READ-ONLY IN THE CSP:
000100                  LOGJAM= 100             ;CSP00: LOG JAM
000101                  LOGSVC= 101             ;CSP01: LOG SERVICE
000102                  LOGPBA= 102             ;CSP02: LOG PHYS BUS ADDR
000103                  LOGCUA= 103             ;CSP03: LOG CURRENT MICROADDRESS
```

```
                        LOGFLG= 104             ;CSP04: LOG FLAG/INTR
                        LOGWHM= 105             ;CSP05: LOG WHAMI
                        LOGCAD= 106             ;CSP06: LOG CACHE DATA
                        LOGCAT= 107             ;CSP07: LOG CACHE TAG

                        RFPA=   066             ;FP "FPA"
                        WFPA=   266             ;

                        RFLAG=  144             ;BM "FLAGS<8:4,2:0>#FPS<7:0>"
                        WFLAG=  344             ;BM "FLAGS<8:4,2:0>" AND "EXFLAG<2:1>"

                        RFEA=   076             ;FP "FEA"
                        WFEA=   276             ;

                        RFEC=   036             ;FP "FPSHI#FEC"
                        WFEC=   236             ;

                        RSERVC= 141             ;BM SERVICE PORT OF STATUS MUX

                        RCSP00= 100             ;BM CSP(00):  FP CONSTANTS, BM ERROR LOG
                        WCSP00= 300             ;

                        ;#FLOATING POINT INTERRUPT VECTOR
                        FPPVEC= 244

                        ;#FLOATING POINT REGISTER DEFINITIONS
                        AC0=    %0
                        AC1=    %1
                        AC2=    %2
                        AC3=    %3
                        AC4=    %4
                        AC5=    %5
                        AC6=    %6
                        AC7=    %7

                        ;#PDP-11/60 PROCESSOR-SPECIFIC UNIBUS ADDRESSES
                        CPUERR= 177766          ;CPU ERROR REGISTER
                        MEMERR= 177744          ;MEMORY ERROR REGISTER
                        CPUBRK= 177770          ;CPU MICROBREAK ADDRESS REGISTER
                        CPUCCR= 177746          ;CPU CACHE CONTROL REGISTER

                        ;#LINE CLOCK (DW11-L) REGISTERS, ETC
                        DW11LC= 177546          ;CSR
                        DW11LV= 100             ;INTERRUPT VECTOR

                        ;#FP11-E - PDP-11/60 SPECIFIC MAINTENANCE INSTRUCTIONS
                        MPP=    170005          ;"MAINTENANCE PARTIAL PRODUCT"
                        MAS=    170007          ;"MAINTENANCE ALIGNMENT SHIFT"
                        MNS=    170004          ;"MAINTENANCE NORMALIZATION SHIFT"

                        ;#BIT PATTERNS FOR TESTS
                        ALTP=   052525          ;0101...01
```

```
                052525                  AP=     ALTP            ;
                125252                  ALTN=   125252          ;1010...1C
                125252                  AN=     ALTN            ;
                007417                  ALT4P=  007417          ;0000111100001111
                170360                  ALT4N=  170360          ;1111000011110000
                177776                  M2=     177776          ;1111...10  MINUS TWO
                177777                  M1=     177777          ;1111...11  MINUS ONE, ALL 1'S
                100000                  M0=     100000          ;1000...00  MINUS ZERO
                077777                  LGP=    077777          ;0111...11  LGST + NUM (1ST WD FLT )
                177777                  LGN=    177777          ;1111...11  LGST - NUM (1ST WD FLT )
                000200                  SMP=    000200          ;+1*2**-128, SMLT + NUM (1ST WD FLT)
                100200                  SMN=    100200          ;-1*2**-128, SMLT - NUM (1ST WD FLT)
                000177                  ZX1MP=  000177          ;ZERO EXP, ALL 1-S MANT (1ST WD FLT)
                100177                  ZX1MN=  100177          ;ZERO EXP, ALL 1-S MANT (1ST WD FLT)
                040200                  F1P=    040200          ;+1.0E+0, 1ST WD FLT
                140200                  F1N=    140200          ;-1.0E+0, 1ST WD FLT
                104210                  P13Z=   104210          ;1000100010001000
                000377                  LB=     000377          ;0000000011111111  LOWER BYTE
                177400                  UB=     177400          ;1111111100000000  UPER BYTE


                                        ;*FPS BIT PATTERNS
                147777                  FPS1=   147777          ;ALL BITS ON (READABLE)
                000000                  FPS0=   000000          ;ALL BITS OFF
                000000                  NA=     000000          ;FOR FEC, WHEN NOT APPLICABLE

                                        ;*PSW BIT PATTERNS
                177760                  CCONLY= 177760          ;FOR BIC TO GET CC BITS ONLY


                                        .SBTTL  TRAP CATCHER

                000000                          .=0
                                        ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
                                        ;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
                                        ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
                                        ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
                                        ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
                                        ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
                                        ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
                                        ;*      PC=YYYYYY UNEXPECTED TRAP TO XXX
                                        ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
                                        ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
                                        ;*      YYYYYY=PC AT TIME OF ~ AP
                                        ;*NOTE: IF THE PROCESSOR IS NO. AN 11/05 THE PROGRAM
                                        ;*      CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.

000000  000000                $4OCAT: HALT                    ;;HALT
000002  000737                        BR      .-100           ;;BRANCH TO 177700 & TIME OUT (NOT ON
                                                                ;;11/05)
000004  003400                        .WORD   START           ;;VECTOR TO STARTING ADDRESS
000006  000340                        .WORD   340             ;;WITH PRIORITY LEVEL 7
                000174                  .=174
000174  000000                DISPREG:.WORD   0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                SWREG:  .WORD   0               ;;SOFTWARE SWITCH REGISTER
                                        .SBTTL  STARTING ADDRES(ES)
```

```
                                        JMP     @#START ;;GO TO START OF PROGRAM

                                .SBTTL  ACT11 HOOKS

                                ;;******************************************************************
                                ;HOOKS REQUIRED BY ACT11
000204                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  030464                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                  .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
000204                                  .=$SVPC                 ;; RESTORE PC
001000                                  .=1000
                                .SBTTL  APT PARAMETER BLOCK

                                ;;******************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;******************************************************************
001000                                  .$X=.   ;;SAVE CURRENT LOCATION
000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                          200     ;;FOR APT START UP
000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044  001000                          $APTHDR ;;POINT TO APT HEADER BLOCK
001000                                  .=.$X   ;;RESET LOCATION COUNTER
                                ;;******************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001000                          $APTHD:
001000  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001002  001356                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001004  000017                  $TSTM:  .WORD   15.     ;;RUN TIM OF LONGEST TEST
001006  000012                  $PASTM: .WORD   10.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001010  000000                  $UNITM: .WORD   0.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001012  000014                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS

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

                                        .=STACK+10
001210                  001210  $CMTAG:                          ;;START OF COMMON TAGS
                                ;;-----START $CMTAG CLEAR-----
001210  000000                          .WORD   0
001212  000000                  $TSTNM: .WORD   0                ;;CONTAINS THE TEST NUMBER
001214  000000                  $ERFLG: .WORD   0                ;;CONTAINS ERROR FLAG
001216  000000                  $ICNT:  .WORD   0                ;;CONTAINS SUBTEST ITERATION COUNT
001220  000000                  $LPADR: .WORD   0                ;;CONTAINS SCOPE LOOP ADDRESS
001222  000000                  $LPERR: .WORD   0                ;;CONTAINS SCOPE RETURN FOR ERRORS
001224  000000                  $ERTTL: .WORD   0                ;;CONTAINS TOTAL ERRORS DETECTED
001226  000000                  $ITEMB: .WORD   0                ;;CONTAINS ITEM CONTROL BYTE
001230  000001                  $ERMAX: .WORD   1                ;;CONTAINS MAX. ERRORS PER TEST
001232  000000                  $ERRPC: .WORD   0                ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001234  000000                  $GDADR: .WORD   0                ;;CONTAINS ADDRESS OF 'GOOD' DATA
001236  000000                  $BDADR: .WORD   0                ;;CONTAINS ADDRESS OF 'BAD' DATA
001240  000000                  $GDDAT: .WORD   0                ;;CONTAINS 'GOOD' DATA
001242  000000                  $BDDAT: .WORD   0                ;;CONTAINS 'BAD' DATA
001244  000000                          .WORD   0                ;;RESERVED--NOT TO BE USED
001246  000000                          .WORD   0
001250     000                  $AUTOB: .BYTE   0                ;;AUTOMATIC MODE INDICATOR
001251     000                  $INTAG: .BYTE   0                ;;INTERRUPT MODE INDICATOR
001252  000000                          .WORD   0
                                ;;-----END $CMTAG CLEAR-----
001254  177570                  $SWR:   .WORD   DSWR             ;;ADDRESS OF SWITCH REGISTER
001256  177570                  DISPLAY: .WORD  DDISP            ;;ADDRESS OF DISPLAY REGISTER
001260  000000                  $LPTST: .WORD   0                ;;CONTAINS TEST NUMBER TO LOOP UPON
001262  000000                  $FPBRK: .WORD   0                ;;CONTAINS HFP UBRK ADDR LOADED DURING 'SCOPE'
001264  177560                  $TKS:   177560                   ;;TTY KBD STATUS
001266  177562                  $TKB:   177562                   ;;TTY KBD BUFFER
001270  177564                  $TPS:   177564                   ;;TTY PRINTER STATUS REG. ADDRESS
001272  177566                  $TPB:   177566                   ;;TTY PRINTER BUFFER REG. ADDRESS
001274     000                  $NULL:  .BYTE   0                ;;CONTAINS NULL CHARACTER FOR FILLS
001275     002                  $FILLS: .BYTE   2                ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001276     012                  $FILLC: .BYTE   12               ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001277     000                  $TPFLG: .BYTE   0                ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001300  000000                  $REGAD: .WORD   0                ;;CONTAINS THE ADDRESS FROM
                                                                 ;;WHICH  ($REG0) WAS OBTAINED
001302  000000                  $REG0:  .WORD   0                ;;CONTAINS (($REGAD)+0)
001304  000000                  $REG1:  .WORD   0                ;;CONTAINS (($REGAD)+2)
001306  000000                  $REG2:  .WORD   0                ;;CONTAINS (($REGAD)+4)
001310  000000                  $REG3:  .WORD   0                ;;CONTAINS (($REGAD)+6)
001312  000000                  $REG4:  .WORD   0                ;;CONTAINS (($REGAD)+10)
001314  000000                  $REG5:  .WORD   0                ;;CONTAINS (($REGAD)+12)
001316  000000                  $REG6:  .WORD   0                ;;CONTAINS (($REGAD)+14)
001320  000000                  $REG7:  .WORD   0                ;;CONTAINS (($REGAD)+16)
001322  000000                  $REG10: .WORD   0                ;;CONTAINS (($REGAD)+20)
001324  000000                  $REG11: .WORD   0                ;;CONTAINS (($REGAD)+22)
001326  000000                  $REG12: .WORD   0                ;;CONTAINS (($REGAD)+24)
001330  000000                  $REG13: .WORD   0                ;;CONTAINS (($REGAD)+26)
001332  000000                  $REG14: .WORD   0                ;;CONTAINS (($REGAD)+30)
```

```
373  001334  000000                  $REG15: .WORD   0               ;;CONTAINS (($REGAD)+32)
374  001336  000000                  $REG16: .WORD   0               ;;CONTAINS (($REGAD)+34)
375  001340  000000                  $REG17: .WORD   0               ;;CONTAINS (($REGAD)+36)
376  001342  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
377  001344  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
378  001346  177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
379  001352     077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
380  001353     015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
381  001354  000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
382                                  ;;**************************************************************
383                                  .SBTTL  APT MAILBOX-ETABLE
384
385                                  ;;**************************************************************
386                                  .EVEN
387  001356                          $MAIL:                          ;;APT MAILBOX
388  001356  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
389  001360  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
390  001362  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
391  001364  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
392  001366  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
393  001370  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
394  001372  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
395  001374  000000                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
396  001376                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
397  001376     000                  $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
398  001377     000                  $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
399  001400  000000                  $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
400  001402  000000                  $USWR:  .WORD   AUSWR           ;;USER SWITCHES
401  001404  000000                  $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
402                                  ;*                              BITS 15-11=CPU TYPE
403                                  ;*                                      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
404                                  ;*                                      11/70=06,PDQ=07,Q=10
405                                  ;*                              BIT 10=REAL TIME CLOCK
406                                  ;*                              BIT  5=FLOATING POINT PROCESSOR
407                                  ;*                              BIT  8=MEMORY MANAGEMENT
408  001406                          $ETEND:
409                                  .MEXIT
```

```
                                        .SBTTL  ERROR POINTER TABLE

001406                                  $ERRTB:

                                        ;*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:

                                        ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                        ;*      DH              ;;POINTS TO THE DATA HEADER
                                        ;*      DT              ;;POINTS TO THE DATA
                                        ;*      DV              ;;POINTS TO THE DATA VECTOR

001406  035656  042052  043460  EMV001: .WORD   EMA,DHA,DTA,000        ;UNEXPECTED FPP TRAP, FPP INSTR EXEC OK
001414  000000
001416  035656  042052  043460  EMV002: .WORD   EMB,DHB,DTB,000        ;UNEXPECTED FPP TRAP, FPP INSTR EXEC NOT OK
001424  000000
001426  035710  042124  043476  EMV003: .WORD   EMC,DHC,DTC,000        ;TRAP TO (4)
001434  000000
001436  035761  042124  043476  EMV004: .WORD   EME,DHC,DTC,000        ;TRAP TO (114) [MEMORY/CACHE PARITY ERROR]
001444  000000
001446  035734  042124  043476  EMV005: .WORD   EMD,DHC,DTC,000        ;TRAP TO (10)
001454  000000
001456  036462  000000  000000  EMV006: .WORD   EMK1,000,000,DVAK      ;FSPAD UNIQUE ADDRESSING ERROR, R[DF]
001464  044140
001466  036511  000000  000000  EMV007: .WORD   EMK2,000,000,DVAK      ;FSPAD UNIQUE ADDRESSING ERROR, R[SF]
001474  044140
001476  036007  000000  000000  EMV010: .WORD   EME1,000,000,DV1       ;MAINT INSTR, ALTERED ACO
001504  043752
001506  036037  042765  043604  EMV011: .WORD   EME2,DHZ,DTZ,000       ;MAINT INSTR, ALTERED FPS
001514  000000
001516  036067  042660  043600  EMV012: .WORD   EME3,DHX,DTX,DV2       ;MPP, BAD MULNET(S+C)
001524  043764
001526  036116  042660  043600  EMV013: .WORD   EME4,DHX,DTX,DV3       ;MNS, BAD NORM.
001534  044002
001536  036140  042660  043600  EMV014: .WORD   EME5,DHX,DTX,DV3       ;MAS, BAD SHIFT
001544  044002
001546  036166  042660  043600  EMV015: .WORD   EME6,DHX,DTX,DV2       ;MAS, BAD CNTR
001554  043764
001556  036215  042167  043512  EMV016: .WORD   EMF,DHF,DTF,000        ;MULNET MULTIPLY ROM ERROR, SPECIFIC DATA
001564  000000
001566  036252  042223  043524  EMV017: .WORD   EMG,DHG,DTG,000        ;MULNET SUM/CARRY ROM ERROR
001574  000000
001576  036307  042257  043536  EMV020: .WORD   EMH,DHH,DTH,000        ;MULNET MULTIPLY ROM ERROR, GENERAL DATA
001604  000000
001606  036333  042322  043552  EMV021: .WORD   EMI,DHI,DTI,000        ;HFP IFORK/-[(ADD+SUB)*MO] DECODE ERROR
001614  000000
001616  036407  042322  043552  EMV022: .WORD   EMJ,DHI,DTI,000        ;HFP IFORK/-[(ADD+SUB)*MO] UNEXPECTED ERROR
001624  000000
001626  036540  042374  043512  EMV023: .WORD   EML,DHL,DTL,000        ;HFP PREPO/1/2, UBRK, */+ ENABLE ERROR
001634  000000
001636  036611  042430  043634  EMV024: .WORD   EMM,DHM,DTM,000        ;PROCESSOR HUNG:  LINE CLOCK TIMEOUT
001644  000000
001646  036647  042464  043512  EMV025: .WORD   EMN,DHN,DTN,000        ;BAD BM WHAMI/FLAG AT INIT
```

```
   466  001654  000000
   467  001656  036677  042522  043562  EMV026: .WORD   EM0,DH0,DT0,000        ;BM FLAGS READ/WRITE ERROR
   468  001664  000000
   469  001666  036720  042541  043572  EMV027: .WORD   EMP,DHP,DTP,000        ;BM FLAGS OK / FP DECODE ERROR
   470  001674  000000
   471  001676  036762  042560  043512  EMV030: .WORD   EMQ,DHQ,DTQ,000        ;BM BM BAD DECODE / FLAGS
   472  001704  000000
   473  001706  037023  042550  043564  EMV031: .WORD   EMR,DHR,DTR,000        ;BM CSP INVALID FLAG NOT CLEARED AFTER RESTORE
   474  001714  000000
   475  001716  037070  042616  043570  EMV032: .WORD   EMS,DHS,DTS,000        ;BM BAD FP CSP CONSTANT RESTORED
   476  001724  000000
   477  001726  037211  042142  043502  EMV033: .WORD   EMAC,DHAC,DTAC,0000    ;BM/ HFP HUNG PROC DURING STATUS INSTR
   478  001734  000000
   479  001736  037116  043022  043616  EMV034: .WORD   EMAA,DHAA,DTAA,000     ;HFP/ FPSRVC &PR7 ERROR
   480  001744  000000
   481  001746  037141  043040  043564  EMV035: .WORD   EMAB,DHAB,DTAB,000     ;BAD UBRK CODE FROM HFP (#7)
   482  001754  000000
   483  001756  037254  000000  000000  EMV036: .WORD   EMAD,0000,0000,DVAD    ;MULNET - DATA STUCK/H OR REGISTER LOAD ERROR
   484  001764  044020
   485  001766  037330  000000  000000  EMV037: .WORD   EMAE,0000,0000,DVAE    ;MULNET - RIPPLE ALT 0/1-S ERROR
   486  001774  044032
   487  001776  037356  000000  000000  EMV040: .WORD   EMAF,0000,0000,DVAE    ;MULNET - RIPPLE ALT 0/1-S ERROR - (S)*(S+C)
   488  002004  044032
   489  002006  037413  000000  000000  EMV041: .WORD   EMAG,0000,0000,DVAG    ;MULD - RIPPLE-A-1 THRU MIER ERROR
   490  002014  044066
   491  002016  037460  000000  000000  EMV042: .WORD   EMAH,0000,0000,DVAG    ;MULD - RIPPLE-A-1 THRU MAND ERROR
   492  002024  044066
   493  002026  037525  000000  000000  EMV043: .WORD   EMAI,0000,0000,0000    ;HFPP STATUS INSTR EXEC MODIFIED FPS
   494  002034  000000
   495  002036  041504  043167  043710  EMV044: .WORD   EMCA,DHCA,DTCA,0000    ;FPINIT FLOW, UNEXP'D FPSHI#FEC ERR
   496  002044  000000
   497  002046  041547  043167  043710  EMV045: .WORD   EMCB,DHCA,DTCA,0000    ;FPINIT FLOW, HFP DIDN'T UBREAK ERR
   498  002054  000000
   499  002056  041612  043372  043716  EMV046: .WORD   EMCC,DHCC,DTCC,0000    ;BM/WFP ILLEGL.INTRNL.ADDR ERR
   500  002064  000000
   501  002066  041650  000000  000000  EMV047: .WORD   EMCD,0000,0000,DVCD    ;ESPAD.B ADDRS ERR; R[DF]
   502  002074  044344
   503  002076  041701  000000  000000  EMV050: .WORD   EMCE,0000,0000,DVCE    ;ESPAD.B ADDRS ERR; R[SF]
   504  002104  044344
   505  002106  041732  000000  000000  EMV051: .WORD   EMCF,0000,0000,DVCF    ;ESPAD.A ADDRS ERR; R[DF]
   506  002114  044344
   507  002116  041763  000000  000000  EMV052: .WORD   EMCG,0000,0000,DVCG    ;ESPAD.A ADDRS ERR; R[SF]
   508  002124  044344
   509  002126  040661  043167  043710  EMV053: .WORD   EMBK,DHBD,DTBD,DVBD    ;EXPNT EALU EA+EB MULF/SEQ ERR
   510  002134  044256
   511  002136  040036  000000  000000  EMV054: .WORD   EMAP,0000,0000,DVAF    ;LDD/STD FRAC DATAPATH ERR
   512  002144  044054
   513  002146  040004  000000  000000  EMV055: .WORD   EMAO,0000,0000,DVAF    ;LDF/STF FRAC DATAPATH ERR
   514  002154  044054
   515  002156  040070  043070  043564  EMV056: .WORD   EMAQ,DHAQ,DTAQ,DVAJ    ;FSPAD DATA ERR
   516  002164  044134
   517  002166  040107  000000  000000  EMV057: .WORD   EMAR,0000,0000,DVAL    ;FSPAD FP-INSTR MODIFIED SRC-ACC ERR
   518  002174  044146
   519  002176  040153  000000  000000  EMV060: .WORD   EMAS,0000,0000,DVAL    ;FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR
   520  002204  044146
   521  002206  037570  000000  000000  EMV061: .WORD   EMAJ,0000,0000,DVAH    ;NORMK-EADJ/SHFTR ERR
```

```
522  002214  044110
523  002216  037615  000000  000000  EMV062: .WORD  EMAK,0000,0000,DVAI     ;SHFTR L(2+4) OF ZERO ERR
524  002224  044126
525  002226  037646  043054  043624  EMV063: .WORD  EMAL,DHAL,DTAL,DVAI     ;SHFTR R(11-1) OF ZERO ERR
526  002234  044126
527  002236  037701  043054  043630  EMV064: .WORD  EMAM,DHAM,DTAM,DVAH     ;SHFTR, MAS-RITE RIPPLE-A-1 ERR
528  002244  044110
529  002246  037740  043054  043630  EMV065: .WORD  EMAN,DHAN,DTAN,DVAH     ;SHFTR, MNS-LEFT/RITE RIPPLE-A-1 ERR
530  002254  044110
531  002256  040220  043077  043646  EMV066: .WORD  EMAT,DHAT,DTAT,0000     ;ESPALU.B LDX/STX DATAPATH ERR
532  002264  000000
533  002266  04025_  043077  043646  EMV067: .WORD  EMAU,DHAU,DTAU,0000     ;ESPAD.A LDX/STX DATAPATH ERR
534  002274  000000
535  002276  040312  000000  000000  EMV070: .WORD  EMAV,0000,0000,DVAV     ;FALU ADDD/CARRY RESULT ERR
536  002304  044160
537  002306  040345  000000  000000  EMV071: .WORD  EMAW,0000,0000,DVAH     ;FSPAD.IN.MUX INBUF-PORT ERR
538  002314  044110
539  002316  040401  043124  043656  EMV072: .WORD  EMAX,DHAX,DTAX,0000     ;FIRB IMMED-H MODE DECODE ERR
540  002324  000000
541  002326  040436  043156  043624  EMV073: .WORD  EMAY,DHAY,DTAY,DVAM     ;IFORK/(ADD+SUB)*MD EXECUTE ERR
542  002334  044214
543  002336  040475  043167  043710  EMV074: .WORD  EMBA,DHBA,DTBA,DVBA     ;EXPNT EA=0, EB=0 DATAPATH ERR
544  002344  044236
545  002346  040533  043167  043710  EMV075: .WORD  EMBB,DHBB,DTBB,DVBB     ;EXPNT EA=0+EB=0 DATAPATH ERR
546  002354  044236
547  002356  040570  043167  043710  EMV076: .WORD  EMBC,DHBC,DTBC,DVBC     ;EXPNT ER=0 DATAPATH ERR
548  002364  044250
549  002366  040620  043167  043710  EMV077: .WORD  EMBD,DHBD,DTBD,DVBD     ;EXPNT EALU EA-PLUS-EB RESULT ERR
550  002374  044256
551  002376  040724  043212  043700  EMV100: .WORD  EMBE,DHBE,DTBE,0000     ;EXPNT CNTR/PRE-SHFT-QUOT-ROM ERR
552  002404  000000
553  002406  040765  043167  043710  EMV101: .WORD  EMBF,DHBF,DTBF,DVBF     ;IFORK/(ADD+SUB)*MD SUMPATH/MD*R6 ERR
554  002414  044300
555  002416  041032  043167  043710  EMV102: .WORD  EMBG,DHBG,DTBG,0000     ;IFORK/(ADD+SUB)*MD [EA+EB]=0/MD*R6 ERR
556  002424  000000
557  002426  041101  043241  043666  EMV103: .WORD  EMBH,DHBH,DTBH,DVBH     ;IFORK/(ADD+SUB)*MD EXPNT.RANGE.CODE.ROM ERR
558  002434  044202
559  002436  041155  043156  043624  EMV104: .WORD  EMBI,DHBI,DTBI,DVBI     ;DIVIDE INBUF/AR-SHIFT FSPAD-SELECT ERR
560  002444  044214
561  002446  041224  043303  043624  EMV105: .WORD  EMBJ,DHBJ,DTBJ,DVBJ     ;LDCP(I->F) FPINMUX/DOUT CONVERT ERR
562  002454  044054
563  002456  041270  043313  043624  EMV106: .WORD  EMBL,DHBL,DTBL,DVBL     ;FEXP/FALU FPS F-D &/+ R-T MODE ERR
564  002464  044312
565  002466  041333  000000  000000  EMV107: .WORD  EMBM,0000,0000,DVBM     ;FRACTION/CLRD-EXEC/FP.EMIT.F DATA ERR
566  002474  044324
567  002476  041401  000000  000000  EMV110: .WORD  EMBN,0000,0000,DVBN     ;FPINIT/CLRD/LDEXP BIT<59:58> INSERT ERR
568  002504  044332
569  002506  041451  043322  043730  EMV111: .WORD  EMBO,DHBO,DTBO,DVBO     ;ECR/FCCR EXCEPTION+FCC ERR
570  002514  044160
571  002516  042014  043442  043744  EMV112: .WORD  EMCH,DHCH,DTCH,0000     ;LDEXP/STEXP FPINMUX(DOUT) ERR
572  002524  000000
573  002526  000000  000000  000000  EMV113: .WORD  0000,0000,0000,0000     ;(NU)
574  002534  000000
575  002536  000000  000000  000000  EMV114: .WORD  0000,0000,0000,0000     ;(NU)
576  002544  000000
577  002546  000000  000000  000000  EMV115: .WORD  0000,0000,0000,0000     ;(NU)
```

```
                                                EMV116: .WORD   0000,0000,0000,0000     ;(NU)
                                                EMV117: .WORD   0000,0000,0000,0000     ;(NU)
                                                EMV120: .WORD   0000,0000,0000,0000     ;(NU)


                                        .SBTTL  PROGRAM DEFINED COMMON TAGS
                                        ;*VARIABLES
                                                .EVEN
                                        ;***** NOT CLEARED EVER *****
                                                .WORD   0               ;TBS

                                        ;********** START CLEARING OF PROGRAMMER DEFINED COMMON TAGS *********
                                        STPDCT:
                                        $FPS:   .WORD   0               ;PROGRAM-CONTROLLED FPS
                                        FPS:    .WORD   0               ;FPS STORED HERE AFTER STFPS
                                        FEC:    .WORD   0               ;FEC STORED HERE AFTER STST
                                        FEA:    .WORD   0               ;FEA STORED HERE AFTER STST

                                        FPESCP: .WORD   0               ;0=NU / NOT-0=ESCAPE ADDR AFTER FP TRAP
                                        ;KEEP THE NEXT TWO LINES IN THIS ORDER
                                        FPTPOK: .BYTE   0               ;377=FP INTERRUPT/TRAP IS OK / 000=FP INTERRUPT/TRAP IS
                                                                                                        AN ERROR
                                        NOFPIE: .BYTE   0               ;377=OK TO EXECUTE FP INSTRUCTIONS / 000=DON'T EXECUTE F
                                                                                                          P INSTRUCTIONS
                                        FPLENF: .BYTE   0               ;[B7=1]=4W MODE / [B7=0]=2W MODE

                                        UBTPOK: .BYTE   0               ;377=UBRK(BM) ENABLED, JUST RTI / 000=UBRK(BM) IS AN ERR
                                                                                                        OR
                                        UBESCP: .WORD   0               ,BM UBRK ESCAPE ADDR
                                        UBCNTR: .WORD   0               ;BM JBRK COUNTER

                                        DWLOPC: .WORD   0               ;DW LAST OLD PC
                                        DWESCP: .WORD   0               ;DW ESCAPE ADDRESS
                                        DWCNTR: .WORD   0               ;DW # TIMES A MATCH COUNTER
                                        DWLOOP: .WORD   0               ;DW LOOP COUNT, FOR CHECKING HUNG AT INSTRUCTION OR IN L
                                                                                                        OOP
                                        DWOLOP: .WORD   0               ;OLD VERSION OF ABOVE
                                        DWFLAG: .BYTE   0               ;DW ESCAPE ENABLE: 377=YES/000=FORCE-AN-ERROR

                                        LPTITE: .BYTE   0               ;377=FORCE TIGHT LOOP, ERROR OR NOT / 000=DISABLED

                                        ;*MEMORY FLOATING POINT ACCUMULATORS
                                        ;  (NO RELATION TO HFP/WFP FP ACCUMULATORS)
                                        MFAC0:  .BLKW   4               ;
                                        MFAC1:  .BLKW   4               ;
                                        MFAC2:  .BLKW   4               ;
                                        MFAC3:  .BLKW   4               ;
                                        MFAC4:  .BLKW   4               ;
```

```
                                        MFAC5:  .BLKW   4               ;
                                        MFAC6:  .BLKW   4               ;
                                        MFAC7:  .BLKW   4               ;

                                        ;*REGISTER CONTENTS, AT ERROR, FOR DISPLAY
                                        EREG0:  .WORD   0
                                        EREG1:  .WORD   0
                                        EREG2:  .WORD   0
                                        EREG3:  .WORD   0
                                        EREG4:  .WORD   0
                                        EREG5:  .WORD   0
                                        EREG6:  .WORD   0
                                        EREG7:  .WORD   0

                                        ;*AFTER A TRAP CONDITION, OLD PC/PS/SP SAVED HERE
                                        OLDPC:  .WORD   0
                                        OLDPS:  .WORD   0
                                        OLDSP:  .WORD   0
                                        FPINST: .WORD   0

                                        ENPDCT: .WORD   0
                                        ;********** END CLEARING OF PROGRAMMER DEFINED COMMON TAGS **********


                                        ;*SOME COMMONLY USED CONSTANTS, STORED IN MEMORY
        000006                          DW$CNT=6.       ;LSE # MATCHES
003000  000006                          DWICNT: .WORD   DW$CNT          ;REQUIRE THIS NUMBER OF MATCHES TO SIGNAL PROC HUNG
                                                                        ; [IE, TICKS OF THE LINE CLOCK]


                                        ;*SOME FP CONSTANTS:
003002  100125                          FPMOAP: .WORD   100125          ;100125,052525,052525,052525
003004  052525  052525                  FPALTP: .WORD   AP,AP           ;052525,052525,052525,052525
003010  052525                          FPAPOO: .WORD   AP              ;052525,052525,000000,000000
003012  052525  000000  000000          FPAPIM: .WORD   AP,0,0,0        ;052525,000000,000000,000000
003020  000000
003022  100052                          FPMOAN: .WORD   100052          ;100052,125252,125252,125252
003024  125252  125252                  FPALTN: .WORD   AN,AN           ;125252,125252,125252,125252
003030  125252                          FPANOO: .WORD   AN              ;125252,125252,000000,000000
003032  125252  000000                  FPANIM: .WORD   AN,0            ;125252,000000,000000,000000
003036  000000  000000  177777  FPOOI1: .WORD   0,0,M1,M1        ;000000,000000,177777,177777
003044  177777
003046  077777                          FPPONE: .WORD   77777           ;077777,177777,177777,177777
003050  177777  177777                  FPONES: .WORD   M1,M1           ;177777,177777,177777,177777
003054  177777  177777                  FP1100: .WORD   M1,M1           ;177777,177777,000000,000000
003060  000000  000000  000000  FPZERO: .WORD   0,0,0,0          ;000000,000000,000000,000000
003066  000000
003070  000125  052525                  FPZEAP: .WORD   125,AP          ;000125,052525,052525,052525
003074  052525  052525  125252  FPAPAN: .WORD   AP,AP,AN,AN      ;052525,052525,125252,125252
003102  125252
003104  000052  125252  125252  FPZEAN: .WORD   52,AN,AN,AN      ;000052,125252,125252,125252
003112  125252
003114  000177  000000  000000  FPZOIM: .WORD   177,0,0,0        ;000177,000000,000000,000000
003122  000000
003124  100177  000000  000000  FPMOIM: .WORD   100177,0,0,0     ;100177,000000,000000,000000
003132  000000
```

```
003134  152525  052525  052525  FPAPMS: .WORD   152525,AP,AP,AP ;152525,052525,052525,052525
003142  052525
003144  025252  125252  125252  FPANMS: .WORD   25252,AN,AN,AN  ;025252,125252,125252,125252
003152  125252
003154  077000  000000  000000  FPEM4Z: .WORD   77000,0,0,0     ;077000,000000,000000,000000
003162  000000
003164  177600  000000  000000  FPSEFZ: .WORD   177600,0,0,0    ;177600,000000,000000,000000
003172  000000


                                ;*VECTOR INITIALIZATION TABLE
                                ;*      WORD 1 = VECTOR ADDRESS
                                ;*      WORD 2 = PC
                                ;*      WORD 3 = PS
                                ;*
003174                          VECTAB:
003174  000244  030616  000340          .WORD   FPPVEC,FPPILT,PR7       ;FPP VECTOR
003202  000004  031000  000340          .WORD   ERRVEC,TRP004,PR7       ;TRAPS TO 4  (TIMEOUT, UBRK, ETC)
003210  000010  031104  000340          .WORD   RESVEC,TRP010,PR7       ;TRAPS TO 10 (ILLEGAL INSTRUCTIONS, ETC)
003216  000114  031126  000340          .WORD      114,TRP114,PR7       ;MEMORY/CACHE PARITY ERRORS
003224  000100  030500  000300          .WORD   DW11LV,DW11LI,PR6       ;LINE CLOCK
003232  000000                          .WORD   000000                  ;END


                                ;*SOME ASCII MESSAGES
003234  000011                  $HT:    .ASCIZ  <HT>            ;HORIZ TAB
003236  000056                  $DT:    .ASCIZ  "."             ;PERIOD
003240  000057                  $SL:    .ASCIZ  "/"             ;SLANT
003242  005015  005012  042115  BGNMES: .ASCII  <CR><LF><LF><LF>"MD-11-DQFPE-"
003250  030455  026461  050504
003256  050106  026505
003262  030101                          .ASCII  "AO"
003264  027056                          .ASCII  ".."
003266  042120  026520  030461          .ASCIZ  "PDP-11/60 FP11-E HARDWARE DIAGNOSTIC"<CR><LF>
003274  033057  020060  050106
003302  030461  042455  044040
003310  051101  053504  051101
003316  020105  044504  043501
003324  047516  052123  041511
003332  005015     000
003335     015  050012  051501  NWPAS1: .ASCIZ  <CR><LF>"PASS #"
003342  020123  000043
```

```
                                .SBTTL  START OF PASS ROUTINE

                                ;;**********************************************************************
                                ;;**********************************************************************
                                ;;**********************************************************************
                                .ENABL  AMA     ;ASSEMBLE ALL RELATIVE REFERENCES AS ABSOLUTE
                                ;;**********************************************************************
                                ;;**********************************************************************
                                ;;**********************************************************************

003400                          START:
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
003400  012706  001210                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
003404  005026                  CLR     (R6)+           ;;CLEAR MEMORY LOCATION
003406  022706  001254          CMP     #SWR,R6 ;;DONE?
003412  001374                  BNE     .-6             ;;LOOP BACK IF NO
003414  012706  001200          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
003420  012737  032320  000020  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
003426  012737  000340  000022  MOV     #PR7,@#IOTVEC+2 ;;LEVEL 7
003434  012737  032724  000030  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
003442  012737  000340  000032  MOV     #PR7,@#EMTVEC+2 ;;LEVEL 7
003450  012737  035020  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
003456  012737  000240  000036  MOV     #PR5,@#TRAPVEC+2 ;;LEVEL 5 (ALLOW LINE CLOCK)
003464  012737  035466  000024  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
003472  012737  000340  000026  MOV     #PR7,@#PWRVEC+2 ;;LEVEL 7
003500  013737  030450  030442  MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
003506  012737  176543  031374  MOV     #176543,$HINUM  ;;PRIME THE RANDOM NUMBER GENERATOR
003514  012737  123456  031376  MOV     #123456,$LONUM  ;;BOTH HIGH AND LOW WORDS
003522  005037  001342          CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
003526  005037  001344          CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
003532  012737  000001  001230  MOV     #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
003540  012737  003540  001220  MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
003546  012737  003546  001222  MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
003554  013746  000004          MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
003560  012737  003614  000004  MOV     #64$,@#ERRVEC   ;;SET UP EPROR VECTOR
003566  012737  177570  001254  MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
003574  012737  177570  001256  MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
003602  022777  177777  175444  CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
003610  001012                  BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
003612  000403                  BR      65$             ;;BRANCH IF NO TIMEOUT
003614  012716  003622  64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
003620  000002                  RTI
003622  012737  000176  001254  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
003630  012737  000174  001256          MOV     #DISPREG,DISPLAY
003636  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

003642  005037  001364          CLR     $PASS           ;;CLEAR PASS COUNT
003646  122737  000001  001376  CMPB    #APTENV,$ENV    ;;TEST USER UNDER APT
003654  001003                  BNE     67$             ;;NO, USE NON-APT SWITCH
003656  012737  001400  001254  MOV     #$SWREG,SWR     ;;YES, USE APT SWITCH REGISTER
003664                  67$:
```

```
                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       POWER FAIL/RESTART ENTERS HERE
                                        ;////////////////////////////////////////////////////////////////////////

003664  012700  001212          RESTRT: MOV     #$TSTNM,R0              ;CLEAR $CMTAG AREA
003670  005020                  1$:     CLR     (R0)+                   ;
003672  020027  001254                  CMP     R0,#SWR                 ;
003676  002774                          BLT     1$                      ;

                                        ;*SETUP VECTOR AREA
003700  012700  003174                  MOV     #VECTAB,R0              ;ADDR(TABLE)
003704  012001                  VECINT: MOV     (R0)+,R1                ;R1=VECTOR, IF ZERO, DONE
003706  001403                          BEQ     VECDON                  ;BR IF DONE WITH SETUP
003710  012021                          MOV     (R0)+,(R1)+             ;SETUP PC
003712  012011                          MOV     (R0)+,(R1)              ;SETUP PS
003714  000773                          BR      VECINT                  ;GO FOR NEXT
003716                          VECDON:

                                        ;*ID MESSAGE AT STARTUP
003716  104401  003242                  TYPE    ,BGNMES

                                        ;////////////////////////////////////////////////////////////////////////
                                        ;       NEW PASS ENTERS HERE
                                        ;////////////////////////////////////////////////////////////////////////

                                        ;*RESET STACK POINTER, FOR INSURANCE
003722  012706  001200          NEWPAS: MOV     #STACK,SP               ;RESET TO KNOWN VALUE

                                        ;*CLEAR PROGRAMMER DEFINED COMMON TAGS AREA
003726  012700  002610                  MOV     #STPDCT,R0              ;FIRST LOCATION
003732  005020                  BGNPCT: CLR     (R0)+                   ;CLEAR IT
003734  020027  002776                  CMP     R0,#ENPDCT              ;UP TO LAST ?
003740  101774                          BLOS    BGNPCT                  ;NO, CONTINUE

                                        ;*START OUT AT PROCESSOR PRIO=0, KERNEL MODE, T-BIT=0
003742  005046                          CLR     -(SP)                   ;PS=(000000)
003744  012746  003752                  MOV     #.+6,-(SP)              ;PC OF RETURN
003750  000006                          RTT                             ;AND NOW POP (000000)->PS

                                        ;*START LINE CLOCK, IF ITS NOT GOING
003752  012737  000006  003000          MOV     #DW$CNT,DWICNT          ;RESET MASTER TICK COUNT
003760  013737  003000  002636          MOV     DWICNT,DWCNTR           ;RESET TICK COUNTER
003766  012737  000100  177546          MOV     #BIT6,DW11LC            ;SET INTR ENABLE, CLEAR READY

                                        ;*NEXT PASS MESSAGE
003774  032777  010000  175252          BIT     #BIT12,@SWR             ;INHIBIT STATUS TYPEOUTS ?
004002  001011                          BNE     TST1                    ;BR IF YES

004004  104401  003335                  TYPE    ,NWPAS1                 ;"PASS #"
004010  013746  001364                  MOV     $PASS,-(SP)             ;PASS COUNT INTO ...
004014  005216                          INC     (SP)                    ;  1-N RANGE
004016  104403                          TYPOS                           ;TYPE OCTAL
004020     006     000                  .BYTE   6,0                     ;  6 DIGITS, NO LEADING ZEROS
004022  104401  001353                  TYPE    ,$CRLF                  ;END THE LINE
```

```
                        ;*****************************************************************
                        ;*TEST 1        BM/ WHAMI AND FLAGS INIT
                        ;
                        ;       THE FOLLOWING TEST USES THE BASE MACHINE "INITIALIZE" ROUTINE
                        ;       (ACCESSIBLE VIA A MED CODE) TO INITIALIZE THE "WHAMI" AND
                        ;       "FLAG" REGISTERS.  THEY ARE THEN READ, USING MED FUNCTIONS,
                        ;       AND COMPARED TO THE FOLLOWING EXPECTED VALUES:
                        ;
                        ;         "WHAMI"<15:00>=(000021)="0000 0000 0001 0001"
                        ;               BIT<04>="1" -> HFP PRESENT
                        ;               BIT<00>="1" -> ERROR LOG ENABLED
                        ;               BIT<08,06,05> ARE IGNORED (DCS/ECS/WCS PRESENT BITS)
                        ;               ALL OTHER BITS SHOULD BE ZEROES
                        ;
                        ;         "FLAGS"<08:00>=(014000)="0001 1000"
                        ;               FLAG<5:4>="11" -> HFP ENABLED / CSP CNST INVALID
                        ;               ALL OTHER FLAGS ARE ZEROED.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       R0      -RECEIVED BM "FLAGS" AFTER INIT, IN HOB
                        ;       R1      -EXPECTED BM "FLAGS" AFTER INIT
                        ;       R2      -EXPECTED BM "WHAMI" AFTER INIT
                        ;       R3      -RECEIVED BM "WHAMI" AFTER INIT
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         [ESSENTIALLY NONE]
                        ;
                        ;       FEXP/K9
                        ;         HFP-PRESENT-LOGIC
                        ;
                        ;       FMUL/K10
                        ;         [ESSENTIALLY NONE]
                        ;
                        ;       FALU/K11
                        ;         [ESSENTIALLY NONE]
                        ;
                        ;       UWORD/K2
                        ;         UCON-FP-LOGIC, UCON-FLAG-LOGIC, WHAMI-REG, INIT MICROCODE
                        ;
                        ;*****************************************************************
004026  000004          TST1:   SCOPE
                        ;
                        ;       ;INIT ROUTINE:  JAM/TRACK/BASCON/GR/PS/MMR0/SLR/FLAGS/WHAMI/HFP
004030  012700  177400          MOV     #177400,R0              ;STICK JUNK (!) IN WHAMI BEFORE
004034  076600  000222          MED     ,WWHAMI                 ;  TO SEE IF REWRITTEN
004040  012700  015537          MOV     #015537,R0              ;CONSTANT THAT GOES IN SR
004044  076600  000352  63$:     MED     ,WINIT                  ;EXECUTE THE BM INIT SUBROUTINE
004050  105737  002645          TSTB    @PTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
004054  001373                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                                ;
004056  012701  014000          MOV     #014000,R1              ;EXPECTED FLAGS AFTER THE INIT
```

```
                        MOV     #000021,R2              ;EXPECTED WHAMI AFTER THE INIT

                        MED     ,RWHAMI                 ;GET 'INIT-TED' WHAMI
                        BIC     #BIT8+BIT6+BIT5,R0      ;ZERO DCS/ECS/WCS PRESENT BITS
                        MOV     R0,R3                   ;SAVE RECEIVED WHAMI IN R3

                        MED     ,RFLAG                  ;GET 'INIT-TED' FLAGS#FPS
                        CLRB    R0                      ;ZAP FPS PORTION, IN LOB

                        ;*COMPARE WHAMI (EXPD):(RCVD)
                        CMP     R2,R3                   ;
                        BEQ     10$                     ;BR IF AGREE
                        ERROR   25                      ;BM WHAMI / BAD INIT
                        ;"BM WHAMI/FLAGS INIT ERROR"
                        ;       R-FLAGS = RECEIVED BM FLAGS<8:0> IN R0<15:08>
                        ;       E-FLAGS = EXPECTED BM FLAGS<8:0> IN R1<15:08>
                        ;       R-WHAMI = RECEIVED BM WHAMI IN R3
                        ;       E-WHAMI = EXPECTED BM WHAMI IN R2
                        BR      TST2                ;;  ;ON TO NEXT TEST

                        ;*COMPARE FLAGS (EXPD):(RCVD)
10$:                    CMP     R1,R0                   ;
                        BEQ     TST2                ;;  ;BR IF OK - NEXT TEST
                        ERROR   25                      ;BM FLAGS / BAD INIT, WHAMI OK
                        ;"BM WHAMI/FLAGS INIT ERROR"
                        ;       R-FLAGS = RECEIVED BM FLAGS<8:0> IN R0<15:08>
                        ;       E-FLAGS = EXPECTED BM FLAGS<8:0> IN R1<15:08>
                        ;       R-WHAMI = RECEIVED BM WHAMI IN R3
                        ;       E-WHAMI = EXPECTED BM WHAMI IN R2


;;*****************************************************************
;;*****************************************************************
;;*****************************************************************

;;*****************************************************************
;*TEST 2          ...ENABLE BM MICROBREAK TRAP-TO-4, IN WHAMI.
;;*****************************************************************
TST2:   SCOPE
                        CLR     $TIMES                  ;NO ITER OF THIS TEST
                        CLR     $ERFLG                  ;OR ERRORS EITHER

                        MED     ,RWHAMI                 ;GET IT
                        BIS     #BIT9,R0                ;SET BIT 9
                        MED     ,WWHAMI                 ;AND REWRITE

;;*****************************************************************
;;*****************************************************************
;;*****************************************************************


;;*****************************************************************
;*TEST 3          BM/ FLAGS AND INSTR1 FP DECODE
;
;       THIS TEST CHECKS THAT:
;
```

```
                        ;       1) BM FLAGS<5:4> CAN BE R/W WITH 0/1 PATTERNS
                        ;       2) BM INSTR1 FP DECODE TARGETS TO (0474)-(0477) IN BM
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       R0      -MED R/W, RECEIVED "FLAGS"
                        ;       R1      -EXPECTED "FLAGS"
                        ;       R2      -EXPECTED "UBREAK"
                        ;       R3      -BM UBREAK ADDRESS (INSTR1 FP DECODE TARGET, 0474-0477)
                        ;       R4      -UBREAK FLAG: 1S=NO/0S=YES: BM UBREAK AT INSTR1 FP DECODE
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;       FEXP/K9
                        ;       FMUL/K10
                        ;       FALU/K11
                        ;         [ESSENTIALLY NONE]
                        ;
                        ;       UWORD/K2
                        ;         UCON-FP-LOGIC, UCON-FLAG-LOGIC
                        ;
                        ;       IRDECODE/K3
                        ;         INSTR1-FP-DECODE, BM-UBREAK
                        ;
                        ;;*****************************************************************
004152  000004          TST3:   SCOPE
004154  012705  004326          MOV     #40$,R5                 ;INIT DATA TABLE PTR
                                ;*DATA LOOP ENTERS HERE
004160  012501          1$:     MOV     (R5)+,R1                ;GET "FLAGS" DATA
004162  001472                  BEQ     TST4            ;;      ;IFF ALL ZERO, DONE WITH TEST
004164  012502                  MOV     (R5)+,R2                ;GET BM UBRK ADDRESS
                                                                ;
004166  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                                ;
004170  104410  004216          SETDW   ,14$                    ;ENABLE PROC HUNG ESCAPE, WITH CLOCK
004174  010237  177770          MOV     R2,CPUBRK               ;LOAD UBRK REGISTER IN BM
004200  104414  004220          SETUB   ,15$                    ;ENABLE PROC UBRK EXIT
004204  010100                  MOV     R1,R0                   ;GET FLAGS TO WRITE
004206  076600  000344          MED     ,WFLAG                  ;SETUP FLAGS
004212  005004                  CLR     R4                      ;CLEAR UBRK FLAG
                                
004214  170003                  LDUB                            ;EXEC THE FP INSTR
                                
004216  005104          14$:    COM     R4                      ;ENTER HERE IF NO UBRK, OR PROC HUNG TIMEOUT
004220  104411          15$:    CLRDW                           ;MAKE TIMEOUT AN ERROR NOW
004222  005704                  TST     R4                      ;TEST FOR UBREAK (0S=YES)
004224  001412                  BEQ     20$                     ;BR IF THERE WAS A UBREAK
                                
                                ;NO UBREAK AT INSTR1/FP DECODE
```

```
1007  004226  076600  000144          MED     ,RFLAG                  ;SO GET ACTUAL FLAGS
1008  004232  105000                  CLRB    R0                      ;ZAP FPS PART
1009  004234  104415                  CLRUB                           ;AND DISABLE FURTHER UBREAKS
1010
1011  004236  020001                  CMP     R0,R1                   ;FLAGS LOADED OK ?
1012  004240  001402                  BEQ     16$                     ;BR IF YES
1013  004242  104026                  ERROR   26                      ;FLAGS LOADED/READ WRONG
1014                                  ;"BM FLAGS R/W ERROR"
1015                                  ;       E-FLAGS = EXPECTED BM FLAGS<8:0> IN R1<15:08>
1016                                  ;       R-FLAGS = RECEIVED BM FLAGS<8:0> IN R0<15:08>
1017  004244  000424                  BR      25$                     ;NEXT
1018
1019                                  ;FORCE "NO-UBRK" ERROR
1020  004246  104027          16$:    ERROR   27                      ;ALSO NO UBREAK ERROR
1021                                  ;"BM INSTR1/FP-DECODE ERROR; FLAGS OK"
1022                                  ;       BMUBRK  = BASE MACHINE EXPECTED MICROBREAK ADDRESS IN R2<11:00>
1023                                  ;       R-FLAGS = RECEIVED BM FLAGS<8:0> IN R0<15:08>
1024  004250  000422                  BR      25$                     ;
1025
1026                                  ;PROC DID UBREAK
1027  004252  104415          20$:    CLRUB                           ;DISABLE FURTHER UBREAKS
1028
1029  004254  076600  000103          MED     ,LOGCUA                 ;GET LOGGED CUA (MICROADDRESS)
1030  004260  072027  177775          ASH     #-3,R0                  ;ALIGN TO BIT<11:00>
1031  004264  042700  170000          BIC     #^C7777,R0              ;ZAP 4.0.BITS
1032  004270  010003                  MOV     R0,R3                   ;SAVE RECEIVED CUA IN R3
1033
1034  004272  076600  000104          MED     ,LOGFLG                 ;GET LOGGED FLAGS/INTR
1035  004276  105000                  CLRB    R0                      ;ZAP NON-FLAGS
1036
1037                                  ;*COMPARE "FLAGS" (LOGGED) (EXPD):(RCVD)
1038  004300  020100                  CMP     R1,R0                   ;AGREE ?
1039  004302  001401                  BEQ     .+4                     ;BR IF OK
1040  004304  104030                  ERROR   30                      ;NOPE - FLAGS LOADED/LOGGED WRONG
1041                                  ;"BM INSTR1/FP-DECODE OR FLAGS ERROR"
1042                                  ;       R-FLAGS = RECEIVED BM LOG-FLAGS<8:0> IN R0<15:08>
1043                                  ;       E-FLAGS = EXPECTED BM LOG-FLAGS<8:0> IN R1<15:08>
1044                                  ;       R-UADDR = RECEIVED BM MICROADDRESS IN R3<11:00>
1045                                  ;       E-UADDR = EXPECTED BM MICROADDR IN R2<11:00>
1046  004306  000403                  BR      25$                     ;NEXT
1047
1048                                  ;*COMPARE "CUA/UBRK ADDRESS" (LOGGED) (EXPD):(RCVD)
1049  004310  020203                  CMP     R2,R3                   ;AGREE ?
1050  004312  001401                  BEQ     .+4                     ;BR IF YES
1051  004314  104030                  ERROR   30                      ;NOPE - CUA LOGGED WRONG ???
1052                                  ;"BM INSTR1/FP-DECODE OR FLAGS ERROR"
1053                                  ;       R-FLAGS = RECEIVED BM LOG-FLAGS<8:0> IN R0<15:08>
1054                                  ;       E-FLAGS = EXPECTED BM LOG-FLAGS<8:0> IN R1<15:08>
1055                                  ;       R-UADDR = RECEIVED BM MICROADDRESS IN R3<11:00>
1056                                  ;       E-UADDR = EXPECTED BM MICROADDR IN R2<11:00>
1057
1058                                  ;EVERYTHING WENT OK - ON TO NEXT DATA SET
1059  004316  005000          25$:    CLR     R0                      ;ZAP UBRK ENABLE AND FLAGS
1060  004320  076600  000344          MED     ,WFLAG                  ;
1061  004324  000715                  BR      1$                      ;NEXT
1062
```

```
1063                                    ;////////////////////////////////////////////////////////////////////
1064                                    ;
1065                                    ;       DATA FOR ABOVE TEST:
1066                                    ;
1067                                    ;               -FLAG-  UBRK    FLAG<8/5:4>     COMENTS
1068
1069  004326  100000  000474    40$:    .WORD   100000, 0474    ;1/00           WFP*VALID
1070
1071  004332  110000  000475            .WORD   110000, 0475    ;1/10           HFP*VALID
1072
1073  004336  114000  000477            .WORD   114000, 0477    ;1/11           HFP*INVALID
1074
1075  004342  104000  000476            .WORD   104000, 0476    ;1/01           WFP*INVALID
1076
1077  004346  000000                    .WORD   0               ;DONE           [EXIT WITH ABOVE STATE]
1078
1079
1080                                    ;;*****************************************************************
1081                                    ;*TEST 4         BM/ FP CNST RESTORE
1082                                    ;
1083                                    ;       THIS TEST VERIFIES THE FP CONSTANT RESTORE PROCEDURE
1084                                    ;       OF WFP/HFP BM INSTR1 DECODE.  THE INVALID-FP-CONSTANT FLAG
1085                                    ;       [FLAG<4>] IS SET, INDICATING THE FP CONSTANTS ARE INVALID.
1086                                    ;       THE ACTUAL FP CONSTANTS ARE DESTROYED BY USING THE MED INSTRUCTION
1087                                    ;       TO WRITE ZEROES INTO CSP (00) -> (13).  THE TEST THEN EXECUTES
1088                                    ;       A -WFP- INSTRUCTION, AND CHECKS THAT:
1089                                    ;
1090                                    ;         1) FLAG<4> IS CLEARED AFTER THE RESTORE
1091                                    ;         2) EACH OF THE CONSTANTS, IN CSP(00)-(05), (07)-(13)
1092                                    ;            IS CHECKED FOR VALIDITY.
1093                                    ;
1094                                    ;       -------
1095                                    ;       REGISTER/LOCATION USE:
1096                                    ;
1097                                    ;       R0      -TEMP, RECEIVED FP CNST, RECEIVED FLAGS
1098                                    ;       R1      -TEMP
1099                                    ;       R2      -EXPECTED FP CNST
1100                                    ;       R3      -(NU)
1101                                    ;       R4      -(NU)
1102                                    ;       R5      -DATA TABLE PTR
1103                                    ;
1104                                    ;       ----------
1105                                    ;       MODULE/ERROR INFO:
1106                                    ;
1107                                    ;       FNUA/K8
1108                                    ;       FEXP/K9
1109                                    ;       FMUL/K10
1110                                    ;       FALU/K11
1111                                    ;         [ESSENTIALLY NONE]
1112                                    ;
1113                                    ;       UWORD/K2
1114                                    ;         UCON-FLAG-LOGIC
1115                                    ;
1116                                    ;       IRDECODE/K3
1117                                    ;         INSTR1-FP-DECODE
1118                                    ;
```

```
1119                                    ;       DATAPATH/K4
1120                                    ;         CSP ADDRESSING FOR FP CONSTANTS
1121                                    ;
1122                                    ;;*****************************************************************
1123  004350  000004                    TST4:   SCOPE
1124
1125  004352  012700  004000                    MOV     #004000,R0              ;SET WFP*INVALID
1126  004356  076600  000344                    MED     ,WFLAG                  ;INTO FLAGS
1127
1128                                            ;NOW ZAP ALL THE FP CONSTANTS
1129  004362  005000                            CLR     R0                      ;INTO ZEROES
1130  004364  012701  000014                    MOV     #14,R1                  ;(14) SP'S
1131  004370  012737  000300  004400            MOV     #WCSP00,2$              ;GET MED CODE
1132  004376  076600                    1$:     MED                             ;DO A WRITE TO THE CSP:
1133  004400  000300                    2$:     WCSP00                          ;USING THIS CODE
1134  004402  005237  004400                    INC     2$                      ;BUMP CODE ALONG
1135  004406  077105                            SOB     R1,1$                   ;AND LOOP
1136
1137                                            ;NOW EXEC A WFP*INVALID MODE FP INSTR
1138  004410  170127  040000                    LDFPS   #040000                 ;SHOULD RESTORE CONSTANTS
1139
1140  004414  076600  000144                    MED     ,RFLAG                  ;GET FLAGS IN H.O.B.
1141  004420  032700  177400                    BIT     #UB,R0                  ;TEST UPPER-BYTE FOR ALL ZERO FLAGS
1142  004424  001401                            BEQ     .+4                     ;FLAG<4> SHOULD BE "0"
1143                                                                            ;  IF CONSTANTS RESTORED
1144  004426  104031                            ERROR   31                      ;ELSE ERROR: F<4> NOT CLEARED
1145                                            ;"BM FLAG4=1 AFTER CSP FP-CNST RESTORE"
1146                                            ;       R-FLAGS = RECEIVED BM FLAGS<8:0> AFTER CSP FP-CNST
1147                                            ;                 RESTORE ROUTINE EXECUTED
1148
1149                                            ;-------NOW CHECK EACH CONSTANT IS CORRECT-------
1150  004430  012705  004506                    MOV     #40$,R5                 ;PTR TO DATA
1151
1152                                            ;*DATA LOOP ENTERS HERE*
1153  004434  005237  002640            10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
1154  004440  012501                            MOV     (R5)+,R1                ;GET R1=CSP LOCATION
1155  004442  100450                            BMI     TST5            ;;      ;IF -1, DONE WITH TEST
1156  004444  012502                            MOV     (R5)+,R2                ;GET EXPECTED FP CNST
1157                                                                            ;
1158  004446  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
1159                                            ;--------ERROR-LOOP-ENTERS-HERE--------
1160                                                                            ;
1161  004450  010100                            MOV     R1,R0                   ;MAKE CSP** INTO MED CODE
1162  004452  052700  000100                    BIS     #100,R0                 ;  (100)-(113)
1163  004456  010037  004466                    MOV     R0,15$                  ;STORE IN MEMORY
1164  004462  170000                    63$:    CFCC                            ;EXEC WFP INSTR TO RESTORE CONSTANTS AGAIN,
1165                                                                            ;  IN CASE OPR HAS BEEN FOOLING AROUND WITH
1166                                                                            ;  ANY BUTTONS ON THE OPERATOR'S CONSOLE
1167  004464  076600                            MED                             ;EXEC MED READ OF:
1168  004466  000100                    15$:    RCSP00                          ;  THE CSP LOCATN
1169  004470  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
1170  004474  001372                            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
1171                                                                            ;
1172
1173                                            ;NOW COMPARE (EXPD):(RCVD) FP CNST
1174  004476  020200                            CMP     R2,R0                   ;EQUAL ?
```

```
1175  004500  001755                    BEQ     10$                     ;BR FOR NEXT LOOP IF OK
1176
1177  004502  104032                    ERROR   32                      ;ELSE ERROR:  BAD FP CNST READ
1178                                    ;"BM BAD FP-CNST IN CSP"
1179                                    ;       CSPADR  = CSP ADDRESS REFERENCED, (00) -> (13)
1180                                    ;       E-FPCNST= EXPECTED FP CNST AT THIS ADDRESS
1181                                    ;       R-FPCNST= RECEIVED FP CNST READ FROM THIS ADDRESS
1182                                    ;               =(000000) IF CNST NOT RESTORE AND NO ERROR LOG
1183  004504  000753                    BR      10$                     ;NEXT
1184
1185                                ;////////////////////////////////////////////////////////////////
1186                                ;
1187                                ;       DATA TABLE USED IN ABOVE TEST:
1188                                ;
1189                                ;               CSP     FP-CNST
1190
1191  004506  000000  077600        40$:    .WORD   00,     077600
1192
1193  004512  000001  000010                .WORD   01,     000010
1194
1195  004516  000002  020000                .WORD   02,     020000
1196
1197  004522  000003  000004                .WORD   03,     000004
1198
1199  004526  000004  050000                .WORD   04,     050000
1200
1201  004532  000005  054000                .WORD   05,     054000
1202
1203  004536  000007  024000                .WORD   07,     024000
1204
1205  004542  000010  177400                .WORD   10,     177400
1206
1207  004546  000011  177600                .WORD   11,     177600
1208
1209  004552  000012  100000                .WORD   12,     100000
1210
1211  004556  000013  000200                .WORD   13,     000200
1212
1213  004562  177777                        .WORD   -1
1214
1215
1216
1217                                ;;**************************************************************
1218                                ;*TEST 5         BM/WFP ILLEGAL INTERNAL ADDRESS TEST
1219                                ;
1220                                ;       THIS TEST CHECKS THE BASE MACHINE FACILITY TO ABORT
1221                                ;       FLOATING POINT INSTRUCTIONS (WARM AND HOT) WHICH REFERENCE
1222                                ;       PROCESSOR INTERNAL ADDRESSES FOR OPERAND STORE/FETCH.
1223                                ;
1224                                ;       THIS TEST IS INDEPENDENT OF THE FP11-E PROCESSOR, AND IS
1225                                ;       EXECUTED IN WARM FLOATING POINT MODE.  AN ERROR
1226                                ;       CONDITION INDICATES SOMETHING IS WRONG IN THE BASE PROCESSOR.
1227                                ;
1228                                ;       ----------
1229                                ;       REGISTER/LOCATION USE:
1230                                ;
```

```
                        ;       $REG0   -SAVES OLD ERRVEC(PC)
                        ;       $REG1   -SAVES OLD ERRVEC(PS)
                        ;
                        ;       R0      -RCV'D CPU ERROR REGISTER AFTER
                        ;       R1      -EXP'D CPU ERROR REGISTER AFTER (000001)=INTRNL.ADDR.ERR
                        ;       R2      -FP INSTR UNDER TEST
                        ;       R3      -FLAG (0=TRAP/-1=NO.TRAP)
                        ;       R5      -SAVE OLD SP
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       TIMING/K6
                        ;       STATUS/K7
                        ;         BUS.CYCLE, INTERNAL.ADDR.DETECT, JAMUPP.LOGIC
                        ;
                        ;       FNUA/FEXP/FALU/FMUL
                        ;         [NONE, YET]
                        ;
                        ;*****************************************************************
004564  000004          TST5:   SCOPE
                                ;
004566  013737  000004  001302          MOV     @#ERRVEC+0,$REG0        ;SAVE OLD ERRVEC PC/PS
004574  013737  000006  001304          MOV     @#ERRVEC+2,$REG1        ;
004602  010605                        MOV     SP,R5                   ;SAVE OLD SP
004604  104417                        ZAPWFP                          ;INIT AND ENABLE WARM
004606  170127  040000                LDFPS   #040000                 ;INTR-DISAB/F-MODE
004612  012705  000001                MOV     #000001,R1              ;EXP'D CPUERR = INTRNL.ADDR.ERR
004616  005037  000006                CLR     @#ERRVEC+2              ;IF TRAP, USE PRO
                                ;
                        ;-------INTERNAL.ADDRESS.ERROR WITH "DATI.NOINTERNAL"-------
                                ;
004622  013702  004640                MOV     11$,R2                  ;GET INSTRUCTION
004626  012737  004652  000004        MOV     #15$,@#ERRVEC+0         ;TRAP-TO-4 GOES HERE
                                ;
004634  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                        ;--------ERROR-LOOP-ENTERS-HERE--------
                                ;
004636  005003                        CLR     R3                      ;CLEAR TRAP FLAG
004640  170537  177570          11$:  TSTF    @#177570                ;DATI.NOINT WITH ADDR(DISPLAY/MMR0)
004644  005000                        CLR     R0                      ;DIDN'T TRAP, FORCE CPUERR=(000000)
004646  005103                        COM     R3                      ;SET NO TRAP
004650  000403                        BR      16$                     ;CONT.
                                ;
004652  013700  177766          15$:  MOV     CPUERR,R0               ;TRAPPED, GET CPUERR
004656  010506                        MOV     R5,SP                   ;AND RESTORE SP
                                ;
004660  020001                  16$:  CMP     R0,R1                   ;CHECK CPUERR IS AS EXPECTED
004662  001401                        BEQ     20$                     ;BR IF WAS INTRNL.ADDR .ERR
004664  104046                        ERROR   46                      ;ELSE ERROR
                                        ;"BM/WFP ILLEGL.INTRNL.ADDR ERR"
                                        ;       FPINST = FP INSTR UNDER TEST, "TSTF/170537"=DATI.NOINT, "CLRF/170437"=DA
                                                                                                                        TO
                                        ;       E-CPUERR = EXP'D CPUERR REG, ILL.INTRNL.ADDR=(000001)
                                        ;       R-CPUERR = RCV'D CPUERR
                                        ;       0=TRAP/-1=NO.TRAP = FLAG INDICATING TRAP-TO-4 OCCURED
```

```
1286
1287                                    ;-------INTERNAL.ADDRESS.ERROR WITH "DATO"-------
1288
1289  004666  013702  004704          20$:  MOV     21$,R2                  ;GET INSTRUCTION
1290  004672  012737  004716  000004        MOV     #25$,@#ERRVEC+0         ;TRAP-TO-4 GOES HERE
1291
1292  004700  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
1293                                        ;--------ERROR-LOOP-ENTERS-HERE--------
1294
1295  004702  005003                        CLR     R3                      ;CLEAR TRAP FLAG
1296  004704  170437  177570          21$:  CLRF    @#177570                ;DATO WITH ADDR(DISPLAY/MMR0)
1297  004710  005000                        CLR     R0                      ;DIDN'T TRAP, FORCE CPUERR=(000000)
1298  004712  005103                        COM     R3                      ;SET NO TRAP
1299  004714  000403                        BR      26$                     ;CONT.
1300
1301  004716  013700  177766          25$:  MOV     CPUERR,R0               ;TRAPPED, GET CPUERR
1302  004722  010506                        MOV     R5,SP                   ;AND RESTORE SP
1303
1304  004724  020001                  26$:  CMP     R0,R1                   ;CHECK CPUERR IS AS EXPECTED
1305  004726  001401                        BEQ     30$                     ;BR IF WAS INTRNL.ADDR .ERR
1306  004730  104046                        ERROR   46                      ;ELSE ERROR
1307                                        ;"BM/WFP ILLEGL.INTRNL.ADDR ERR"
1308                                        ;       FPINST = FP INSTR UNDER TEST, "TSTF/170537"=DATI.NOINT. "CLRF/170437"=DA
                                                                                                                          TO
1309                                        ;       E-CPUERR = EXP'D CPUERR REG, ILL.INTRNL.ADDR=(000001)
1310                                        ;       R-CPUERR = RCV'D CPUERR
1311                                        ;       0=TRAP/-1=NO.TRAP = FLAG INDICATING TRAP-TO-4 OCCURED
1312                                        ;
1313                                        ;----NOW RESTORE OLD ERRVEC-----
1314  004732  010506                  30$:  MOV     R5,SP                   ;RESTORE SP
1315  004734  013737  001304  000006        MOV     $REG1,@#ERRVEC+2        ;RESTORE PS
1316  004742  013737  001302  000004        MOV     $REG0,@#ERRVEC+0        ;RESTORE PC
1317  004750  104416                        ZAPHFP                          ;RESET TO FP11-E ENABLED
1318
1319
1320
1321                                    ;*****************************************************************
1322                                    ;*TEST 6        HFP/BM:  FLPGO-FPACK; SRVC-GRANT
1323                                    ;
1324                                    ;       THE FOLLOWING TEST CONSISTS OF TWO SECTIONS:
1325                                    ;
1326                                    ;       1) TEST OF LDUB/STFPS/LDFPS/STST DECODE BY HFP, AND THAT
1327                                    ;          THE BM/HFP ARE ABLE TO INTERACT VIA FLPGO/FPACK.  THIS
1328                                    ;          INCLUDES INSURING THE HFP WILL RESPOND, AND NOT HANG
1329                                    ;          THE BM, WHICH IS WAITING FOR AN 'FPACK'.
1330                                    ;
1331                                    ;       2) TEST OF THE HFP UBREAK LOGIC, AND HFP/BM-SERVICE
1332                                    ;          REQUEST INTERACTION.  ACTUAL CODE PASSED FROM HFP TO
1333                                    ;          THE BM NOT TESTED FOR VALIDITY HERE.
1334                                    ;
1335                                    ;       ----------
1336                                    ;       REGISTER/LOCATION USE:
1337                                    ;
1338                                    ;       UBCNTR  -COUNTS # TIMES BM UBROKE AT (4222) IN HFP.SRVC CODE
1339                                    ;       $REG0-3 -(TEMPS)
1340                                    ;
```

```
1341                            ;       R0      -BM SERVICE PORT (RECEIVED)
1342                            ;       R1      -(NU)
1343                            ;       R2      -(TEMP)
1344                            ;       R3      -HFP UBRK @ PREP1 ADDR, =(022)
1345                            ;       R4      -(NU)
1346                            ;       R5      -(NU)
1347                            ;
1348                            ;       ----------
1349                            ;       MODULE/ERROR INFO:
1350                            ;
1351                            ;       FNUA/K8
1352                            ;         FPINMUX, FIR-A/B, FP-INSTR/INSTRCVD-LOGIC, FIR-CLK-A/B,
1353                            ;         FNUA-GENERATION/JREG-LOGIC, BUTA(SUBR/RETURN),
1354                            ;         HFP-UBRK, CROM/LATCHES
1355                            ;
1356                            ;       FEXP/K9
1357                            ;         HFP/BM-INTERFACE-LOGIC, FPBRAN<2:0>/DECODE-LOGIC,
1358                            ;         HFP-SRVC-REQ/GRANT-LOGIC, JAMUPP-LOGIC, CROM/LATCHES
1359                            ;
1360                            ;       FMUL/K10
1361                            ;         [ESSENTIALLY NONE]
1362                            ;
1363                            ;       FALU/K11
1364                            ;         [ESSENTIALLY NONE]
1365                            ;
1366                            ;       UWORD/K2
1367                            ;         UCON-FP, FLPGO
1368                            ;
1369                            ;       IRDECODE/K3
1370                            ;         BUTR(FPACK-SRVC)
1371                            ;
1372                            ;       TIMING/K6
1373                            ;         HFP-SRVC/SERVICE
1374                            ;
1375                            ;       STATUS/K7
1376                            ;         HFP-SRVC/STATUS-MUX
1377                            ;
1378                            ;*****************************************************************
1379  004752  000004            TST6:   SCOPE
1380
1381  004754  104410  005000            SETDW   ,1$                     ;ESCAPE ADDR IF PROC HANGS
1382  004760  104416                    ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
1383  004762  012703  040000            MOV     #040000,R3              ;FPS W/ FID=1, FMM=0, ZEROES FOR LDUB
1384  004766  170103                    LDFPS   R3                      ;LOAD FPS, BUT ALSO
1385  004770  170003                    LDUB                            ;  EXEC THE FOUR STATUS INSTRUCTIONS
1386  004772  170200                    STFPS   R0                      ;  TO SEE THAT NONE OF THEM HANGS
1387  004774  170302                    STST    R2                      ;  THE BM/HFP HANDSHAKE SEQUENCE.
1388  004776  000401                    BR      10$                     ;OK IF GOT TO HERE
1389
1390  005000  104033            1$:     ERROR   33                      ;PROC HUNG BY HFP AT ONE
1391                                                                    ;  OF THE ABOVE INSTRUCTIONS
1392                                    ;"CM HUNG DURING HFP FLPGO/FPACK SEQ"
1393                                    ;       OLD-SP = SP AFTER TRAP TO LINE CLOCK ROUTINE
1394                                    ;       OLD-PC = PC BEFORE TRAP, POINTS AT OFFENDING INSTRUCTION
1395                                    ;       OLD-PS = PS BEFORE TRAP
1396
```

```
1397  005002  012737  004222  177770  10$:    MOV     #4222,CPUBRK            ;BM @ "HFPTRAP7" IN BM HFP SRVC CODE
1398  005010  012703  000022                  MOV     #022,R3                 ;(022)="PREP1" IN HFP
1399
1400                                          ;*NOW CHECK THAT THE HFP IS ABLE TO UBREAK, AND REQUEST FP SRVC
1401                                          ;
1402  005014  104406                         ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
1403                                          ;--------ERROR-LOOP-ENTERS-HERE--------
1404                                          ;
1405  005016  104411                         CLRDW                           ;MAKE A PROC HANG AN ERROR
1406  005020  104416                         ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
1407  005022  005037  002630                 CLR     UBCNTR                  ;START COUNT AT ZERO
1408  005026  104414  000000                 SETUB   ,0                      ;NO ESCAPE, BUT ENABLE AND COUNT
1409
1410  005032  170003                         LDUB                            ;INTO HFP UBRK
1411  005034  104416                         ZAPHFP                          ;CLEAR ANY EFFECTS
1412
1413  005036  052737  000340  177776         BIS     #PR7,PS                 ;SET PR7 TO IGNORE FP SRVC, AND ALSO
1414                                          ;  LOCK OUT LINE CLOCK
1415
1416  005044  170127  040020                 LDFPS   #040020                 ;SET FMM=1
1417
1418                                          ;HFP UBRKS AT START OF THIS INSTR:
1419  005050  012700  104000                 MOV     #104000,R0              ;SET BM UBRK, DISB HFP, CSP CNST INVALID
1420                                          ;HFP SERVICE REQ / BM IGNORE SINCE PR7*-FP IN IR
1421
1422  005054  076600  000344                 MED     ,WFLAG                  ;ZAP HFP ENABLE, KEEP BM UBRK ENABL
1423                                          ;HFP SERVICE REQ / BM IGNORE SINCE PR7*-FP IN IR
1424
1425  005060  170337  001302                 STST    $REGO                   ;EXEC -WFP- STATUS (IE, NO HFP SYNC)
1426                                          ;HFP SERVICE REQ / BM HONOR SINCE PR7*FP IN IR
1427                                          ;STATUS IN $REGO/1 IS STATUS BEFORE HFP SERVICE
1428                                          ;UBCNTR=1 AFTER BM BREAK
1429
1430                                          ;HFP AGAIN UBRKS AT START OF NEXT INSTR:
1431  005064  076600  000141                 MED     ,RSERVC                 ;GET BM SERVICE PORT
1432                                          ;HFP SERVICE REQ / BM IGNORE SINCE PR7*-FP IN IR
1433
1434                                                                          ;NEXT INSTR DISABLES FMM=(0), HFP UBRK OFF
1435  005070  170127  040000                 LDFPS   #040000                 ;EXEC -WFP- STATUS (IE, NO HFP SYNC)
1436                                          ;HFP SERVICE REQ / BM HONOR SINCE PR7*FP IN IR
1437                                          ;UBCNTR=2 NOW AFTER BM UBRK
1438
1439                                          ;FMM=(0) NOW SO SHOULD BE NO FURTHER HFP UBRK/SRVC REQUESTS
1440  005074  170337  001306                 STST    $REG2                   ;EXEC -WFP- STATUS (IE, NO HFP SYNC)
1441                                          ;SHOULD BE NO HFP SRVC PENDING NOW
1442
1443  005100  105037  177776                 CLRB    PS                      ;TURN LINE CLOCK BACK ON
1444
1445                                          ;CHECK FP SRVC WAS SET WHEN "SERVICE" PORT WAS READ
1446  005104  020027  100350                 CMP     R0,#100350              ;FP-SRVC-H IN BIT<03>H
1447  005110  001004                         BNE     20$                     ;ERROR IF DIFFERENT
1448
1449                                          ;CHECK FP SRVC WAS HONORED TWICE BY BM (UBCNTR)
1450  005112  023727  002630  000002         CMP     UBCNTR,#2               ;TWICE ?
1451  005120  001401                         BEQ     30$                     ;BR IF OK
1452
```

```
1453  005122  104034                20$:    ERROR   34                      ;HFP INSTR-RCVD ERROR, AND/OR
1454                                          ;"HFP SRVC GRANT ERROR"
1455                                          ;       **SRVC = COUNT OF NUMBER OF TIMES BM MICROBREAK AT (4222)
1456                                          ;                (@HFPPTRAP7) OCCURRED
1457                                          ;       BMSRVC = RECEIVED SERVICE PORT OF STATUS MUX IN R0<15:00>
1458                                                  ;HFP FP-SRVC REQ ERROR
1459
1460  005124  104415                30$:    CLRUB                           ;MAKE BM UBRK ILLEGAL
1461  005126  104416                        ZAPHFP                          ;INIT HFP, SET FMM=0
1462
1463
1464
1465                                  ;;*****************************************************************
1466                                  ;*TEST 7        BM/ HFP UBREAK SRVC CODE
1467                                  ;
1468                                  ;       THIS TEST DOES ESSENTIALLY THE SAME THING AS THE PREVIOUS ONE;
1469                                  ;       HOWEVER, HERE THE HFP-SRVC CONDITION IS ALLOWED TO PROCEED TO
1470                                  ;       COMPLETION, TO CHECK THAT THE HFP UNIT IS ABLE TO PASS A
1471                                  ;       MEANINGFUL CODE BACK TO THE BASE MACHINE.
1472                                  ;
1473                                  ;       ----------
1474                                  ;       REGISTER/LOCATION USE:
1475                                  ;
1476                                  ;       R0      -RECEIVED "FPSHI*FEC" AFTER HFP UBRK
1477                                  ;       R1      -(NU)
1478                                  ;       R2      -(NU)
1479                                  ;       R3      -HFP/PREP1 UBRK ADDR
1480                                  ;       R4      -(NU)
1481                                  ;       R5      -SP SAVED HERE
1482                                  ;
1483                                  ;       ----------
1484                                  ;       MODULE/ERROR INFO:
1485                                  ;
1486                                  ;       FNUA/K8
1487                                  ;         FPEMITF-DRIVERS/ENABL, FSPAD[A,B]-ENABL/ADDRS,
1488                                  ;         JREG/NUA-GENERATE, BUTA(SUBR/RETURN), FALU-CONTROL, CROM/LATCHES
1489                                  ;
1490                                  ;       FEXP/K9
1491                                  ;         FPOUTMUX-ENABL, FSPAD[A,B]-WRITE, AR-CLK, CROM/LATCHES
1492                                  ;
1493                                  ;       FMUL/K10
1494                                  ;         MNET-ALU/PASS-A-SIDE, FPOUTMUX-DATA
1495                                  ;
1496                                  ;       FALU/K11
1497                                  ;         FSPAD[A,B]-WRITE/ENABLE, AR-LOAD/READ
1498                                  ;
1499                                  ;*****************************************************************
1500  005130  000004                TST7:   SCOPE
1501
1502  005132  104416                        ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
1503                                                                        ;  ALSO FMM=(0)
1504  005134  005000                        CLR     R0                      ;FOR FLAGS
1505  005136  005003                        CLR     R3                      ;FOR HFP UBRK ADDR,
1506  005140  170003                        LDUB                            ;  POINT AWAY FROM PREP0/1/2
1507  005142  012703  000022               MOV     #22,R3                  ;(22)=PREP1 IN HFP
1508  005146  010605                        MOV     SP,R5                   ;SAVE SP
```

```
1509  005150  170127  047420                LDFPS   #047420                 ;FPS WITH:  FER=0, FID=1,
1510                                                                          ;  F<ENABL>=1S, FMM=1
1511  005154  170003                       LDUB                            ;PUT ADDR(PREP1) INTO HFP UBRK
1512
1513                                        ;HFP UBRK AT START OF THIS INSTR:
1514  005156  076600  000344                MED     ,WFLAG                  ;ZAP HFP ENABL FLAG
1515                                        ;HFP SRVC REQ / HONOR SINCE -PR7
1516                                ;///////////////////////////////////////////////////////////////////
1517                                        ;READ CODE FROM THE HFP/UBRK SRVC REQ [WHICH SHOULD = (07)]
1518                                        ;THE BM HFP SRVC ROUTINE THEN DOES A BUTR(SR3-0) ON THIS CODE,
1519                                        ;WITH A BASE ADDRESS OF (4540).  THUS THE BM CAN BRANCH TO
1520                                        ;A NUMBER OF DIFFERENT LOCATIONS [IE, (4540)-(4557)] FOR A
1521                                        ;RETURNED CODE VALUE OF (00)-(17) RESPECTIVELY.  SOME OF THESE
1522                                        ;WILL CAUSE THE BM PROCESSOR TO BRANCH OFF INTO AN INDETERMINATE
1523                                        ;STATE; OTHERS WILL CAUSE OTHER FP SERVICE CONDITIONS TO BE
1524                                        ;SIGNALED.  THE FOLLOWING TABLE SUMMARIZES THE POSSIBILITIES:
1525                                        ;
1526                                        ;CODE  ADDR/SYMB-LABL   COMMENTS
1527                                        ;----  -------------   ----------------------------------------
1528                                        ; 00   4540/LDCPW14    --> FET01, NO FP SRVC CODE RECORDED
1529                                        ;----  -------------   ----------------------------------------
1530                                        ; 01   4541/OPCODERR   FEC/02 CODE RETURNED
1531                                        ; 02   4542/ZERODIV    FEC/04 CODE RETURNED
1532                                        ; 03   4543/CONVTRAP   FEC/06 CODE RETURNED
1533                                        ; 04   4544/VTRAP5     FEC/10 CODE RETURNED
1534                                        ; 05   4545/UFLOTRAP   FEC/12 CODE RETURNED
1535                                        ; 06   4546/NZERTRAP   FEC/14 CODE RETURNED
1536                                        ; 07   4547/MAINTRAP   FEC/16 CODE RETURNED ****EXPECTED****
1537                                        ;----  -------------   ----------------------------------------
1538                                        ; 10   4550/LDCPW17    --> FET01, NO FP SRVC CODE RECORDED
1539                                        ; 11   4551/CTRAP2     --> FET01, NO FP SRVC CODE RECORDED
1540                                        ; 12   4552/FFLT5      GENERATE ODD ADDRS ERROR, TRAP-TO-4
1541                                        ; 13   4553/FFLT6      GENERATE ODD ADDRS ERROR, TRAP-TO-4
1542                                        ; 14   4554/NROUNDEND3  \
1543                                        ; 15   4555/NROUNDEND4   \
1544                                        ; 16   4556/NROUNDEND5    )- DOES A BUTA(RETURN) TO ... ???
1545                                        ; 17   4557/NROUNDEND6   /       [ENTERS A SUNSET LOOP ???]
1546                                        ;----  -------------   ----------------------------------------
1547                                ;///////////////////////////////////////////////////////////////////
1548
1549                                        ;GETTING BACK TO THIS POINT MEANS NOTHING DEADLY HAPPENS ...
1550                                        ;HFP UBRKS AGAIN ON STARTING THIS INSTR:
1551  005162  170127  047400                LDFPS   #047400                 ;SET FMM=(0) TO DISABL UBRKS, KEEP ENABLES
1552                                        ;HFP SRVC REQ / HONOR SINCE -PR7
1553                                        ;REQ IS SERVICED AGAIN, JUST AS IT WAS ABOVE
1554
1555  005166  010506                       MOV     R5,SP                   ;RESET OUR SP TO A GOOD VALUE
1556
1557  005170  076600  000036                MED     ,RFEC                   ;GET FPSHI*FEC REGISTER
1558  005174  020027  147416                CMP     R0,#147416              ;SHOULD HAVE SET FER=(1), FEC=(16)
1559  005200  001401                       BEQ     40$                     ;BR IF OK
1560
1561  005202  104035                       ERROR   35                      ;BAD CODE RETURNED FROM HFP
1562                                        ;"BAD UBRK CODE FROM HFPP [CODE#07/FEC#16]"
1563                                        ;       R-FPSHI/FEC = FPSHI<15:08> IN R0<15:08>,
1564                                        ;                     FEC<03:00> IN R0<07:00>
```

```
1565                                    ;                       AFTER FP SRVC REQ HONORED BY BM
1566
1567  005204  104416                    40$:    ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
1568
1569
1570                                    ;;*****************************************************************
1571                                    ;*TEST 10       HFP ENABLE/DISABLE, FP INSTR DECODE
1572                                    ;
1573                                    ;       THE FOLLOWING TEST CHECKS THE FUNCTIONALITY OF:
1574                                    ;
1575                                    ;         1) HFP LDUB ("LOAD MICROBREAK") INSTR MUST WORK
1576                                    ;
1577                                    ;         2) HFP FP-INSTR-L DECODE LOGIC [FIR<15:12>=(17)]
1578                                    ;
1579                                    ;         3) HFP ENABLE/DISABLE VIA FLAG<5>
1580                                    ;
1581                                    ;       PROCEDURE:
1582                                    ;
1583                                    ;       -FIRB-  FLAG<5> UBF.K@(021)      COMMENTS
1584                                    ;
1585                                    ;       17XXXX     1       YES           FP*ENABLED, ENTER PREP2
1586                                    ;
1587                                    ;       07XXXX     1       NO            -FP*ENABLED, STAY PREP0/1 LOOP
1588                                    ;       13XXXX     1       NO            -FP*ENABLED, STAY PREP0/1 LOOP
1589                                    ;       15XXXX     1       NO            -FP*ENABLED, STAY PREP0/1 LOOP
1590                                    ;       16XXXX     1       NO            -FP*ENABLED, STAY PREP0/1 LOOP
1591                                    ;       17XXXX     0       NO            FP*-ENABLED, STAY PREP0/1 LOOP
1592                                    ;
1593                                    ;       NOTE:  PREP0=(023), PREP1=(022), PREP2=(021)
1594                                    ;
1595                                    ;       ----------
1596                                    ;       REGISTER/LOCATION USE:
1597                                    ;
1598                                    ;       R0      -(TEMP)
1599                                    ;       R1      -EXPECTED FPSHI/FEC AFTER TEST
1600                                    ;       R2      -COPY OF INSTR UNDER TEST
1601                                    ;       R3      -BIT12=FLAG5 DURING TEST
1602                                    ;       R4      -(NU)
1603                                    ;       R5      -DATA TABLE PTR
1604                                    ;
1605                                    ;       ----------
1606                                    ;       MODULE/ERROR INFO:
1607                                    ;
1608                                    ;       FNUA/K8
1609                                    ;         FPINMUX, FIR-A/B, FP-INSTR/INSTRCVD/CLK-FIRA-B,
1610                                    ;         FNUA-GENERATE/JREG-LOGIC, BUTA(SUBR/RETURN), HFP-MICROBREAK,
1611                                    ;         CROM/LATCHES
1612                                    ;
1613                                    ;       FEXP/K9
1614                                    ;         JAMUPP, HFP/BM-INTERFACE, FBRAN<2:0>, FBRAN-DECODE,
1615                                    ;         CROM/LATCHES
1616                                    ;
1617                                    ;       FMUL/K10
1618                                    ;         [ESSENTIALLY NONE]
1619                                    ;
1620                                    ;       FALU/K11
```

```
1621                                    ;          [ESSENTIALLY NONE]
1622                                    ;
1623                                    ;.*************************************************************
1624  005206  000004                    TST10:  SCOPE
1625                                                                    ;
1626                                            ;*FIRST TRY TO LOAD HFP UBREAK REGISTER
1627  005210  104410  005224                      SETDW   ,11$                      ;SETUP PROC HUNG ESCAPE
1628  005214  104416                            ZAPHFP                            ;INIT HFP, LEAVE IT ENABLED
1629  005216  012703  000021                    MOV     #021,R3                   ;R3=UBRK ADDR, PREP2=(021)
1630  005222  170003                            LDUB                              ;TRY HFP UBRK LOAD
1631  005224  012705  005334          11$:      MOV     #40$,R5                   ;PTR TO DATA TABLE FOR TEST
1632  005230  104410  005302                    SETDW   ,23$                      ;SETUP CLOCK ESCAPE, FOR BELOW
1633                                                                            ;
1634                                            ;*DATA LOOP ENTERS HERE*
1635  005234  005237  002640          1$:       INC     DWLOOP                    ;BUMP CLOCK IN.A.LOOP COUNT
1636  005240  104416                            ZAPHFP                            ;RESET HFP PRIOR TO EXIT
1637  005242  012502                            MOV     (R5)+,R2                  ;GET TEST INSTR FROM TABLE
1638  005244  001472                            BEQ     TST11             ;;      ;NEXT TEST IF ZEROES
1639  005246  010237  005300                    MOV     R2,63$                    ;KEEP COPY IN R2
1640  005252  012503                            MOV     (R5)+,R3                  ;GET R3=FLAGS DURING TEST
1641  005254  012501                            MOV     (R5)+,R1                  ;GET R1=EXPD FPSHI/FEC AFTER
1642  005256  104417                            ZAPWFP                            ;INIT TO HFP, LEAVE WFP ENABLED
1643  005260  170127  000020                    LDFPS   #000020                   ;WFP: FER=0, FID=0, FMM=1
1644  005264  010300                            MOV     R3,R0                     ;GET FLAG(5) FOR TEST
1645  005266  076600  000344                    MED     ,WFLAG                    ;SET/CLEAR FLAGS AS PER TEST
1646                                                                            ;
1647  005272  104406                            ERRPNT                            ;DONT CHANGE DATA IN ERROR LOOP
1648                                            ;--------ERROR-LOOP-ENTERS-HERE--------
1649                                                                            ;
1650  005274  104412  005302                    SETFP   ,23$                      ;SETUP FP TRAP OK, ESCAPE ADDR
1651                                                                            ;
1652  005300  000240                  63$:      NOP                               ;TEST INSTR GOES HERE
1653  005302                          23$:
1654  005302  105737  002645                    TSTB    LPTITE                    ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
1655  005306  001374                            BNE     63$                       ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
1656                                                                            ;
1657  005310  005000                            CLR     R0                        ;ZAP FLAGS=0
1658  005312  076600  000344                    MED     ,WFLAG                    ;DISABLE HFP
1659  005316  104413                            CLRFP                             ;FP TRAP NOW AN ERROR
1660                                                                            ;
1661  005320  076600  000036                    MED     ,RFEC                     ;GET R0=FPSHI/FEC
1662  005324  020001                            CMP     R0,R1                     ;WAS THIS CODE EXPECTED ?
1663  005326  001742                            BEQ     1$                        ;BR IF OK
1664  005330  104023                            ERROR   23                        ;NO - SIGNAL AN ERROR
1665                                            ;"HFP PREP0/1/2, UBRK, SRVC, */+ ENABL ERROR"
1666                                            ;       R-FPSHI/FEC = RCVD FPS(15:08)/FEC(03:00) IN R0
1667                                            ;       E-FPSHI/FEC = EXPD FPS(15:08)/FEC(03:00) IN R1
1668                                            ;       FLG(5) = RECEIVED BM FLAGS(08:00) IN R3(15:08).
1669                                            ;                            FLAG(5) IN BIT(12)
1670                                            ;       -FIR-- = COPY OF HFP FIR(15.00) IN R2
1671                                                                            ;
1672                                            ;GO ON TO NEXT TEST VALUE
1673  005332  000740                            BR      1$                        ;NEXT
1674                                    .
1675                                    ;////////////////////////////////////////////////////////////////////
1676                                    ;
```

```
1677                                    ;       DATA TABLE FOR ABOVE TEST:
1678                                    ;
1679                                    ;               --IR--  FLAG<5> FPSHI/FEC COMMENTS
1680
1681  005334  170000  010000  100016  40$:    .WORD   170000, 010000, 100016  ; FP*ENABLED, UBRK      [CFCC]
1682
1683  005342  074000  010000  000377          .WORD   074000, 010000, 000377  ;-FP*ENABLED, -UBRK     [XOR  R0,R0]
1684
1685  005350  170000  010000  100016          .WORD   170000, 010000, 100016  ; FP*ENABLED, UBRK      [CFCC]
1686
1687  005356  130000  010000  000377          .WORD   130000, 010000, 000377  ;-FP*ENABLED, -UBRK     [BITB R0,R0]
1688
1689  005364  170000  010000  100016          .WORD   170000, 010000, 100016  ; FP*ENABLED, UBRK      [CFCC]
1690
1691  005372  150000  010000  000377          .WORD   150000, 010000, 000377  ;-FP*ENABLED, -UBRK     [BISB R0,R0]
1692
1693  005400  170000  010000  100016          .WORD   170000, 010000, 100016  ; FP*ENABLED, UBRK      [CFCC]
1694
1695  005406  160000  010000  000377          .WORD   160000, 010000, 000377  ;-FP*ENABLED, -UBRK     [SUB  R0,R0]
1696
1697  005414  170000  010000  100016          .WORD   170000, 010000, 100016  ; FP*ENABLED, UBRK      [CFCC]
1698
1699  005422  170000  000000  000377          .WORD   170000, 000000, 000377  ;FP*-ENABLED, -UBRK     [CFCC]
1700
1701  005430  000000                          .WORD   0                       ;TERMINATOR
1702
1703
1704
```

```
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756  005432  000004                    TST11:  SCOPE
1757
1758  005434  012737  000024  001342            MOV     #20.,$TIMES             ;DO 20. ITERATIONS OF THIS TEST
1759  005442  012737  000003  003000            MOV     #3,DWICNT               ;SETUP FOR 3. CLOCK TICKS FOR A MATCH
1760  005450  012704  000377                    MOV     #377,R4                 ;LOOP FOR ADDRESS (377)-(000)
```

```
;;*****************************************************************
;*TEST 11       IFORK(LEFT), -[(ADD+SUB)*MO] FP INSTR DECODE

;       THIS TEST VERIFIES THE HFP/IFORK INSTRUCTION DECODE LOGIC
;       FOR THE "LEFT" BRANCH OF THE TREE; IE, THE:
;               -[ (ADD+SUB) * MODE-0 ]
;       CLASS OF INSTRUCTIONS.  THIS ENCOMPASSES VIRTUALLY THE
;       ENTIRE REPETOIRE OF OPCODES.
;
;       THE FP-INSTR-DECODE ROM ADDRESS IS GENERATED AS:
;
;               ADDR<7:0>H = FIRB<11:06>H * MO*R(6+7)L * FPS-FD-H
;               [RANGES FROM (002)-(377)]
;
;       THE RESULTANT IFORK MICROADDRESS DECODE VALUE IS:
;
;               UBRK<8:0>H = "0010" * FPDECODE<3:0>H * MO-H
;               [RANGES FROM (100)-(137)]
;
;       ----------
;       REGISTER/LOCATION USE:
;
;       $FPS    -FPS, BEFORE HFP UBREAK
;
;       R0      -(TEMP), "FEA"
;       R1      -(TEMP), "FPSHI*FEC"
;       R2      -MODE/REG PTR, FOR -[MODE0*R(6+7)] VALUE
;       R3      -UBRK/HFP EXPECTED IFORK MICROADDRESS, (100)-(137)
;       R4      -HFP FP-INSTR DECODE ROM ADDRESS, (002)-(377)
;       R5      -COPY OF FP-INSTR EXECUTED
;
;       ----------
;       MODULE/ERROR INFO:
;
;       FNUA/K8
;         FPINMUX, FIR-A/B, FP-INSTR/INSTRCVD/CLK-FIRA-B,
;         FNUA-GENERATE/JREG-LOGIC, HFP-MICROBREAK,
;         IFORK-MUX, FP-INSTR-DECODE-ROM, ADD+SUB/MODE-0-LOGIC,
;         CROM/LATCHES
;
;       FEXP/K9
;         JAMUPP, HFP/BM-INTERFACE, FBRAN<2:0>, FBRAN-DECODE,
;         CROM/LATCHES
;
;       FMUL/K10
;         [ESSENTIALLY NONE]
;
;       FALU/K11
;         [ESSENTIALLY NONE]
;
;*****************************************************************
```

```
1761  005454  012702  005712                  MOV     #CODEJ,R2               ;PRIME THE MODE/REG PTR
1762
1763                                          ;*LOOP ON ROM ADDRESS ENTERS HERE
1764  005460  005237  002640          1$:     INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
1765  005464  104411                          CLRDW                           ;SETUP FOR PROC HANG ERROR
1766  005466  104416                          ZAPHFP                          ;INIT TO HFP,
1767                                                                          ;FPS=(040000), FEC=(377), FEA=FPA=(177777)
1768
1769                                          ;CALC R5=FP-INSTR CODE NECESSARY TO GENERATE THIS ROM ADDRESS
1770  005470  010405                          MOV     R4,R5                   ;
1771  005472  072527  000004                  ASH     #4,R5                   ;LEFT-4, IR<11:06>
1772  005476  042705  000070                  BIC     #000070,R5              ;SET MODE-0
1773  005502  052705  170006                  BIS     #170006,R5              ; AND DR6, FP-INSTR
1774  005506  032704  000002                  BIT     #BIT01,R4               ;WANT MODE-0*R(6+7) ?
1775  005512  001406                          BEQ     10$                     ;BR IF YES
1776  005514  152205                          BISB    (R2)+,R5                ;SETUP MODE/REG OF:
1777  005516  142205                          BICB    (R2)+,R5                ;  (00), (16), (26), (46)
1778  005520  105712                          TSTB    (R2)                    ;
1779  005522  003002                          BGT     ...                     ;
1780  005524  012702  005712                  MOV     #CODEJ,R2               ;RESET PTR AT END OF TABLE
1781  005530  010537  005622          10$:    MOV     R5,63$                  ;SET THE INSTR IN MEMORY
1782
1783                                          ;CALC R3=HFP UBRK ADDRESS EXPECTED OUT OF IFORK
1784  005534  004737  005724                  JSR     PC,GETBRK               ;FROM THE SUBR, BELOW
1785  005540  103456                          BCS     23$                     ;MUST SKIP THIS ROM-ADDR IF SET
1786  005542  170003                          LDUB                            ;LOAD INTO HFP UBRK
1787  005544  104416                          ZAPHFP                          ;CLEAR OUT ANY LDUB EFFECTS
1788
1789                                          ;CALC $FPS=FPS VALUE NECESSARY (FD, SPECIFICALLY)
1790  005546  012700  000040                  MOV     #000040,R0              ;GET $FPS=FPS WITH:
1791  005552  010401                          MOV     R4,R1                   ;  FER=0, FID=0, FMM=1,
1792  005554  006001                          ROR     R1                      ;  AND FD=ROMADR<0>
1793  005556  106000                          RORB    R0                      ;
1794  005560  010037  002610                  MOV     R0,$FPS                 ;SAVE IN MEMORY
1795  005564  170100                          LDFPS   R0                      ;INTO FPS REGISTER
1796
1797  005566  104410  005624                  SETDW   ,21$                    ;SETUP PROC HUNG ESCAPE ENABLE
1798
1799  005572  010600                          MOV     SP,R0                   ;COPY KSP -> R0
1800  005574  162700  000040                  SUB     #40,R0                  ;LEAVE SOME SPACE
1801                                                                          ;
1802  005600  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
1803                                          ;--------ERROR-LOOP-ENTERS-HERE--------
1804                                                                          ;
1805  005604  052737  140300  177776          BIS     #BIT15+BIT14+PR6,PS     ;SET USER MODE, FOR R6; =PR6 FOR CLOCK DISABLED
1806  005610  010006                          MOV     R0,SP                   ;INIT USP <- R0
1807  005612  105037  177776                  CLRB    PS                      ;PR0 FOR LINE CLOCK ENABLED
1808  005616  104412  005624                  SETFP   ,21$                    ;ENABLE FP ESCAPE
1809                                                                          ;
1810  005622  000240                  63$:    NOP                             ;FP-INSTR GOES HERE
1811                                          ;RETURN HERE AFTER FP TRAP
1812  005624                          21$:
1813  005624  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
1814  005630  001374                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
1815                                                                          ;
1816  005632  042737  140000  177776          BIC     #BIT15+BIT14,PS         ;BACK TO KERNAL SP
```

```
1817  005640  104413                  CLRFP                           ;ZAP FP ESCAPE ENABLE
1818
1819  005642  076600  000036          MED     ,RFEC                   ;
1820  005646  010001                  MOV     R0,R1                   ;GET R1="FPSHI*FEC"
1821  005650  076600  000076          MED     ,RFEA                   ;GET R0="FEA"
1822
1823  005654  020127  100016          CMP     R1,#100016              ;FPSHI*FEC:  FER=1 & FEC=(16) ?
1824  005660  001406                  BEQ     23$                     ;BR IF YES
1825  005662  020127  000377          CMP     R1,#000377              ;FPSHI*FEC:  WERE THEY UNMODIFIED ?
1826  005666  001002                  BNE     22$                     ;BR IF SOME OTHER ERROR
1827
1828  005670  104021                  ERROR   21                      ;IFORK DECODE ERROR
1829                                  ;"HFP/IFORK/-[(ADD+SUB)*MO]; BAD IFORK DECODE
1830                                  ;       ROMADR = EXPD ROM ADDRESS TO HFP FP DECODE ROM
1831                                  ;       --FIR- = EXPD CONTENTS OF HFP FIR<15:00>
1832                                  ;       FPUBRK = EXPD HFP IFORK TARGET MICROADDRESS [HFP UBRK REG]
1833                                  ;       PRVFPS = FPS LOADED BEFORE HFP STARTED ($FPS)
1834                                  ;       FPSFEC = RCVD FPSHI<15:08>/FEC<03:00> AFTER HFP STARTED
1835                                  ;       -FEA-- = RCVD FEA AFTER HFP STARTED
1836  005672  000401                  BR      23$                     ;
1837
1838  005674  104022          22$:    ERROR   22                      ;UNEXPECTED FEC/FEA VALUE
1839                                  ;"HFP/IFORK/-[(ADD+SUB)*MO]; UNEXPECTED FEC/FEA"
1840                                  ;       ROMADR = EXPD ROM ADDRESS TO HFP FP DECODE ROM
1841                                  ;       --FIR- = EXPD CONTENTS OF HFP FIR<15:00>
1842                                  ;       FPUBRK = EXPD HFP IFORK TARGET MICROADDRESS [HFP UBRK REG]
1843                                  ;       PRVFPS = FPS LOADED BEFORE HFP STARTED ($FPS)
1844                                  ;       FPSFEC = RCVD FPSHI<15:08>/FEC<03:00> AFTER HFP STARTED
1845                                  ;       -FEA-- = RCVD FEA AFTER HFP STARTED
1846
1847  005676  005304          23$:    DEC     R4                      ;NEXT DECODE ROM ADDR
1848  005700  020427  000004          CMP     R4,#004                 ;TEST FOR LOWEST ROM ADDR
1849  005704  002265                  BGE     1$                      ;LOOP IF MORE
1850                                  ;
1851  005706  104416                  ZAPHFP                          ;RESET HFP PRIOR TO EXIT
1852  005710  000542                  BR      TST12           ;;      ;NEXT TEST WHEN DONE
1853
1854                          ;///////////////////////////////////////////////////////////////////////
1855                          ;
1856                          ;       THIS LITTLE TABLE IS USED IN CONJUNCTION WITH THE FP-INSTR
1857                          ;       GENERATING CODE ABOVE.
1858                          ;
1859  005712      000     077 CODEJ:  .BYTE   00,77   ;M0/R0  -  AC0 OR #0
1860  005714      016     061         .BYTE   16,61   ;M1/R6  -  (SP)
1861  005716      026     051         .BYTE   26,51   ;M2/R6  -  (SP)+
1862  005720      046     031         .BYTE   46,31   ;M4/R6  -  -(SP)
1863  005722      000     000         .BYTE   00,00   ;[RESET]
1864                          ;
1865                          ;
1866                          ;
1867                          ;///////////////////////////////////////////////////////////////////////
1868                          ;
1869                          ;       THIS SUBROUTINE IS USED ABOVE TO CALCULATE, GIVEN
1870                          ;         R4=DESIRED FP-INSTR DECODE ROM ADDRESS; 000-377
1871                          ;         R5=THE FP-INSTR ASSEMBLED
1872                          ;       THEN:
```

```
1873                                    ;       R3=THE IFORK EXPECTED MICROADDRESS, (100)-(137)
1874                                    ;       C-BIT=SET IF IR=[(ADD+SUB)*MODE0]
1875                                    ;
1876  005724  010403                   GETBRK: MOV     R4,R3                   ;COPY ROM ADDR
1877  005726  006203                           ASR     R3                      ;R3=OFFSET (000-177), C-BIT=LO-4/HI-4
1878  005730  116303  006016                   MOVB    CODEI(R3),R3            ;GET DATA
1879  005734  103002                           BCC     11$                     ;GET LOW 4
1880  005736  072327  177774                   ASH     #-4,R3                  ;GET HIGH 4
1881  005742  042703  177760           11$:    BIC     #^C17,R3                ;ZAP H.O.B.
1882
1883  005746  000241                   40$:    CLC                             ;SETUP UBRK<0>=MODE-0-H
1884  005750  032705  000070                   BIT     #BIT5+BIT4+BIT3,R5      ;
1885  005754  001001                           BNE     41$                     ;
1886  005756  000261                           SEC                             ;
1887  005760  006103                   41$:    ROL     R3                      ;
1888
1889  005762  052703  000100                   BIS     #100,R3                 ;BASE ADDR = (100)
1890
1891  005766  020327  000101                   CMP     R3,#101                 ;ADD/SUB/CFCC/SETX ?
1892  005772  003007                           BGT     42$                     ;BR IF NOT
1893  005774  032705  002000                   BIT     #BIT10,R5               ;CFCC/SETX OR ADD/SUB ?
1894  006000  001404                           BEQ     42$                     ;BR IF CFCC/SETX
1895  006002  006203                           ASR     R3                      ;MO-H="1" IN BIT00 ?
1896  006004  103403                           BCS     43$                     ;BR IF YES, WITH C-BIT=1
1897  006006  012703  000110                   MOV     #110,R3                 ;NO, (ADD+SUB)*-MO -> (110)
1898
1899  006012  000241                   42$:    CLC                             ;OK EXIT
1900  006014  000207                   43$:    RTS     PC                      ;AND RETURN
1901
1902                                    ;
1903                                    ;       THE TABLE BELOW REPRESENTS THE CONTENTS, MORE OR LESS, OF THE
1904                                    ;       HFP INSTRUCTION DECODE ROM.
1905                                    ;
1906                                    ;                       --DECODE DATA---        FIRB    ROM ADDR        SYMBOLIC FP
1907                                    ;                       DDDDCCCCBBBBAAAA        <11:6>  DDDD/AAAA       INSTRUCTION
1908  006016                           CODEI:  ;               ----------------        ------  ---------       -----------
1909  006016  000000                           .WORD   ^B0000000000000000      ;(00)   (003)-(000)     CFCC/SET-X, (001)-(000)
                                                                                                                    NOT USED
1910  006020  063146                           .WORD   ^B0110011001100110      ;(01)   (007)-(004)     LDFPS
1911  006022  063146                           .WORD   ^B0110011001100110      ;(02)   (013)-(010)     STFPS
1912  006024  063146                           .WORD   ^B0110011001100110      ;(03)   (017)-(014)     STST
1913  006026  010504                           .WORD   ^B0001000101000100      ;(04)   (023)-(020)     C/T/A/N-X
1914  006030  010504                           .WORD   ^B0001000101000100      ;(05)   (027)-(024)     C/T/A/N-X
1915  006032  010504                           .WORD   ^B0001000101000100      ;(06)   (033)-(030)     C/T/A/N-X
1916  006034  010504                           .WORD   ^B0001000101000100      ;(07)   (037)-(034)     C/T/A/N-X
1917  006036  031104                           .WORD   ^B0011001001000100      ;(10)   (043)-(040)     MUL-X
1918  006040  031104                           .WORD   ^B0011001001000100      ;(11)   (047)-(044)     MUL-X
1919  006042  031104                           .WORD   ^B0011001001000100      ;(12)   (053)-(050)     MUL-X
1920  006044  031104                           .WORD   ^B0011001001000100      ;(13)   (057)-(054)     MUL-X
1921  006046  031104                           .WORD   ^B0011001001000100      ;(14)   (063)-(060)     MOD-X
1922  006050  031104                           .WORD   ^B0011001001000100      ;(15)   (067)-(064)     MOD-X
1923  006052  031104                           .WORD   ^B0011001001000100      ;(16)   (073)-(070)     MOD-X
1924  006054  031104                           .WORD   ^B0011001001000100      ;(17)   (077)-(074)     MOD-X
1925  006056  000000                           .WORD   ^B0000000000000000      ;(20)   (103)-(100)     ADD-X, ALMOST
1926  006060  000000                           .WORD   ^B0000000000000000      ;(21)   (107)-(104)     ADD-X, ALMOST
1927  006062  000000                           .WORD   ^B0000000000000000      ;(22)   (113)-(110)     ADD-X, ALMOST
```

```
1928  006064  000000          .WORD   ↑B0000000000000000      ;(23)   (117)-(114)     ADD-X, ALMOST
1929  006066  052504          .WORD   ↑B0101010101000100      ;(24)   (123)-(120)     LD-X
1930  006070  052504          .WORD   ↑B0101010101000100      ;(25)   (127)-(124)     LD-X
1931  006072  052504          .WORD   ↑B0101010101000100      ;(26)   (133)-(130)     LD-X
1932  006074  052504          .WORD   ↑B0101010101000100      ;(27)   (137)-(134)     LD-X
1933  006076  000000          .WORD   ↑B0000000000000000      ;(30)   (143)-(140)     SUB-X, ALMOST
1934  006100  000000          .WORD   ↑B0000000000000000      ;(31)   (147)-(144)     SUB-X, ALMOST
1935  006102  000000          .WORD   ↑B0000000000000000      ;(32)   (153)-(150)     SUB-X, ALMOST
1936  006104  000000          .WORD   ↑B0000000000000000      ;(33)   (157)-(154)     SUB-X, ALMOST
1937  006106  073504          .WORD   ↑B0111011101000100      ;(34)   (163)-(160)     CMP-X
1938  006110  073504          .WORD   ↑B0111011101000100      ;(35)   (167)-(164)     CMP-X
1939  006112  073504          .WORD   ↑B0111011101000100      ;(36)   (173)-(170)     CMP-X
1940  006114  073504          .WORD   ↑B0111011101000100      ;(37)   (177)-(174)     CMP-X
1941  006116  104104          .WORD   ↑B1000100001000100      ;(40)   (203)-(200)     ST-X
1942  006120  104104          .WORD   ↑B1000100001000100      ;(41)   (207)-(204)     ST-X
1943  006122  104104          .WORD   ↑B1000100001000100      ;(42)   (213)-(210)     ST-X
1944  006124  104104          .WORD   ↑B1000100001000100      ;(43)   (217)-(214)     ST-X
1945  006126  114504          .WORD   ↑B1001100101000100      ;(44)   (223)-(220)     DIV-X
1946  006130  114504          .WORD   ↑B1001100101000100      ;(45)   (227)-(224)     DIV-X
1947  006132  114504          .WORD   ↑B1001100101000100      ;(46)   (233)-(230)     DIV-X
1948  006134  114504          .WORD   ↑B1001100101000100      ;(47)   (237)-(234)     DIV-X
1949  006136  125252          .WORD   ↑B1010101010101010      ;(50)   (243)-(240)     STEXP
1950  006140  125252          .WORD   ↑B1010101010101010      ;(51)   (247)-(244)     STEXP
1951  006142  125252          .WORD   ↑B1010101010101010      ;(52)   (253)-(250)     STEXP
1952  006144  125252          .WORD   ↑B1010101010101010      ;(53)   (257)-(254)     STEXP
1953  006146  135673          .WORD   ↑B1011110111011011      ;(54)   (263)-(260)     STC-T
1954  006150  135673          .WORD   ↑B1011110111011011      ;(55)   (267)-(264)     STC-T
1955  006152  135673          .WORD   ↑B1011110111011011      ;(56)   (273)-(270)     STC-T
1956  006154  135673          .WORD   ↑B1011110111011011      ;(57)   (277)-(274)     STC-T
1957  006156  146104          .WORD   ↑B1100110001000100      ;(60)   (303)-(300)     STC-P
1958  006160  146104          .WORD   ↑B1100110001000100      ;(61)   (307)-(304)     STC-P
1959  006162  146104          .WORD   ↑B1100110001000100      ;(62)   (313)-(310)     STC-P
1960  006164  146104          .WORD   ↑B1100110001000100      ;(63)   (317)-(314)     STC-P
1961  006166  156735          .WORD   ↑B1101110111011101      ;(64)   (323)-(320)     LDEXP
1962  006170  156735          .WORD   ↑B1101110111011101      ;(65)   (327)-(324)     LDEXP
1963  006172  156735          .WORD   ↑B1101110111011101      ;(66)   (333)-(330)     LDEXP
1964  006174  156735          .WORD   ↑B1101110111011101      ;(67)   (337)-(334)     LDEXP
1965  006176  167356          .WORD   ↑B1111011101101110      ;(70)   (343)-(340)     LDC-T
1966  006200  167356          .WORD   ↑B1111011101101110      ;(71)   (347)-(344)     LDC-T
1967  006202  167356          .WORD   ↑B1111011101101110      ;(72)   (353)-(350)     LDC-T
1968  006204  167356          .WORD   ↑B1111011101101110      ;(73)   (357)-(354)     LDC-T
1969  006206  177504          .WORD   ↑B1111111101000100      ;(74)   (363)-(360)     LDC-P
1970  006210  177504          .WORD   ↑B1111111101000100      ;(75)   (367)-(364)     LDC-P
1971  006212  177504          .WORD   ↑B1111111101000100      ;(76)   (373)-(370)     LDC-P
1972  006214  177504          .WORD   ↑B1111111101000100      ;(77)   (377)-(374)     LDC-P
1973
1974
1975
1976                  ;*****************************************************************
1977                  ;*TEST 12       FIRB IMMEDIATE-H ADDRESS MODE DECODE
1978                  ;
1979                  ;       THIS TEST RUNS THRU THE SEQUENCE OF SF<5:0> MODE/REGISTER VALUES
1980                  ;       TO CHECK THAT THE "IMMEDIATE-H" MODE DECODE OF THE LDF/LDD INSTRUCTION
1981                  ;       IS PERFORMED CORRECTLY.
1982                  ;
1983                  ;       MICROWORD "LOAD.02" PERFORMS THE "BUTR(IMMEDIATE)", TC TARGETS:
```

```
1984
1985                    ;       LOAD.04 (231) IF IMMEDIATE-H = L
1986                    ;
1987                    ;       LOADIMM (233) IF IMMEDIATE-H = H
1988                    ;
1989                    ;       ----------
1990                    ;       REGISTER/LOCATION USE:
1991                    ;
1992                    ;       R0      -RCV'D FPSHI#FEC AFTER INSTR EXEC
1993                    ;       R1      -EXP'D HFP UBRK ADDRESS
1994                    ;       R2      -COPY OF FP "LDF" INSTR EXEC
1995                    ;       R3      -LDUB, PTR TO (0,0,0,0)
1996                    ;       R4      -DATA TABLE PTR
1997                    ;       R5      -PTR TO (0,0,0,0)
1998                    ;
1999                    ;       ----------
2000                    ;       MODULE/ERROR INFO:
2001                    ;
2002                    ;       FNUA/K8
2003                    ;         FPINMUX, FIR-A/B, FP-INSTR/INSTRCVD/CLK-FIRA-B,
2004                    ;         FNUA-GENERATE/JREG-LOGIC, HFP-MICROBREAK,
2005                    ;         IMMEDIATE-H-DECODE-LOGIC, CROM/LATCHES
2006                    ;
2007                    ;       FEXP/K9
2008                    ;         JAMUPP, HFP/BM-INTERFACE, FBRAN<2:0>, FBRAN-DECODE,
2009                    ;         CROM/LATCHES
2010                    ;
2011                    ;       FMUL/K10
2012                    ;         [ESSENTIALLY NONE]
2013                    ;
2014                    ;       FALU/K11
2015                    ;         [ESSENTIALLY NONE]
2016                    ;
2017                    ;.*************************************************************
2018  006216  000004    TST12:  SCOPE
2019                    ;
2020  006220  012704  006336        MOV     #40$,R4                 ;PTR TO DATA TABLE
2021  006224  012705  003060        MOV     #FPZERO,R5              ;USED AS PTR TO (0,0,0,0)
2022  006230  012737  000036  001342        MOV     #30.,$TIMES             ;30. ITER OF THIS TEST
2023  006236  012737  000003  003000        MOV     #3,DWICNT               ;3 HUNGS TO AN ESCAPE
2024                    ;                                               ;
2025                    ;*DATA LOOP ENTERS HERE*
2026  006244  005237  002640    10$:  INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
2027  006250  104416        ZAPHFP                                  ;INIT TO HFP, FID=1/FMM=0
2028  006252  012402        MOV     (R4)+,R2                        ;GET MODE/REG FP 'LDF' INSTR
2029  006254  001451        BEQ     TST13           ;;              ;NEXT TEST IF ALL ZERO
2030  006256  010237  006304        MOV     R2,63$                  ;STORE IN MEMORY
2031  006262  012403        MOV     (R4)+,R3                        ;GET EXPEC'D HFP UBRK ADDR
2032  006264  010301        MOV     R3,R1                           ;SAVE
2033  006266  170003        LDUB                                    ;GIVE IT TO HFP
2034  006270  010503        MOV     R5,R3                           ;SET R3 TO PTR TO (0,0,0,0)
2035  006272  170127  040020        LDFPS   #040020                 ;SET FID=1/FMM=1 TO EN HFP UBRK
2036  006276  104410  006306        SETDW   .11$                    ;LINE CLOCK ESCAPES TO HERE
2037                    ;                                               ;
2038  006302  104406        ERRPNT                                  ;DONT CHANGE DATA IN ERROR LOOP
2039                    ;--------ERROR-LOOP-ENTERS-HERE--------
```

```
2040                                                                    ;
2041                                                                    ;
2042  006304  170000                63$:    CFCC                      ;FP TEST INSTR HERE
2043  006306  000240                11$:    NOP                       ;BM FOLLOW UP
2044  006310  105737  002645              TSTB    LPTITE              ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2045  006314  001373                      BNE     63$                 ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2046                                                                    ;
2047  006316  104411                      CLRDW                       ;CLOCK ESCAPE CLEARED OUT
2048  006320  076600  000036              MED     ,RFEC               ;GET FPSHI*FEC AFTER
2049  006324  020027  140016              CMP     R0,#140016          ;DID HFP UBRK? (FER=1/FEC=16)
2050  006330  001745                      BEQ     10$                 ;YES - NEXT DATA LOOP
2051  006332  104072                      ERROR   72                  ;BAD IMM-H DECODE
2052                                      ;"FIRB IMMED-H MODE DECODE ERR"
2053                                      ;       FPINST = COPY OF FP.INSTRUCTION/SF.MODE UNDER TEST
2054                                      ;       E-UBRK = EXP'D TARGET ADDRESS, (231/233)
2055                                      ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER EXEC, EXP'D (140016)
2056  006334  000743                      BR      10$                 ;NEXT LOOP
2057
2058                                    ;////////////////////////////////////////////////////////////////////////
2059                                    ;
2060                                    ;       DATA TABLE FOR ABOVE TEST:
2061                                    ;
2062                                    ;       LDX-FP  HFP     ADDR
2063                                    ;       INSTR   UBREAK  MODE
2064
2065  006336  172413  000231   40$:     172413, 231     ;M1-R3  -IMM    (R3)
2066
2067  006342  172415  000231            172415, 231     ;M1-R5  -IMM    (R5)
2068
2069  006346  172416  000231            172416, 231     ;M1-R6  -IMM    (R6)
2070
2071  006352  172417  000233            172417, 233     ;M1-R7  +IMM+   (PC)
2072
2073  006356  172437  000231            172437, 231     ;M3-R7  -IMM    @(PC)+
2074
2075  006362  172427  000233            172427, 233     ;M2-R7  +IMM+   (PC)+
2076
2077  006366  172467  000231            172467, 231     ;M6-R7  -IMM    X(PC)
2078
2079  006372  172447  000233            172447, 233     ;M4-R7  +IMM+   -(PC)
2080
2081  006376  000000                    0       ;<DONE>
2082
2083
2084
```

```
                        ;;*****************************************************************
                        ;*TEST 13       UFLOW - FPINIT, F-MODE
                        ;
                        ;       THIS SEQUENCE OF CODE "FOLLOWS" THE FP11-E THRU ITS INITIALIZATION CODE
                        ;       TO CHECK THAT EACH MICROWORD IS EXECUTED IN ORDER.  THE MICROBREAK
                        ;       FEATURE IS USED FOR TRACKING.
                        ;
                        ;       AN INIT IS GIVEN TO THE FP11-E, WHICH WAS PREVIOUSLY IN "F-MODE".
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         NEXT-MICROADDRESS-GATING-LOGIC, NUA-ROMS/LATCHES, HFP-UBRK
                        ;         CROM/LATCHES, FP-EMIT-F, FSPAD-WRITE
                        ;
                        ;       FEXP/K9
                        ;         HFP/BM-INTERFACE-LOGIC, FPBRAN<2:0>-UBF-DECODE,
                        ;         HFP-SRVC-REQ/GRANT-LOGIC, JAMUPP-LOGIC, CROM/LATCHES
                        ;
                        ;       FMUL/K10
                        ;         MNET-ALU/SELECT-A-SIDE
                        ;
                        ;       FALU/K11
                        ;         FSPAD/AR/FPOUTMUX-DATAPATH
                        ;
                        ;*****************************************************************
006400  000004          TST13:  SCOPE
                                                                ;
006402  012705  006534          MOV     #40$,R5                 ;UBRK ADDRESS TABLE PTR
                                                                ;
                                ;*DATA LOOP ENTERS HERE*
006406  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
006412  012503                  MOV     (R5)+,R3                ;GET NEXT UBRK ADDRESS, FROM TABLE
006414  104416                  ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
006416  170003                  LDUB                            ;UBRK(R3) -> HFP
                                                                ;
006420  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                                ;
006422  104417                  ZAPWFP                          ;FLIM SIDE EFFECTS, DISABL HFP
006424  170127  000037          LDFPS   #000037                 ;WFP EXEC, FER=0/FID=0/FD=0/FMM=1/FCC=1111
                                                                ;
006430  012700  010000          MOV     #010000,R0              ;FLAG<5:4>="10" FOR HFP-EN*FPCNST-OK
006434  076600  000344          MED     ,WFLAG                  ;
                                                                ;
006440  104410  006460          SETDW   ,29$                    ;SETUP PROC-HUNG ESC VIA CLOCK
006444  104412  006460          SETFP   ,29$                    ;SETUP FP-TRAP ESCAPE
006450                  63$:                                    ;
006450  012700  000001          MOV     #000001,R0              ;SELECT HFP INIT FROM BM MED SUBROUTINE
006454  076600  000352  20$:     MED     ,WINIT                  ;DO ONLY THE INIT OF HFP
                                                                ;
006460                  29$:                                    ;
006460  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
006464  001371                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                                ;
```

```
2141  006466  104413                    CLRFP                           ;MAKE FP/PROC-HUNG TRAPS
2142  006470  104411                    CLRDW                           ;  INTO ERRORS NOW
2143                                                                      ;
2144  006472  020327  000666            CMP     R3,#666                 ;END OF UBRK TABLE ?
2145  006476  001414                    BEQ     30$                     ;YES - CHECK RESULT OF EXEC
2146                                                                      ;
2147  006500  076600  000036            MED     ,RFEC                   ;NO, GET RO=FPSHI#FEC
2148                                                                      ;
2149  006504  020027  100016            CMP     R0,#100016              ;DID HFP UBRK ?  (FER=1/FEC=16)
2150  006510  001736                    BEQ     10$                     ;YES - ON TO NEXT UADDR
2151                                                                      ;
2152  006512  020027  000377            CMP     R0,#000377              ;DIDN'T - CHECK FEC CODE ...
2153  006516  001402                    BEQ     25$                     ;
2154  006520  104044                    ERROR   44                      ;UNRECOGNIZABLE FEC-CODE
2155                                    ;"FPINIT FLOW, UNEXP'D FPSHI#FEC ERR"
2156                                    ;       E-UBRK = EXP'D MICROADDRESS FOR FP11-E
2157                                    ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (100016) OR (000377)
2158                                                                      ;
2159  006522  000731                    BR      10$                     ;  MORE
2160  006524  104045            25$:    ERROR   45                      ;HFP DIDN'T UBRK AT ADDRESS
2161                                    ;"FPINIT FLOW, HFP DIDN'T UBREAK ERR"
2162                                    ;       E-UBRK = EXP'D MICROADDRESS FOR FP11-E
2163                                    ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (100016) OR (000377)
2164                                                                      ;
2165  006526  000727                    BR      10$                     ;  MORE
2166                                                                      ;
2167  006530                    30$:    ;*END OF UBRK TABLE - CHECK RESULT OF EXEC*
2168  006530  104416            39$:    ZAPHFP                          ;INIT HFP, SET FID=1/FMM=0
2169  006532  000421                    BR      TST14           ;;      ;ON TO NEXT TEST
2170                            ;
2171                            ;
2172                            ;-------MICROFLOW TABLE FOR ABOVE TEST-------
2173                            ;       UADDR   ;LABEL          UBRANCH-CONDITION
2174  006534            40$:;   -----   ----------      ----------------
2175  006534  000522                    522     ;FPINIT         BUT(FD)="0"
2176  006536  000500                    500     ;FPINIT.01
2177  006540  000562                    562     ;FPINIT.03      BUT(FD)="0"
2178  006542  000574                    574     ;FPINIT.04
2179  006544  000563                    563     ;FPINIT.06
2180  006546  000564                    564     ;FPINIT.07
2181  006550  000565                    565     ;FPINIT.08
2182  006552  000566                    566     ;FPINIT.09
2183  006554  000567                    567     ;FPINIT.10
2184  006556  000570                    570     ;FPINIT.11
2185  006560  000571                    571     ;FPINIT.12
2186  006562  000572                    572     ;FPINIT.13
2187  006564  000573                    573     ;FPINIT.14
2188  006566  000576                    576     ;FPINIT.16
2189  006570  000577                    577     ;FPINIT.18
2190  006572  000600                    600     ;FPINIT.20
2191  006574  000666                    666     ;<END-OF-TABLE>
2192
2193
2194
2195                            ;;*****************************************************************
2196                            ;*TEST 14        UFLOW - FPINIT, D-MODE
```

```
2197
2198                    ;       THIS SEQUENCE OF CODE "FOLLOWS" THE FP11-E THRU ITS INITIALIZATION CODE
2199                    ;       TO CHECK THAT EACH MICROWORD IS EXECUTED IN ORDER.  THE MICROBREAK
2200                    ;       FEATURE IS USED FOR TRACKING.
2201                    ;
2202                    ;       AN INIT IS GIVEN TO THE FP11-E, WHICH WAS PREVIOUSLY IN "D-MODE"
2203                    ;
2204                    ;       ----------
2205                    ;       MODULE/ERROP INFO:
2206                    ;
2207                    ;       FNUA/K8
2208                    ;         NEXT-MICROADDRESS-GATING-LOGIC, NUA-ROMS/LATCHES, HFP-UBRK
2209                    ;         CROM/LATCHES, FP-EMIT-F, FSPAD-WRITE
2210                    ;
2211                    ;       FEXP/K9
2212                    ;         HFP/BM-INTERFACE-LOGIC, FPBRAN<2:0>-UBF-DECODE,
2213                    ;         HFP-SRVC-REQ/GRANT-LOGIC, JAMUPP-LOGIC, CROM/LATCHES
2214                    ;
2215                    ;       FMUL/K10
2216                    ;         MNET-ALU/SELECT-A-SIDE
2217                    ;
2218                    ;       FALU/K11
2219                    ;         FSPAD/AR/FPOUTMUX-DATAPATH
2220                    ;
2221                    ;*****************************************************************
2222  006576  000004    TST14:  SCOPE
2223                    ;
2224  006600  012705  006732       MOV     #40$,R5                 ;UBRK ADDRESS TABLE PTR
2225                    ;
2226                            ;*DATA LOOP ENTERS HERE*
2227  006604  005237  002640 10$:  INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
2228  006610  012503            MOV     (R5)+,R3                ;GET NEXT UBRK ADDRESS, FROM TABLE
2229  006612  104416            ZAPHFP                          ;INIT TO HFP, LEAVE IT ENABLED
2230  006614  170003            LDUB                            ;UBRK(R3) -> HFP
2231                    ;
2232  006616  104406            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
2233                            ;--------ERROR-LOOP-ENTERS-HERE--------
2234                    ;
2235  006620  104417            ZAPWFP                          ;ELIM SIDE EFFECTS, DISABL HFP
2236  006622  170127  000237    LDFPS   #000237                 ;WFP EXEC, FER=0/FID=0/FD=1/FMM=1/FCC=1111
2237                    ;
2238  006626  012700  010000    MOV     #010000,R0              ;FLAG<5:4>="10" FOR HFP-EN*FPCNST-OK
2239  006632  076600  000344    MED     ,WFLAG                  ;
2240                    ;
2241  006636  104410  006656    SETDW   ,29$                    ;SETUP PROC-HUNG ESC VIA CLOCK
2242  006642  104412  006656    SETFP   ,29$                    ;SETUP FP-TRAP ESCAPE
2243  006646            63$:    ;
2244  006646  012700  000001    MOV     #000001,R0              ;SELECT HFP INIT FROM BM MED SUBROUTINE
2245  006652  076600  000352 20$:  MED     ,WINIT                  ;DO ONLY THE INIT OF HFP
2246                    ;
2247  006656            29$:    ;
2248  006656  105737  002645    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2249  006662  001371            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2250                    ;
2251  006664  104413            CLRFP                           ;MAKE FP/PROC-HUNG TRAPS
2252  006666  104411            CLRDW                           ;  INTO ERRORS NOW
```

```
2253
2254  006670  020327  000666          CMP     R3,#666                 ;END OF UBRK TABLE ?
2255  006674  001414                  BEQ     30$                     ;YES - CHECK RESULT OF EXEC
2256
2257  006676  076600  000036          MED     ,RFEC                   ;NO, GET R0=FPSHI#FEC
2258
2259  006702  020027  100016          CMP     R0,#100016              ;DID HFP UBRK ?  (FER=1/FEC=16)
2260  006706  001736                  BEQ     10$                     ;YES - ON TO NEXT UADDR
2261
2262  006710  020027  000377          CMP     R0,#000377              ;DIDN T - CHECK FEC CODE ...
2263  006714  001402                  BEQ     25$
2264  006716  104044                  ERROR   44                      ;UNRECOGNIZABLE FEC-CODE
2265                                  ;"FPINIT FLOW, UNEXP'D FPSHI#FEC ERR"
2266                                  ;       E-UBRK = EXP'D MICROADDRESS FOR FP11-E
2267                                  ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (100016) OR (000377)
2268
2269  006720  000731                  BR      10$                     ;  MORE
2270  006722  104045          25$:    ERROR   45                      ;HFP DIDN'T UBRK AT ADDRESS
2271                                  ;"FPINIT FLOW, HFP DIDN'T UBREAK ERR"
2272                                  ;       E-UBRK = EXP'D MICROADDRESS FOR FP11-E
2273                                  ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (100016) OR (000377)
2274
2275  006724  000727                  BR      10$                     ;  MORE
2276
2277  006726          30$:    ;*END OF UBRK TABLE - CHECK RESULT OF EXEC*
2278  006726  104416          39$:    ZAPHFP                          ;INIT HFP, SET FID=1/FMM=0
2279  006730  000407                  BR      TST15           ;;      ;ON TO NEXT TEST
2280
2281                                  ;
2282                                  ;-------MICROFLOW TABLE FOR ABOVE TEST-------
2283                                  ;       UADDR   ;LABEL          UBRANCH-CONDITION
2284  006732          40$:;           -----   ----------      ----------------
2285  006732  000522                  522     ;FPINIT         BUT(FD)="1"
2286  006734  000501                  501     ;FPINIT.02
2287  006736  000562                  562     ;FPINIT.03      BUT(FD)="1"
2288  006740  000575                  575     ;FPINIT.05
2289  006742  000563                  563     ;FPINIT.06
2290                                  ;...    (TRACKED IN PREV TEST)
2291  006744  000600                  600     ;FPINIT.20
2292  006746  000666                  666     ;<END-OF-TABLE>
2293
2294
2295
```

```
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;;*****************************************************************

                                ;;*****************************************************************
                                ;*TEST 15        ...ENABLE HFP MICROBREAK LOAD...
                                ;;*****************************************************************
006750  000004                  TST15:  SCOPE
006752  005037  001342                  CLR     $TIMES                  ;NO ITER. OF THIS "TEST"
006756  005037  001214                  CLR     $ERFLG                  ;OR ERROS EITHER

006762  104416                          ZAPHFP                          ;INIT HFP, SET FID=1, FMM=0
006764  112737  000377  002623          MOVB    #377,NOFPIE             ;ENABLE HFP-UBRK LOAD (VIA LDUB)
                                                                        ;  TO TAKE PLACE IN 'SCOPE'

                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;;*****************************************************************
```

```
2314                    ;*****************************************************************
2315                    ;*TEST 16       EXPNT, ESPAD.B[AC0/3] DATAPATH
2316                    ;
2317                    ;       THIS TEST CONSISTS OF 4 SEPARATE SUBTESTS OF THE EXPONENT/SIGN DATAPATH.
2318                    ;       SPECIFICALLY, THE EXPONENT/SIGN SCRATCHPADS ON THE "B" SIDE, AC0-AC3, ARE
2319                    ;       EMPLOYED.
2320                    ;
2321                    ;       DATA IS PASSED THRU THE:
2322                    ;
2323                    ;         INBUF -> SD/FBUS.E -> EXPNT.ALU -> SD/ER -> SSPAD/ESPAD.B[AC0...AC3]
2324                    ;
2325                    ;       AND
2326                    ;
2327                    ;         SSPAD/ESPAD.B[AC0...AC3] -> SD/FBUS.E -> FPOUTMUX -> BUSDIN
2328                    ;
2329                    ;       DATAPATH, TO VERIFY ITS INTEGRITY.  THERE IS ONE SUBTEST FOR EACH
2330                    ;       ACCUMULATOR; A "1" IS RIPPLED THRU BITS<7:0> FOR THE DATA PATTERN.
2331                    ;
2332                    ;       ----------
2333                    ;       REGISTER/LOCATION USE:
2334                    ;
2335                    ;       MFAC0+  -INITIAL SIGN/EXPNT DATA, FROM TABLE
2336                    ;       MFAC1+  -STORED HFP WORD-A (SIGN/EXPNT) DATA
2337                    ;
2338                    ;       AC0     -(TEMP)
2339                    ;       AC1     -(TEMP)
2340                    ;       AC2     -(TEMP)
2341                    ;       AC3     -(TEMP)
2342                    ;
2343                    ;       R0      -ACCUMULATOR ## CNTR, (0) -> (3)
2344                    ;       R5      -DATA TABLE PTR
2345                    ;
2346                    ;       ----------
2347                    ;       MODULE/ERROR INFO:
2348                    ;
2349                    ;       FNUA/K8
2350                    ;         CROM/LATCHES, JREG/BUA,
2351                    ;         F.BUS.E/ENABLES/DRIVERS, INBUF.A, FPIN.MUX(DMUX)<15:07>
2352                    ;
2353                    ;       FEXP/K9
2354                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.B, SSPAD.B, SS/SD.LOGIC,
2355                    ;         EALU.DATA/CNTL(B), ER-REG/CLK
2356                    ;
2357                    ;       FMUL/K10
2358                    ;         FPOUT.MUX(PORT-0/ENABLE)
2359                    ;
2360                    ;       FALU/K11
2361                    ;         [PREVIOUSLY VERIFIED]
2362                    ;
2363                    ;
2364                    ;*****************************************************************
2365  006772  000004   TST16:  SCOPE
2366                    ;
2367  006774  170127  040040          LDFPS   #040040                 ;INTR-DISABLE/F-MODE/TRUNC
2368  007000  005037  002650          CLR     MFAC0+2                 ;WORD-B WILL ALWAYS BE ZERO
2369                                                                   ;
```

```
2370                                                    ;
2371                                    ;---------------ESPAD.B[AC0] DATAPATH---------------
2372  007004  012705  007344            10$:    MOV     #40$,R5         ;DATA TABLE PTR
2373  007010  005000                            CLR     R0              ;BUMP ACC CNTR
2374                                                    ;
2375                                    ;*DATA LOOP ENTERS HERE*
2376  007012  005237  002640            20$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2377  007016  012537  002646                    MOV     (R5)+,MFAC0+0   ;GET WORD-A
2378  007022  023727  002646  000012          CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2379  007030  001421                            BEQ     11$             ;YES
2380                                                    ;
2381  007032  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2382                                    ;--------ERROR-LOOP-ENTERS-HERE--------
2383                                                    ;
2384  007034  172437  002646            60$:    LDF     MFAC0,AC0       ;INBUF<14:07> -> ER -> E[DF]
2385                                                    ;INBUF<15> -> SD -> S[DF]
2386  007040  174037  002656                    STF     AC0,MFAC1       ;EB[DF] -> FPOUTMUX
2387                                                    ;S[DF] -> SD -> FPOUTMUX
2388  007044  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2389  007050  001371                            BNE     60$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2390                                                    ;
2391  007052  042737  000177  002658          BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
2392                                                    ;
2393  007060  023737  002646  002656          CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2394  007066  001751                            BEQ     20$             ;BR IF AGREE
2395  007070  104066                            ERROR   66              ;ELSE ESPAD.B DATA PATH ERROR
2396                                    ;"ESPAD.B LDX/STX DATAPATH ERR"
2397                                    ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2398                                    ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2399                                    ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2400                                                    ;
2401  007072  000747                            BR      20$             ;LOOP
2402
2403                                    ;---------------ESPAD.B[AC1] DATAPATH---------------
2404  007074  012705  007344            11$:    MOV     #40$,R5         ;DATA TABLE PTR
2405  007100  005200                            INC     R0              ;BUMP ACC CNTR
2406                                                    ;
2407                                    ;*DATA LOOP ENTERS HERE*
2408  007102  005237  002640            21$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2409  007106  012537  002646                    MOV     (R5)+,MFAC0+0   ;GET WORD-A
2410  007112  023727  002646  000012          CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2411  007120  001421                            BEQ     12$             ;YES
2412                                                    ;
2413  007122  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2414                                    ;--------ERROR-LOOP-ENTERS-HERE--------
2415                                                    ;
2416  007124  172537  002646            61$:    LDF     MFAC0,AC1       ;INBUF<14:07> -> ER -> E[DF]
2417                                                    ;INBUF<15> -> SD -> S[DF]
2418  007130  174137  002656                    STF     AC1,MFAC1       ;EB[DF] -> FPOUTMUX
2419                                                    ;S[DF] -> SD  > FPOUTMUX
2420  007134  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2421  007140  001371                            BNE     61$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2422                                                    ;
2423  007142  042737  000177  002658          BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
2424                                                    ;
2425  007150  023737  002646  002658          CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
```

```
                        BEQ     21$                     ;BR IF AGREE
007156  001751          ERROR   66                      ;ELSE ESPAD.B DATA PATH ERROR
007160  104066          ;"ESPAD.B LDX/STX DATAPATH ERR"
                        ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
                        ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
                        ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
                        ;
007162  000747          BR      21$                     ;LOOP

                    ;---------------ESPAD.B[AC2] DATAPATH---------------
007164  012705  007344  12$:    MOV     #40$,R5         ;DATA TABLE PTR
007170  005200          INC     R0                      ;BUMP ACC CNTR
                        ;
                    ;*DATA LOOP ENTERS HERE*
007172  005237  002640  22$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
007176  012537  002646          MOV     (R5)+,MFAC0+0   ;GET WORD-A
007202  023727  002646  000012  CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
007210  001421          BEQ     13$                     ;YES
                        ;
007212  104406          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                        ;--------ERROR-LOOP-ENTERS-HERE--------
                        ;
007214  172637  002646  62$:    LDF     MFAC0,AC2       ;INBUF<14:07> -> ER -> E[DF]
                                                        ;INBUF<15> -> SD -> S[DF]
007220  174237  002656          STF     AC2,MFAC1       ;EB[DF] -> FPOUTMUX
                                                        ;S[DF] -> SD -> FPOUTMUX
007224  105737  002645          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
007230  001371          BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                        ;
007232  042737  000177  002656  BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
                        ;
007240  023737  002646  002656  CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
007246  001751          BEQ     22$                     ;BR IF AGREE
007250  104066          ERROR   66                      ;ELSE ESPAD.B DATA PATH ERROR
                        ;"ESPAD.B LDX/STX DATAPATH ERR"
                        ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
                        ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
                        ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
                        ;
007252  000747          BR      22$                     ;LOOP

                    ;---------------ESPAD.B[AC3] DATAPATH---------------
007254  012705  007344  13$:    MOV     #40$,R5         ;DATA TABLE PTR
007260  005200          INC     R0                      ;BUMP ACC CNTR
                        ;
                    ;*DATA LOOP ENTERS HERE*
007262  005237  002640  23$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
007266  012537  002646          MOV     (R5)+,MFAC0+0   ;GET WORD-A
007272  023727  002646  000012  CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
007300  001435          BEQ     TST17           ;;      ;NEXT TEST IF DONE
                        ;
007302  104406          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                        ;--------ERROR-LOOP-ENTERS-HERE--------
                        ;
007304  172737  002646  63$:    LDF     MFAC0,AC3       ;INBUF<14:07> -> ER -> E[DF]
                                                        ;INBUF<15> -> SD -> S[DF]
```

```
2482  007310  174337  002656          STF     AC3,MFAC1               ;EB[DF] -> FPOUTMUX
2483                                                                    ;S[DF] -> SD -> FPOUTMUX
2484  007314  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2485  007320  001371                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2486                                                                    ;
2487  007322  042737  000177  002656  BIC     #177,MFAC1+0            ;ZAP FRAC TO ZEROES
2488                                                                    ;
2489  007330  023737  002646  002656  CMP     MFAC0,MFAC1             ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2490  007336  001751                  BEQ     23$                     ;BR IF AGREE
2491  007340  104066                  ERROR   66                      ;ELSE ESPAD.B DATA PATH ERROR
2492                                  ;"ESPAD.B LDX/STX DATAPATH ERR"
2493                                  ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2494                                  ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2495                                  ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2496                                                                    ;
2497  007342  000747                  BR      23$                     ;LOOP
2498
2499                          ;
2500                          ;////////////////////////////////////////////////////////////////////////
2501                          ;
2502                          ;       DATA FOR ABOVE TEST:
2503                          ;
2504                          ;                                       SIGN    EXPNT
2505
2506  007344  000000          40$:    .WORD   000000                  ;0      000
2507
2508  007346  177600                  .WORD   177600                  ;1      377
2509
2510  007350  000200                  .WORD   000200                  ;0      001
2511
2512  007352  100400                  .WORD   100400                  ;1      002
2513
2514  007354  001000                  .WORD   001000                  ;0      004
2515
2516  007356  102000                  .WORD   102000                  ;1      010
2517
2518  007360  004000                  .WORD   004000                  ;0      020
2519
2520  007362  010000                  .WORD   010000                  ;0      040
2521
2522  007364  120000                  .WORD   120000                  ;1      100
2523
2524  007366  140000                  .WORD   140000                  ;1      200
2525
2526  007370  000000                  .WORD   000000                  ;0      000
2527
2528  007372  000012                  .WORD   12      ;<DONE>
2529
2530
2531
2532                          ;;**********************************************************************
2533                          ;*TEST 17       EXPNT, ESPAD.A[AC0/3] DATAPATH
2534                          ;
2535                          ;       THIS TEST VERIFIES THE INTEGRITY OF THE "A" SIDE EXPONENT/SIGN
2536                          ;       DATAPATH AND SCRATCHPADS.
2537                          ;       THIS TEST REPEATS THE SAME DATA PATTERNS AS ABOVE, BUT THIS TIME
```

```
                        ;       EMPLOYS THE "A" SIDE EXPONENT/SIGN SCRATCHPADS.
                        ;
                        ;       DATA IS PASSED THRU THE:
                        ;
                        ;         INBUF -> SD/FBUS.E -> EXPNT.ALU -> SD/ER -> SSPAD/ESPAD.B[AC0...AC3] ->
                        ;           -> EXPNT.ALU -> SS/ER -> SSPAD/ESPAD.A[AC0...AC3]
                        ;
                        ;       AND
                        ;
                        ;         SSPAD/ESPAD.A[AC0...AC3] -> EXPNT.ALU -> SS/ER -> SSPAD/ESPAD.B[AC0...AC3] ->
                        ;           -> SD/FBUS.E -> FPOUTMUX -> BUSDIN
                        ;
                        ;       DATAPATH, TO VERIFY ITS INTEGRITY.  THERE IS ONE SUBTEST FOR EACH
                        ;       ACCUMULATOR; A "1" IS RIPPLED THRU BITS<7:0> FOR THE DATA PATTERN.
                        ;
                        ;       THE PASSAGE THRU ESPAD.B MUST BE DONE BECAUSE THERE IS NO WAY TO
                        ;       READ ESPAD A DIRECTLY, VIA LOAD/STORE-TYPE INSTRUCTIONS.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -INITIAL SIGN/EXPNT DATA, FROM TABLE
                        ;       MFAC1+  -STORED HFP WORD-A (SIGN/EXPNT) DATA
                        ;
                        ;       AC0     -(TEMP)
                        ;       AC1     -(TEMP)
                        ;       AC2     -(TEMP)
                        ;       AC3     -(TEMP)
                        ;
                        ;       R0      -ACCUMULATOR ## CNTR, (0) -> (3)
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA,
                        ;         F.BUS.E/ENABLES/DRIVERS, INBUF.A, FPIN.MUX(DMUX)<15:07>
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.A, SSPAD.A, SS/SD.LOGIC,
                        ;         EALU.DATA/CNTL(A,B), ER-REG/CLK
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;
                        ;*****************************************************************
007374  000004           ST17:  SCOPE
                        ;                                               :
007376  170127  040040          LDFPS   #040040                         ;INTR-DISABLE/F-MODE/TRUNC
007402  005037  002650          CLR     MFAC0+2                         ;WORD-B WILL ALWAYS BE ZERO
                        ;                                               :
```

```
2594                                                    ;
2595                                    ;---------------ESPAD.A[AC0] DATAPATH---------------
2596  007406  012705  007766            10$:    MOV     #40$,R5         ;DATA TABLE PTR
2597  007412  005000                            CLR     R0              ;BUMP ACC CNTR
2598                                                    ;
2599                                    ;*DATA LOOP ENTERS HERE*
2600  007414  005237  002640            20$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2601  007420  012537  002646                    MOV     (R5)+,MFAC0+0   ;GET WORD-A
2602  007424  023727  002646  000012            CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2603  007432  001423                            BEQ     11$             ;YES
2604                                                    ;
2605  007434  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2606                                            ;--------ERROR-LOOP-ENTERS-HERE--------
2607                                                    ;
2608  007436  172437  002646            60$:    LDF     MFAC0,AC0       ;INBUF<14:07> -> ER -> E[DF]
2609                                                    ;INBUF<15> -> SD -> S[DF]
2610  007442  174000                            STF     AC0,AC0         ;EA[DF] -> E[SF]
2611                                                    ;S[DF] -> SS -> S[SF]
2612  007444  172400                            LDF     AC0,AC0         ;EB[SF] -> E[DF]
2613                                                    ;SS -> SD -> S[DF]
2614  007446  174037  002656                    STF     AC0,MFAC1       ;EB[DF] -> FPOUTMUX
2615                                                    ;S[DF] -> SD -> FPOUTMUX
2616  007452  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2617  007456  001367                            BNE     60$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2618                                                    ;
2619  007460  042737  000177  002656          BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
2620                                                    ;
2621  007466  023737  002646  002656          CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2622  007474  001747                            BEQ     20$             ;BR IF AGREE
2623  007476  104067                            ERROR   67              ;ELSE ESPAD.A DATA PATH ERROR
2624                                            ;"ESPAD.A LDX/STX DATAPATH ERR"
2625                                            ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2626                                            ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2627                                            ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS '
2628                                                    ;
2629  007500  000745                            BR      20$             ;LOOP
2630
2631                                    ;---------------ESPAD.A[AC1] DATAPATH---------------
2632  007502  012705  007766            11$:    MOV     #40$,R5         ;DATA TABLE PTR
2633  007506  005200                            INC     R0              ;BUMP ACC CNTR
2634                                                    ;
2635                                    ;*DATA LOOP ENTERS HERE*
2636  007510  005237  002640            21$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2637  007514  012537  002646                    MOV     (R5)+,MFAC0+0   ;GET WORD-A
2638  007520  023727  002646  000012            CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2639  007526  001423                            BEQ     12$             ;YES
2640                                                    ;
2641  007530  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2642                                            ;--------ERROR-LOOP-ENTERS-HERE--------
2643                                                    ;
2644  007532  172537  002646            61$:    LDF     MFAC0,AC1       ;INBUF<14:07> -> ER -> E[DF]
2645                                                    ;INBUF<15> -> SD -> S[DF]
2646  007536  174101                            STF     AC1,AC1         ;EA[DF] -> E[SF]
2647                                                    ;S[DF] -> SS -> S[SF]
2648  007540  172501                            LDF     AC1,AC1         ;EB[SF] -> E[DF]
2649                                                    ;SS -> SD -> S[DF]
```

```
2650  007542  174137  002656                  STF     AC1,MFAC1               ;EB[DF] -> FPOUTMUX
2651                                                                          ;S[DF] -> SD -> FPOUTMUX
2652  007546  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2653  007552  001367                          BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2654                                                                          ;
2655  007554  042737  000177  002656          BIC     #177,MFAC1+0            ;ZAP FRAC TO ZEROES
2656                                                                          ;
2657  007562  023737  002646  002656          CMP     MFAC0,MFAC1             ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2658  007570  001747                          BEQ     21$                     ;BR IF AGREE
2659  007572  104067                          ERROR   67                      ;ELSE ESPAD.A DATA PATH ERROR
2660                                          ;"ESPAD.A LDX/STX DATAPATH ERR"
2661                                          ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2662                                          ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2663                                          ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2664                                                                          ;
2665  007574  000745                          BR      21$                     ;LOOP
2666
2667                                  ;---------------ESPAD.A[AC2] DATAPATH---------------
2668  007576  012705  007766          12$:    MOV     #40$,R5                 ;DATA TABLE PTR
2669  007602  005200                          INC     R0                      ;BUMP ACC CNTR
2670                                                                          ;
2671                                  ;*DATA LOOP ENTERS HERE*
2672  007604  005237  002640          22$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
2673  007610  012537  002646                  MOV     (R5)+,MFAC0+0           ;GET WORD-A
2674  007614  023727  002646  000012          CMP     MFAC0+0,#12             ;DONE WITH THIS ACC ?
2675  007622  001423                          BEQ     13$                     ;YES
2676                                                                          ;
2677  007624  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
2678                                          ;--------ERROR-LOOP-ENTERS-HERE--------
2679                                                                          ;
2680  007626  172637  002646          62$:    LDF     MFAC0,AC2               ;INBUF<14:07> -> ER -> E[DF]
2681                                                                          ;INBUF<15> -> SD -> S[DF]
2682  007632  174202                          STF     AC2,AC2                 ;EA[DF] -> E[SF]
2683                                                                          ;S[DF] -> SS -> S[SF]
2684  007634  172602                          LDF     AC2,AC2                 ;EB[SF] -> E[DF]
2685                                                                          ;SS -> SD -> S[DF]
2686  007636  174237  002656                  STF     AC2,MFAC1               ;EB[DF] -> FPOUTMUX
2687                                                                          ;S[DF] -> SD -> FPOUTMUX
2688  007642  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2689  007646  001367                          BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2690                                                                          ;
2691  007650  042737  000177  002656          BIC     #177,MFAC1+0            ;ZAP FRAC TO ZEROES
2692                                                                          ;
2693  007656  023737  002646  002656          CMP     MFAC0,MFAC1             ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2694  007664  001747                          BEQ     22$                     ;BR IF AGREE
2695  007666  104067                          ERROR   67                      ;ELSE ESPAD.A DATA PATH ERROR
2696                                          ;"ESPAD.A LDX/STX DATAPATH ERR"
2697                                          ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2698                                          ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2699                                          ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2700                                                                          ;
2701  007670  000745                          BR      22$                     ;LOOP
2702
2703                                  ;----------   -ESPAD.A[AC3] DATAPATH---------------
2704  007672  012705  007766          13$:    MOV     #40$,R5                 ;DATA TABLE PTR
2705  007676  005200                          INC                             ;BUMP ACC CNTR
```

```
2706                                                    ;
2707                                    ;*DATA LOOP ENTERS HERE*
2708  007700  005237  002640    23$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2709  007704  012537  002646            MOV     (R5)+,MFAC0+0   ;GET WORD-A
2710  007710  023727  002646  000012    CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2711  007716  001437                    BEQ     TST20     ::    ;NEXT TEST IF DONE
2712                                                    ;
2713  007720  104406                    ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2714                                    ;--------ERROR-LOOP-ENTERS-HERE--------
2715                                                    ;
2716  007722  172737  002646    63$:    LDF     MFAC0,AC3       ;INBUF<14:07> -> ER -> E[DF]
2717                                                    ;INBUF<15> -> SD -> S[DF]
2718  007726  174303                    STF     AC3,AC3         ;EA[DF] -> E[SF]
2719                                                    ;S[DF] -> SS -> S[SF]
2720  007730  172703                    LDF     AC3,AC3         ;EB[SF] -> E[DF]
2721                                                    ;SS -> SD -> S[DF]
2722  007732  174337  002656            STF     AC3,MFAC1       ;EB[DF] -> FPOUTMUX
2723                                                    ;S[DF] -> SD -> FPOUTMUX
2724  007736  105737  002645            TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2725  007742  001367                    BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2726                                                    ;
2727  007744  042737  000177  002656    BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
2728                                                    ;
2729  007752  023737  002646  002656    CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2730  007760  001747                    BEQ     23$             ;BR IF AGREE
2731  007762  104067                    ERROR   67              ;ELSE ESPAD.A DATA PATH ERROR
2732                                    ;"ESPAD.A LDX/STX DATAPATH ERR"
2733                                    ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (0) -> (3)
2734                                    ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2735                                    ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2736                                                    ;
2737  007764  000745                    BR      23$             ;LOOP
2738
2739                                    ;
2740                                    ;////////////////////////////////////////////////////////////////////////
2741                                    ;
2742                                    ;       DATA FOR ABOVE TEST:
2743                                    ;
2744                                    ;                               SIGN    EXPNT
2745
2746  007766  000000            40$:    .WORD   000000          ;0      000*
2747
2748  007770  177600                    .WORD   177600          ;1      377
2749
2750  007772  000200                    .WORD   000200          ;0      001
2751
2752  007774  100400                    .WORD   100400          ;1      002
2753
2754  007776  001000                    .WORD   001000          ;0      004
2755
2756  010000  102000                    .WORD   102000          ;1      010
2757
2758  010002  004000                    .WORD   004000          ;0      020
2759
2760  010004  010000                    .WORD   010000          ;0      040
2761
```

```
2762  010006  120000                    .WORD   120000          ;1      100
2763
2764  010010  140000                    .WORD   140000          ;1      200
2765
2766  010012  000000                    .WORD   000000          ;0      000
2767
2768  010014  000012                    .WORD   12      ;<DONE>
2769
2770
2771
2772                          ;;*****************************************************************
2773                          ;*TEST 20       EXPNT, ESPAD.B[AC4/5] DATAPATH
2774                          ;
2775                          ;       THIS TEST VERIFIES THE INTEGRITY OF THE "B" SIDE EXPONENT/SIGN
2776                          ;       DATAPATH AND SCRATCHPADS, AC4 AND AC5.
2777                          ;       THIS TEST REPEATS THE SAME DATA PATTERNS AS ABOVE, BUT THIS TIME
2778                          ;       EMPLOYS THE "B" SIDE EXPONENT/SIGN SCRATCHPADS.
2779                          ;
2780                          ;       DATA IS PASSED THRU THE:
2781                          ;
2782                          ;         INBUF -> SD/FBUS.E -> E.XPNT.ALU -> SD/ER -> SSPAD/ESPAD.A[AC2/3] ->
2783                          ;           -> EXPNT.ALU -> SS/ER -> SSPAD/ESPAD.B[AC4/5]
2784                          ;
2785                          ;       AND
2786                          ;
2787                          ;         SSPAD/ESPAD.B[AC4/5] -> EXPNT.ALU -> SS/ER -> SSPAD/ESPAD.B[AC2/3] ->
2788                          ;           -> SD/FBUS.E -> FPOUTMUX -> BUSDIN
2789                          ;
2790                          ;       DATAPATH, TO VERIFY ITS INTEGRITY.  THERE IS ONE SUBTEST FOR EACH
2791                          ;       ACCUMULATOR; A "1" IS RIPPLED THRU BITS<7:0> FOR THE DATA PATTERN.
2792                          ;
2793                          ;       THE PASSAGE THRU ESPAD.A[DF] MUST BE DONE BECAUSE THERE IS NO WAY TO
2794                          ;       READ ESPAD.B[AC4/5] DIRECTLY, VIA LOAD/STORE-TYPE INSTRUCTIONS.
2795                          ;
2796                          ;       ----------
2797                          ;       REGISTER/LOCATION USE:
2798                          ;
2799                          ;       MFAC0+  -INITIAL SIGN/EXPNT DATA, FROM TABLE
2800                          ;       MFAC1+  -STORED HFP WORD-A (SIGN/EXPNT) DATA
2801                          ;
2802                          ;       AC2     -(TEMP)
2803                          ;       AC3     -(TEMP)
2804                          ;       AC4     -(TEMP)
2805                          ;       AC5     -(TEMP)
2806                          ;
2807                          ;       R0      -ACCUMULATOR ## CNTR, (4) -> (5)
2808                          ;       R5      -DATA TABLE PTR
2809                          ;
2810                          ;       ----------
2811                          ;       MODULE/ERROR INFO:
2812                          ;
2813                          ;       FNUA/K8
2814                          ;         CROM/LATCHES, JREG/BUA, FIR.SF/DF,
2815                          ;         F.BUS.E/ENABLES/DRIVERS, INBUF.A, FPIN.MUX(DMUX)<15:07>
2816                          ;
2817                          ;       FEXP/K9
```

```
2818                                    ;       CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.A/B, SSPAD.A/B, SS/SD.LOGIC,
2819                                    ;       ESPAD.A/B.ADDR, EALU.DATA/CNTL(A,B), ER-REG/CLK
2820                                    ;
2821                                    ;       FMUL/K10
2822                                    ;         [PREVIOUSLY VERIFIED]
2823                                    ;
2824                                    ;       FALU/K11
2825                                    ;         [PREVIOUSLY VERIFIED]
2826                                    ;
2827                                    ;
2828                                    ;.*****************************************************************
2829  010016  000004                    TST20:  SCOPE
2830                                                                    ;
2831  010020  170127  040040                   LDFPS   #040040         ;INTR-DISABLE/F-MODE/TRUNC
2832  010024  005037  002650                   CLR     MFAC0+2         ;WORD-B WILL ALWAYS BE ZERO
2833                                                                    ;
2834                                                                    ;
2835                                    ;---------------ESPAD.B[AC4] DATAPATH---------------
2836  010030  012705  010222            10$:    MOV     #40$,R5         ;DATA TABLE PTR
2837  010034  012700  000004                   MOV     #4,R0           ;SET ACC CNTR
2838                                                                    ;
2839                                           ;*DATA LOOP ENTERS HERE*
2840  010040  005237  002640            20$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
2841  010044  012537  002646                   MOV     (R5)+,MFAC0+0   ;GET WORD-A
2842  010050  023727  002646  000012        CMP     MFAC0+0,#12     ;DONE WITH THIS ACC ?
2843  010056  001423                        BEQ     11$             ;YES
2844                                                                    ;
2845  010060  104406                        ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
2846                                         ;--------ERROR-LOOP-ENTERS-HERE--------
2847                                                                    ;
2848  010062  172737  002646          60$:    LDF     MFAC0,AC3       ;INBUF<14:07> -> ER -> E[DF]
2849                                                                    ;INBUF<15> -> SD -> S[DF]
2850  010066  174304                        STF     AC3,AC4         ;EA[DF] -> E[SF]
2851                                                                    ;S[DF] -> SS -> S[SF]
2852  010070  172704                        LDF     AC4,AC3         ;EB[SF] -> E[DF]
2853                                                                    ;SS -> SD -> S[DF]
2854  010072  174337  002656                STF     AC3,MFAC1       ;EB[DF] -> FPOUTMUX
2855                                                                    ;S[DF] -> SD -> FPOUTMUX
2856  010076  105737  002645                TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2857  010102  001367                        BNE     60$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2858                                                                    ;
2859  010104  042737  000177  002656        BIC     #177,MFAC1+0    ;ZAP FRAC TO ZEROES
2860                                                                    ;
2861  010112  023737  002646  002656        CMP     MFAC0,MFAC1     ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2862  010120  001747                        BEQ     20$             ;BR IF AGREE
2863  010122  104066                        ERROR   66              ;ELSE ESPAD.B DATA PATH ERROR
2864                                         ;"ESPAD.B LDX/STX DATAPATH ERR"
2865                                         ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (4) -> (5)
2866                                         ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2867                                         ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2868                                                                    ;
2869  010124  000745                        BR      20$             ;LOOP
2870
2871                                    ;---------------ESPAD.B[AC5] DATAPATH---------------
2872  010126  012705  010222            11$:    MOV     #40$,R5         ;DATA TABLE PTR
2873  010132  005200                         INC     R0              ;BUMP ACC CNTR
```

```
2874                                                    ;
2875                                                    ;*DATA LOOP ENTERS HERE*
2876  010134  005237  002640          21$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
2877  010140  012537  002646                  MOV     (R5)+,MFAC0+0           ;GET WORD-A
2878  010144  023727  002646  000012          CMP     MFAC0+0,#12             ;DONE WITH THIS ACC ?
2879  010152  001437                          BEQ     TST21           ;;      ;NEXT TEST IF DONE
2880                                                                          ;
2881  010154  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
2882                                                    ;--------ERROR-LOOP-ENTERS-HERE--------
2883                                                                          ;
2884  010156  172637  002646          61$:    LDF     MFAC0,AC2               ;INBUF<14:07> -> ER -> E[DF]
2885                                                                          ;INBUF<15> -> SD -> S[DF]
2886  010162  174205                          STF     AC2,AC5                 ;EA[DF] -> E[SF]
2887                                                                          ;S[DF] -> SS -> S[SF]
2888  010164  172605                          LDF     AC5,AC2                 ;EB[SF] -> E[DF]
2889                                                                          ;SS -> SD -> S[DF]
2890  010166  174237  002656                  STF     AC2,MFAC1               ;EB[DF] -> FPOUTMUX
2891                                                                          ;S[DF] -> SD -> FPOUTMUX
2892  010172  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2893  010176  001367                          BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2894                                                                          ;
2895  010200  042737  000177  002656          BIC     #177,MFAC1+0            ;ZAP FRAC TO ZEROES
2896                                                                          ;
2897  010206  023737  002646  002656          CMP     MFAC0,MFAC1             ;(EXPECTED/LOADED) = (RECEIVED/STORED) ?
2898  010214  001747                          BEQ     21$                     ;BR IF AGREE
2899  010216  104066                          ERROR   66                      ;ELSE ESPAD.B DATA PATH ERROR
2900                                                    ;"ESPAD.B LDX/STX DATAPATH ERR"
2901                                                    ;       ACC### = HFP ACCUMULATOR NUMBER UNDER TEST, (4) -> (5)
2902                                                    ;       E-DATA = LOADED/EXP'D DATA IN SIGN/EXPNT BIT<15:07>
2903                                                    ;       R-DATA = STORED/RECEIVED DATA, FORMAT AS LOADED
2904                                                                          ;
2905  010220  000745                          BR      21$                     ;LOOP
2906
2907                                        ;
2908                                        ;
2909                                        ;////////////////////////////////////////////////////////////////////////////////
2910                                        ;
2911                                        ;       DATA FOR ABOVE TEST:
2912                                        ;
2913                                        ;                                       SIGN    EXPNT
2914
2915  010222  000000                  40$:    .WORD   000000          ;0      000
2916
2917  010224  177600                          .WORD   177600          ;1      377
2918
2919  010226  000200                          .WORD   000200          ;0      001
2920
2921  010230  100400                          .WORD   100400          ;1      002
2922
2923  010232  001000                          .WORD   001000          ;0      004
2924
2925  010234  102000                          .WORD   102000          ;1      010
2926
2927  010236  004000                          .WORD   004000          ;0      020
2928
2929  010240  010000                          .WORD   010000          ;0      040
```

```
2930
2931  010242  120000                    .WORD   120000          ;1      100
2932
2933  010244  140000                    .WORD   140000          ;1      200
2934
2935  010246  000000                    .WORD   000000          ;0      000
2936
2937  010250  000012                    .WORD   12      ;<DONE>
2938
2939
2940
2941                              ;;*****************************************************************
2942                              ;*TEST 21       EXPNT, "LDEXP/STEXP" FPINMUX(DOUT) DATAPATH
2943                              ;
2944                              ;       THIS TEST RUNS A RIPPLING-"1" PATTERN THRU THE:
2945                              ;
2946                              ;         BM/GPR -> DOUT -> FPINMUX(DOUT) -> INBUF -> ER -> ESPAD
2947                              ;
2948                              ;       PATH TO CHECK ITS VALIDITY, IN POSITIONS<14:07>.
2949                              ;
2950                              ;       ----------
2951                              ;       REGISTER/LOCATION USE:
2952                              ;
2953                              ;       AC0     -(TEMP)
2954                              ;
2955                              ;       R0      -INPUT EXPNT FOR "LDEXP", EXP'D RESULT BACK
2956                              ;       R1      -OUTPUT FROM "STEXP"
2957                              ;       R5      -DATA TABLE PTR
2958                              ;
2959                              ;       ----------
2960                              ;       MODULE/ERROR INFO:
2961                              ;
2962                              ;       FNUA/K8
2963                              ;         CROM/LATCHES, JREG/BUA,
2964                              ;         F.BUS.E/ENABLES/DRIVERS, INBUF.A, FPIN.MUX(DOUT)<14:07>
2965                              ;
2966                              ;       FEXP/K9
2967                              ;         CROM/LATCHES, BUT<2:0>.LOGIC, EALU.DATA/CNTL(B);
2968                              ;
2969                              ;       FMUL/K10
2970                              ;         [PREVIOUSLY VERIFIED]
2971                              ;
2972                              ;       FALU/K11
2973                              ;         [PREVIOUSLY VERIFIED]
2974                              ;
2975                              ;;*****************************************************************
2976  010252  000004              TST21:  SCOPE
2977                                                                  ;
2978  010254  170127  040040             LDFPS   #040040             ;INTR-DISABLE/F-MODE/TRUNC
2979  010260  012705  010340             MOV     #40$,R5             ;DATA TABLE PTR
2980                                                                  ;
2981                                      ;*DATA TABLE LOOP ENTERS HERE*
2982  010264  005237  002640     10$:    INC     DWLOOP              ;BUMP CLOCK IN.A.LOOP COUNT
2983  010270  012500                     MOV     (R5)+,R0            ;GET INITIAL DATA
2984  010272  020027  000666             CMP     R0,#666             ;DONE ?
2985  010276  001431                     BEQ     TST22           ;;  ;NEXT TEST IF YES
```

```
2986  010300  162700  000200                SUB     #200,R0                 ;BIAS EXPNT BY -200, SINCE LDEXP ADDS +200
2987                                        ;                               ;
2988  010304  104406                       ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
2989                                        ;--------ERROR-LOOP-ENTERS-HERE--------
2990                                        ;                               ;
2991  010306  176400                 63$:   LDEXP   R0,AC0                  ;(R0)+200 -> FPINMUX(DOUT) -> E[AC0]
2992  010310  175001                       STEXP   AC0,R1                  ;EB[AC0]-(200) -> R1
2993  010312  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
2994  010316  001373                       BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
2995                                        ;                               ;
2996  010320  062701  000200               ADD     #200,R1                 ;BIAS EXPNT BY +200, SINCE STEXP SUBS -200
2997  010324  062700  000200               ADD     #200,R0                 ;PUT R0 BACK WHERE IT WAS, FOR DISPLAY
2998                                        ;                               ;
2999  010330  020001                       CMP     R0,R1                   ;LOADED = STORED ???
3000  010332  001754                       BEQ     10$                     ;BR IF AOK
3001  010334  104112                       ERROR   112                     ;ELSE ERROR
3002                                        ;"LDEXP/STEXP FPINMUX(DOUT) ERR"
3003                                        ;       E-EXPNT = EXP'D/LOADED EXPNT
3004                                        ;       R-EXPNT = RCV'D/STORED EXPNT
3005                                        ;                               ;
3006  010336  000752                       BR      10$                     ;MORE
3007                                ;
3008                                ;
3009                                ;////////////////////////////////////////////////////////////////////////
3010                                ;
3011                                ;       DATA FOR ABOVE TEST:
3012                                ;
3013                                ;          GPR/EXPNT  FPINMUX<14:07>
3014                                ;
3015  010340  000200                40$:    .WORD   200     ;10.000.000
3016                                ;
3017  010342  000100                        .WORD   100     ;01.000.000
3018                                ;
3019  010344  000040                        .WORD   040     ;00.100.000
3020                                ;
3021  010346  000020                        .WORD   020     ;00.010.000
3022                                ;
3023  010350  000010                        .WORD   010     ;00.001.000
3024                                ;
3025  010352  000004                        .WORD   004     ;00.000.100
3026                                ;
3027  010354  000002                        .WORD   002     ;00.000.010
3028                                ;
3029  010356  000001                        .WORD   001     ;00.000.001
3030                                ;
3031  010360  000666                        .WORD   666     ;<DONE>
3032
3033
3034
3035                                ;;*****************************************************************
3036                                ;*TEST 22        EXPNT, ESPAD.B ADDRESSING VIA R[DF] AND R[SF]
3037                                ;
3038                                ;       THIS TEST VERIFIES THE SCRATCHPAD ADDRESSING FUNCTIONS:
3039                                ;
3040                                ;         R[SF]<3:0> == "0"*FIR8<2:0>  AND
3041                                ;
```

```
                        ;       R[DF]<3:0> == "00"*FIR8<7:6>
                        ;
                        ;       ADDRESS MODES IN THE ESPAD.B SCRATCHPAD ADDRESSING LOGIC.
                        ;
                        ;       THIS TEST LOOKS FOR STUCK 0/1 CONDITIONS IN THE 3 LOW ORDER
                        ;       BITS OF THE SCRATCHPAD ADDRESS PATH, FROM FIR -> THE SPAD.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -OUTPUT, AFTER ADDRESSING CHECK
                        ;
                        ;       AC0     -(TEMP)
                        ;
                        ;       AC5     -(TEMP)
                        ;
                        ;       R4      -MASK TO ZAP FRACTION TO ZEROES, WORD.A
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FIR.SF/DF,
                        ;         FP.EMIT.E, F.BUS.E/ENABLES/DRIVERS
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.B, SSPAD.B, SS/SD.LOGIC,
                        ;         ESPAD.B.ADDR, EALU.DATA/CNTL
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
010362  000004          TST22:  SCOPE
                        ;
010364  170127  040040          LDFPS   #040040                 ;INTR-DISAB/F-MODE/TRUNC
010370  012704  000177          MOV     #000177,R4              ;MASK TO ZAP FRACTION
                        ;
                        ;-----------R[DF]=FIR8<7:6> ADDRESSING, CHECK FOR STUCK-L'S, BITS<7:6>-----------
010374  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
010376  172437  003164  62$:    LDF     FPSEFZ,AC0              ;ONES -> E[DF]"000"
010402  172537  003060          LDF     FPZERO,AC1              ;ZEROES -> E[DF]"001"
010406  172637  003060          LDF     FPZERO,AC2              ;ZEROES -> E[DF]"010"
010412  174037  002646          STF     AC0,MFAC0               ;EB[DF]"000" -> MEMORY
010416  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
010422  001365                  BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                        ;
010424  040437  002646          BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
010430  005037  002650          CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
010434  104426  003164  002646  CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
010442  001401                  BEQ     .+4                     ;BR IF AGREE
010444  104047                  ERROR   47                      ;SIGNAL ERROR ... IF NOT
```

```
                                        ;"ESPAD.B ADDRS ERROR; R[DF]"
                                        ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
                                        ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
                                                ;
                                        ;-----------R[DF]=FIRB<7:6> ADDRESSING, CHECK FOR STUCK-H'S, BITS<7:6>----------
010446  104406                                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                                ;--------ERROR-LOOP-ENTERS-HERE--------
010450  172737  003164          63$:    LDF     FPSEFZ,AC3              ;ONES -> E[DF]"011"
010454  172537  003060                  LDF     FPZERO,AC1              ;ZEROES -> E[DF]"001"
010460  172637  003060                  LDF     FPZERO,AC2              ;ZEROES -> E[DF]"010"
010464  174337  002646                  STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
010470  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
010474  001365                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                ;
010476  040437  002646                  BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
010502  005037  002650                  CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
010506  104426  003164  002646          CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
010514  001401                          BEQ     .+4                     ;BR IF AGREE
010516  104047                          ERROR   47                      ;SIGNAL ERROR ... IF NOT
                                        ;"ESPAD.B ADDRS ERROR; R[DF]"
                                        ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
                                        ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
                                                ;
                                        ;-----------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-L'S, BITS<2:0>----------
010520  104406                                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                                ;--------ERROR-LOOP-ENTERS-HERE--------
010522  172737  003164          59$:    LDF     FPSEFZ,AC3              ;ONES -> E[DF]"011"
010526  174300                          STF     AC3,AC0                 ;EA[DF]"011" -> E[SF]"000"
010530  170401                          CLRF    AC1                     ;ZEROES -> E[SF]"001"
010532  170402                          CLRF    AC2                     ;ZEROES -> E[SF]"010"
010534  170404                          CLRF    AC4                     ;ZEROES -> E[SF]"100"
010536  172700                          LDF     AC0,AC3                 ;EB[SF]="000" -> E[DF]"011"
010540  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
010544  001366                          BNE     59$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                ;
010546  174337  002646                  STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
010552  040437  002646                  BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
010556  005037  002650                  CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
010562  104426  003164  002646          CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
010570  001401                          BEQ     .+4                     ;BR IF AGREE
010572  104050                          ERROR   50                      ;SIGNAL ERROR ... IF NOT
                                        ;"ESPAD.B ADDRS ERROR; R[SF]"
                                        ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
                                        ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
                                                ;
                                        ;-----------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-H'S, BITS<1:0>----------
010574  104406                                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                                ;--------ERROR-LOOP-ENTERS-HERE--------
010576  172437  003164          60$:    LDF     FPSEFZ,AC0              ;ONES -> E[DF]"000"
010602  174003                          STF     AC0,AC3                 ;EA[DF]"000" -> E[SF]"011"
010604  170401                          CLRF    AC1                     ;ZEROES -> E[SF]"001"
010606  170402                          CLRF    AC2                     ;ZEROES -> E[SF]"010"
010610  172403                          LDF     AC3,AC0                 ;EB[SF]="011" -> E[DF]"000"
010612  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
010616  001367                          BNE     60$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                ;
```

```
3154  010620  174037  002646          STF     AC0,MFAC0               ;EB[DF]"000" -> MEMORY
3155  010624  040437  002646          BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3156  010630  005037  002650          CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3157  010634  104426  003164  002646  CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3158  010642  001401                  BEQ     .+4                     ;BR IF AGREE
3159  010644  104050                  ERROR   50                      ;SIGNAL ERROR ... IF NOT
3160                                  ;"ESPAD.B ADDRS ERROR; R[SF]"
3161                                  ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3162                                  ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3163                                                  ;
3164                          ;-----------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-H, BIT<2>-----------
3165  010646  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3166                                  ;--------ERROR-LOOP-ENTERS-HERE--------
3167  010650  172737  003164  61$:    LDF     FPSEFZ,AC3              ;ONES  , E[DF]"011"
3168  010654  174304                  STF     AC3,AC4                 ;EA[DF]"011" -> E[SF]"100"
3169  010656  170400                  CLRF    AC0                     ;ZEROES -> E[SF]"000"
3170  010660  172704                  LDF     AC4,AC3                 ;EB[SF]="100" -> E[DF]"011"
3171  010662  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3172  010666  001370                  BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3173                                                                  ;
3174  010670  174337  002646          STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
3175  010674  040437  002646          BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3176  010700  005037  002650          CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3177  010704  104426  003164  002646  CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3178  010712  001401                  BEQ     .+4                     ;BR IF AGREE
3179  010714  104050                  ERROR   50                      ;SIGNAL ERROR ... IF NOT
3180                                  ;"ESPAD.B ADDRS ERROR; R[SF]"
3181                                  ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3182                                  ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3183                                                  ;
3184
3185
3186
3187                  ;;*****************************************************************
3188                  ;*TEST 23       EXPNT, ESPAD.A ADDRESSING VIA R[DF] AND R[SF]
3189                  ;
3190                  ;       THIS TEST VERIFIES THE SCRATCHPAD ADDRESSING FUNCTIONS:
3191                  ;
3192                  ;         R[SF]<3:0> == "0"#FIRB<2:0>  AND
3193                  ;
3194                  ;         R[DF]<3:0> == "00"#FIRB<7:6>
3195                  ;
3196                  ;       ADDRESS MODES IN THE ESPAD.A SCRATCHPAD ADDRESSING LOGIC.
3197                  ;
3198                  ;       THIS TEST LOOKS FOR STUCK 0/1 CONDITIONS IN THE 3 LOW ORDER
3199                  ;       BITS OF THE SCRATCHPAD ADDRESS PATH, FROM FIR -> THE SPAD.
3200                  ;
3201                  ;       ----------
3202                  ;       REGISTER/LOCATION USE:
3203                  ;
3204                  ;       MFAC0+  -OUTPUT, AFTER ADDRESSING CHECK
3205                  ;
3206                  ;       AC0     -(TEMP)
3207                  ;
3208                  ;       AC5     -(TEMP)
3209                  ;
```

```
3210                                    ;       R4      -MASK TO ZAP FRACTION TO ZEROES, WORD.A
3211                                    ;
3212                                    ;       ----------
3213                                    ;       MODULE/ERROR INFO:
3214                                    ;
3215                                    ;       FNUA/K8
3216                                    ;         CROM/LATCHES, JREG/BUA, FIR.SF/DF,
3217                                    ;         FP.EMIT.E, F.BUS.E/ENABLES/DRIVERS
3218                                    ;
3219                                    ;       FEXP/K9
3220                                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.A, SSPAD.A, SS/SD.LOGIC,
3221                                    ;         ESPAD.A.ADDR, EALU.DATA/CNTL
3222                                    ;
3223                                    ;       FMUL/K10
3224                                    ;         [PREVIOUSLY VERIFIED]
3225                                    ;
3226                                    ;       FALU/K11
3227                                    ;         [PREVIOUSLY VERIFIED]
3228                                    ;
3229                                    ;*****************************************************************
3230  010716  000004                    TST23:  SCOPE
3231
3232  010720  170127  040040                    LDFPS   #040040                 ;INTR-DISAB/F-MODE/TRUNC
3233  010724  012704  000177                    MOV     #000177,R4              ;MASK TO ZAP FRACTION
3234                                                                            ;
3235                                    ;-----------R[DF]=FIRB<7:6> ADDRESSING, CHECK FOR STUCK-L'S, BITS<7:6>-----------
3236  010730  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3237                                            ;--------ERROR-LOOP-ENTERS-HERE--------
3238  010732  172437  003164          62$:     LDF     FPSEFZ,AC0              ;ONES -> E[DF]"000"
3239  010736  172537  003060                    LDF     FPZERO,AC1              ;ZEROES -> E[DF]"001"
3240  010742  172637  003060                    LDF     FPZERO,AC2              ;ZEROES -> E[DF]"010"
3241  010746  174003                            STF     AC0,AC3                 ;EA[DF]"000" -> E[SF]"011"
3242  010750  174337  002646                    STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
3243  010754  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3244  010760  001364                            BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3245                                                                            ;
3246  010762  040437  002646                    BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3247  010766  005037  002650                    CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3248  010772  104426  003164  002646            CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3249  011000  001401                            BEQ     .+4                     ;BR IF AGREE
3250  011002  104051                            ERROR   51                      ;SIGNAL ERROR ... IF NOT
3251                                            ;"ESPAD.A ADDRS ERROR; R[DF]"
3252                                            ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3253                                            ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3254                                                                            ;
3255                                    ;-----------R[DF]=FIRB<7:6> ADDRESSING, CHECK FOR STUCK-H'S, BITS<7:6>-----------
3256  011004  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3257                                            ;--------ERROR-LOOP-ENTERS-HERE--------
3258  011006  172737  003164          63$:     LDF     FPSEFZ,AC3              ;ONES -> E[DF]"011"
3259  011012  172537  003060                    LDF     FPZERO,AC1              ;ZEROES -> E[DF]"001"
3260  011016  172637  003060                    LDF     FPZERO,AC2              ;ZEROES -> E[DF]"010"
3261  011022  174300                            STF     AC3,AC0                 ;EA[DF]"011" -> E[SF]"000"
3262  011024  174037  002646                    STF     AC0,MFAC0               ;EB[DF]"000" -> MEMORY
3263  011030  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3264  011034  001364                            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3265                                                                            ;
```

```
3266  011036  040437  002646                  BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3267  011042  005037  002650                  CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3268  011046  104426  003164  002646          CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3269  011054  001401                          BEQ     .+4                     ;BR IF AGREE
3270  011056  104051                          ERROR   51                      ;SIGNAL ERROR ... IF NOT
3271                                          ;"ESPAD.A ADDRS ERROR; R[DF]"
3272                                          ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3273                                          ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3274                                                                          ;
3275                                  ;------------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-L'S, BITS<2:0>------------
3276  011060  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3277                                          ;--------ERROR-LOOP-ENTERS-HERE--------
3278  011062  172737  003164          59$:    LDF     FPSEFZ,AC3              ;ONES -> E[DF]"011"
3279  011066  174300                          STF     AC3,AC0                 ;EA[DF]"011" -> E[SF]"000"
3280  011070  170401                          CLRF    AC1                     ;ZEROES -> E[SF]"001"
3281  011072  170402                          CLRF    AC2                     ;ZEROES -> E[SF]"010"
3282  011074  170404                          CLRF    AC4                     ;ZEROES -> E[SF]"100"
3283  011076  174003                          STF     AC0,AC3                 ;EA[DF]"000" -> E[SF]"011"
3284  011100  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3285  011104  001366                          BNE     59$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3286                                                                          ;
3287  011106  174337  002646                  STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
3288  011112  040437  002646                  BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3289  011116  005037  002650                  CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3290  011122  104426  003164  002646          CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3291  011130  001401                          BEQ     .+4                     ;BR IF AGREE
3292  011132  104052                          ERROR   52                      ;SIGNAL ERROR ... IF NOT
3293                                          ;"ESPAD.A ADDRS ERROR; R[SF]"
3294                                          ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3295                                          ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3296                                                                          ;
3297                                  ;------------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-H'S, BITS<1:0>------------
3298  011134  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3299                                          ;--------ERROR-LOOP-ENTERS-HERE--------
3300  011136  172437  003164          60$:    LDF     FPSEFZ,AC0              ;ONES -> E[DF]"000"
3301  011142  174003                          STF     AC0,AC3                 ;EA[DF]"000" -> E[SF]"011"
3302  011144  170401                          CLRF    AC1                     ;ZEROES -> E[SF]"001"
3303  011146  170402                          CLRF    AC2                     ;ZEROES -> E[SF]"010"
3304  011150  174300                          STF     AC3,AC0                 ;EA[DF]"011" -> E[SF]"000"
3305  011152  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3306  011156  001367                          BNE     60$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3307                                                                          ;
3308  011160  174037  002646                  STF     AC0,MFAC0               ;EB[DF]"000" -> MEMORY
3309  011164  040437  002646                  BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3310  011170  005037  002650                  CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3311  011174  104426  003164  002646          CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3312  011202  001401                          BEQ     .+4                     ;BR IF AGREE
3313  011204  104052                          ERROR   52                      ;SIGNAL ERROR ... IF NOT
3314                                          ;"ESPAD.A ADDRS ERROR; R[SF]"
3315                                          ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3316                                          ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3317                                                                          ;
3318                                  ;------------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-H, BIT<2>-------- ---
3319  011206  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3320                                          ;--------ERROR-LOOP-ENTERS-HERE---- ---
3321  011210  172737  003164          61$:    LDF     FPSEFZ,AC3              ;ONES -> E[DF]"011"
```

```
3322  011214  174300                    STF     AC3,AC0                 ;EA[DF]"011" -> E[SF]"000"
3323  011216  170404                    CLRF    AC4                     ;ZEROES -> E[SF]"100"
3324  011220  174003                    STF     AC0,AC3                 ;EA[DF]"000" -> E[SF]"011"
3325  011222  105737  002645            TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3326  011226  001370                    BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3327                                                                    ;
3328  011230  174337  002646            STF     AC3,MFAC0               ;EB[DF]"011" -> MEMORY
3329  011234  040437  002646            BIC     R4,MFAC0                ;ZERO UNWANTED BITS OF RESULT,
3330  011240  005037  002650            CLR     MFAC0+2                 ;IGNORE FRACTION, WORD.B
3331  011244  104426  003164  002646    CMP32M  ,FPSEFZ,MFAC0           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
3332  011252  001401                    BEQ     .+4                     ;BR IF AGREE
3333  011254  104052                    ERROR   52                      ;SIGNAL ERROR ... IF NOT
3334                                    ;"ESPAD.A ADDRS ERROR; R[SF]"
3335                                    ;       EXPD ACC = EXP'D ACC STORED = (177600,000000)
3336                                    ;       RCVD ACC = RCV'D ACC STORED, ADDRS ERR = (000000,000000)
3337                                                                    ;
3338
3339
3340
3341                    ;;**************************************************************
3342                    ;*TEST 24        EXPNT, (EA=0, EB=0) W/"CMPF"
3343                    ;
3344                    ;       THIS TEST VERIFIES THE EXPNT DATAPATH ESPAD.A[XX]=ZERO AND
3345                    ;       ESPAD.B[XX]=ZERO LOGIC.  THE TEST IS PERFORMED USING THE
3346                    ;       "CMPF" INSTRUCTION, WHICH DOES THE FOLLOWING:
3347                    ;
3348                    ;               CMPXZ:  BUT(EA=0*EB=0)
3349                    ;                                 !
3350                    ;                                 !
3351                    ;       ---------------------------------
3352                    ;       !         !         !         !
3353                    ;       \!/       \!/       \!/       \!/
3354                    ;     CMPXZ.02  CMPXZ.22  CMPXZ.24  CMPXZ.26
3355                    ;    EA*0/EB*0 EA*0/EB=0 EA=0/EB*0 EA=0/EB=0
3356                    ;      (254)     (255)     (256)     (257)
3357                    ;
3358                    ;FP11-E MICROBREAK IS EMPLOYED TO VERIFY THAT THE CORRECT PATH WAS CHOSEN.
3359                    ;DATA, FROM THE TABLE BELOW, RIPPLES A "1" THRU THE SELECTED LOGIC.
3360                    ;
3361                    ;---------
3362                    ;REGISTER/LOCATION USE:
3363                    ;
3364                    ;AC0      -EA[SF] DATA
3365                    ;AC3      -EB[DF] DATA
3366                    ;
3367                    ;MFAC0+   -EA[SF] DATA, IN MEMORY
3368                    ;MFAC1+   -EB[DF] DATA, IN MEMORY
3369                    ;
3370                    ;R0       -RCV'D FPSHI/FEC AFTER EXEC
3371                    ;R3       -EXP'D FP11-E UBREAK TARGET
3372                    ;R4       -TABLE CNTR
3373                    ;R5       -DATA TABLE PTR
3374                    ;
3375                    ;----------
3376                    ;MODULE/ERROR INFO:
3377
```

```
3378                                    ;       FNUA/K8
3379                                    ;         CROM/LATCHES, JREG/BUA, FP.EMIT.E, F.BUS.E/ENABLES/DRIVERS
3380                                    ;
3381                                    ;       FEXP/K9
3382                                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, ECR(EA=0/EB=0)
3383                                    ;
3384                                    ;       FMUL/K10
3385                                    ;         [PREVIOUSLY VERIFIED]
3386                                    ;
3387                                    ;       FALU/K11
3388                                    ;         [PREVIOUSLY VERIFIED]
3389                                    ;
3390                                    ;;*****************************************************************
3391  011256  000004                    TST24:  SCOPE
3392                                    ;
3393  011260  170127  040040                    LDFPS   #040040                 ;INTR-DISAB/F-MODE/TRUNC
3394  011264  105037  002624                    CLRB    FPLENF                  ;SET F-MODE KEY
3395  011270  012705  011402                    MOV     #40$,R5                 ;DATA TABLE PTR.
3396  011274  012704  000022                    MOV     #18.,R4                 ;#TABLE ENTRIES
3397                                    ;
3398                                            ;*DATA TABLE LOOP ENTERS HERE*
3399  011300  005237  002640            10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
3400  011304  012537  002646                    MOV     (R5)+,MFAC0+0           ;GET EA[SF] FOR AC0
3401  011310  005037  002650                    CLR     MFAC0+2                 ;
3402  011314  012537  002656                    MOV     (R5)+,MFAC1+0           ;GET EB[DF] FOR AC3
3403  011320  005037  002660                    CLR     MFAC1+2                 ;
3404  011324  012503                            MOV     (R5)+,R3                ;GET EXP'D UBRK ADDR
3405  011326  104416                            ZAPHFP                          ;RE-INIT HFP
3406  011330  170003                            LDUB                            ;SETUP EXP'D PATH
3407  011332  172437  002645                    LDF     MFAC0,AC0               ;GET OPERANDS, E[SF]
3408  011336  172737  002656                    LDF     MFAC1,AC3               ;AND E[DF]
3409                                    ;
3410  011342  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3411                                            ;--------ERROR-LOOP-ENTERS-HERE--------
3412                                    ;
3413  011344  170127  040060                    LDFPS   #040060                 ;SET FMM=1
3414  011350  173700                    63$:    CMPF    AC0,AC3                 ;EA[SF=0], EB[DF=3]
3415  011352  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3416  011356  001374                            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3417                                    ;
3418  011360  076600  000036                    MFD     RFEC                    ;GET FPSHI#FEC AFTER
3419  011364  020027  140016                    CMP     R0,#140016              ;DID HFP UBREAK?
3420  011370  001401                            BEQ     20$                     ;BR IF YES
3421  011372  104074                            ERROR   74                      ;HFP DIDN'T UBREAK
3422                                            ;"EXPNT EA=0, EB=0 DATAPATH ERR"
3423                                    ;               E-UBRK = EXP'D HFP UBRK TARGET
3424                                    ;               R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER EXEC
3425                                    ;               EA[SF] = EXPNT SF OPERAND
3426                                    ;               EB[DF] = EXPNT DF OPERAND
3427                                    ;
3428                                            ;*NEXT DATA PATTERN*
3429  011374  077437                    20$:    SOB     R4,10$                  ;COUNT & LOOP
3430  011376  104416                            ZAPHFP                          ;ON LEAVING TEST
3431  011400  000466                            BR      TST25           ::      ;NEXT TEST WHEN DONE
3432                                    ;
3433                                    ;
```

```
3434                                    ;
3435                                    ;////////////////////////////////////////////////////////////////////////
3436                                    ;
3437                                    ;       DATA FOR ABOVE TEST:
3438                                    ;
3439                                    ;       EA[SF]  EB[DF] "CMPF"
3440                                    ;       (LDF)   (LDF)   UBRK
3441
3442  011402  077600  000000  000256  40$:    377*S,  000*S,  256     ;EB[DF]=(000)
3443
3444  011410  077600  000200  000254          377*S,  001*S,  254     ;  /!\
3445  011416  077600  000400  000254          377*S,  002*S,  254     ;   !
3446  011424  077600  001000  000254          377*S,  004*S,  254     ;   !
3447  011432  077600  002000  000254          377*S,  010*S,  254     ;RIPPLE-A-1
3448  011440  077600  004000  000254          377*S,  020*S,  254     ;THRU EB[DF]
3449  011446  077600  010000  000254          377*S,  040*S,  254     ;   !
3450  011454  077600  020000  000254          377*S,  100*S,  254     ;   !
3451  011462  077600  040000  000254          377*S,  200*S,  254     ;  \!/
3452
3453  011470  000000  077600  000255          000*S,  377*S,  255     ;EA[SF]=(000)
3454
3455  011476  000200  077600  000254          001*S,  377*S,  254     ;  /!\
3456  011504  000400  077600  000254          002*S,  377*S,  254     ;   !
3457  011512  001000  077600  000254          004*S,  377*S,  254     ;   !
3458  011520  002000  077600  000254          010*S,  377*S,  254     ;RIPPLE-A-1
3459  011526  004000  077600  000254          020*S,  377*S,  254     ;THRU EA[SF]
3460  011534  010000  077600  000254          040*S,  377*S,  254     ;   !
3461  011542  020000  077600  000254          100*S,  377*S,  254     ;   !
3462  011550  040000  077600  000254          200*S,  377*S,  254     ;  \!/
3463
3464
3465
3466                                    ;;*****************************************************************
3467                                    ;*TEST 25       EXPNT, (EA=0.OR.EB=0) W/"MULF"
3468                                    ;
3469                                    ;       THIS TEST VERIFIES THE EXPNT DATAPATH
3470                                    ;
3471                                    ;          ESPAD.A[XX]=ZERO .OR. ESPAD.B[XX]=ZERO
3472                                    ;
3473                                    ;        LOGIC.  THE TEST IS PERFORMED USING THE
3474                                    ;       "MULF" INSTRUCTION, WHICH DOES THE FOLLOWING:
3475                                    ;
3476                                    ;               MULXZ:  BUT(EA.OR.EB=ZERO)
3477                                    ;                                  !
3478                                    ;                                  !
3479                                    ;             ---------------------------------
3480                                    ;             !                                !
3481                                    ;            \!/                              \!/
3482                                    ;          MULFZ.02                         MULFABZ
3483                                    ;         EA#0*EB#0                        EA=0+EB=0
3484                                    ;           (042)                            (043)
3485                                    ;
3486                                    ;       FP11-E MICROBREAK IS EMPLOYED TO VERIFY THAT THE CORRECT PATH WAS CHOSEN.
3487                                    ;       DATA, FROM THE TABLE BELOW, RIPPLES A "1" THRU THE SELECTED LOGIC.
3488                                    ;
3489                                    ;       ---------
```

```
3490
3491                    ;       REGISTER/LOCATION USE:
3492                    ;
3493                    ;       AC0     -EA[SF] DATA
3494                    ;       AC3     -EB[DF] DATA
3495                    ;
3496                    ;       MFAC0+  -EA[SF] DATA, IN MEMORY
3497                    ;       MFAC1+  -EB[DF] DATA, IN MEMORY
3498                    ;
3499                    ;       R0      -RCV'D FPSHI/FEC AFTER EXEC
3500                    ;       R3      -EXP'D FP11-E UBREAK TARGET
3501                    ;       R4      -TABLE CNTR
3502                    ;       R5      -DATA TABLE PTR
3503                    ;
3504                    ;       ----------
3505                    ;       MODULE/ERROR INFO:
3506                    ;
3507                    ;       FNUA/K8
3508                    ;         CROM/LATCHES, JREG/BUA, FP.EMIT.E, F.BUS.E/ENABLES/DRIVERS
3509                    ;
3510                    ;       FEXP/K9
3511                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, ECR(EA=0/EB=0)
3512                    ;
3513                    ;       FMUL/K10
3514                    ;         [PREVIOUSLY VERIFIED]
3515                    ;
3516                    ;       FALU/K11
3517                    ;         [PREVIOUSLY VERIFIED]
3518                    ;
3519  011556  000004    ;.*****************************************************************
                        TST25:  SCOPE
3520                                                            ;
3521  011560  170127  040040           LDFPS   #040040         ;INTR-DISAB/F-MODE/TRUNC
3522  011564  105037  002524           CLRB    FPLENF          ;SET F-MODE KEY
3523  011570  012705  011702           MOV     #40$,R5         ;DATA TABLE PTR
3524  011574  012704  000004           MOV     #4.,R4          ;#TABLE ENTRIES
3525                                                            ;
3526                                   ;*DATA TABLE LOOP ENTERS HERE*
3527  011600  005237  002640   10$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
3528  011604  012537  002646           MOV     (R5)+,MFAC0+0   ;GET EA[SF] FOR AC0
3529  011610  005037  002650           CLR     MFAC0+2         ;
3530  011614  012537  002656           MOV     (R5)+,MFAC1+0   ;GET EB[DF] FOR AC3
3531  011620  005037  002660           CLR     MFAC1+2         ;
3532  011624  012503                   MOV     (R5)+,R3        ;GET EXP'D UBRK ADDR
3533  011626  104416                   ZAPHFP                  ;RE-INIT HFP
3534  011630  170003                   LDUB                    ;SETUP EXP'D PATH
3535  011632  172437  002646           LDF     MFAC0,AC0       ;GET OPERANDS, E[SF]
3536  011636  172737  002656           LDF     MFAC1,AC3       ;AND E[DF]
3537                                                            ;
3538  011642  104406                   ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
3539                                   ;--------ERROR-LOOP-ENTERS-HERE--------
3540                                                            ;
3541  011644  170127  040060           LDFPS   #040060         ;SET FMM=1
3542  011650  171300           63$:    MULF    AC0,AC3         ;EA[SF=0], EB[DF=3]
3543  011652  105737  002645           TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3544  011656  001374                   BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3545                                                            ;
```

```
3546  011660  076600  000036              MED     ,RFEC                   ;GET FPSHI#FEC AFTER
3547  011664  020027  140016              CMP     R0,#140016              ;DID HFP UBRK?
3548  011670  001401                      BEQ     20$                     ;BR IF YES
3549  011672  104075                      ERROR   75                      ;HFP DIDN'T UBRK
3550                                      ;"EXPNT EA=0+EB=0 DATAPATH ERR"
3551                                      ;       E-UBRK = EXP'D HFP UBRK TARGET
3552                                      ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER EXEC
3553                                      ;       EA[SF] = EXPNT SF OPERAND
3554                                      ;       EB[DF] = EXPNT DF OPERAND
3555                                      ;
3556                                      ;*NEXT DATA PATTERN*
3557  011674  077437              20$:    SOB     R4,10$                  ;COUNT & LOOP
3558  011676  104416                      ZAPHFP                          ;ON LEAVING TEST
3559  011700  000414                      BR      TST26           ;;      ;NEXT TEST WHEN DONE
3560                                  ;
3561                                  ;////////////////////////////////////////////////////////////////////////
3562                                  ;
3563                                  ;       DATA FOR ABOVE TEST:
3564                                  ;
3565                                  ;       EA[SF]  EB[DF] "MULF"
3566                                  ;       (LDF)   (LDF)   UBRK
3567
3568  011702  077600  000000  000043  40$:    377*S,  000*S,  043     ;EB[DF] ZERO
3569
3570  011710  077600  077600  000042          377*S,  377*S,  042     ;NEITHER ZERO
3571
3572  011716  000000  077600  000043          000*S,  377*S,  043     ;EA[SF] ZERO
3573
3574  011724  000000  000000  000043          000*S,  000*S,  043     ;BOTH ZERO
3575
3576
3577
3578                              ;*****************************************************************
3579                              ;*TEST 26       EXPNT, (ER=0) W/"ABSF"
3580                              ;
3581                              ;       THIS TEST VERIFIES THE EXPNT DATAPATH
3582                              ;
3583                              ;          ER<7:0> = ZERO
3584                              ;
3585                              ;        LOGIC.  THE TEST IS PERFORMED USING THE
3586                              ;       "ABSF" INSTRUCTION, WHICH DOES THE FOLLOWING:
3587                              ;
3588                              ;               ABSXZ:  BUT(ER=ZERO)
3589                              ;                               !
3590                              ;                               !
3591                              ;            -----------------------------------
3592                              ;            !                                  !
3593                              ;           \!/                                \!/
3594                              ;       ABSXZ.02                           ABSXZ.04
3595                              ;        ER#ZERO                            ER=ZERO
3596                              ;        (032)                              (033)
3597                              ;
3598                              ;       FP11-E MICROBREAK IS EMPLOYED TO VERIFY THAT THE CORRECT PATH WAS CHOSEN.
3599                              ;       DATA, FROM THE TABLE BELOW, RIPPLES A "1" THRU THE SELECTED LOGIC.
3600                              ;
3601                              ;       ---------
```

```
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       AC0     -EA[SF] DATA
                        ;
                        ;       MFAC0+  -EA[SF] DATA, IN MEMORY
                        ;
                        ;       R0      -RCV'D FPSHI/FEC AFTER EXEC
                        ;       R3      -EXP'D FP11-E UBREAK TARGET
                        ;       R4      -TABLE CNTR
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FP.EMIT.E, F.BUS.E/ENABLES/DRIVERS
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>.LOGIC, ECR(ER=0)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
011732  000004          TST26:  SCOPE
                        ;
011734  170127  040040          LDFPS   #040040                 ;INTR-DISAB/F-MODE/TRUNC
011740  105037  002624          CLRB    FPLENF                  ;SET F-MODE KEY
011744  012705  012042          MOV     #40$,R5                 ;PTR TO DATA TABLE
011750  012704  000014          MOV     #12.,R4                 ;#TABLE ENTRIES
                        ;
                        ;*DATA TABLE LOOP ENTERS HERE*
011754  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
011760  012537  002646          MOV     (R5)+,MFAC0+0           ;GET E[SF] FOR AC0
011764  005037  002650          CLR     MFAC0+2                 ;
011770  012503                  MOV     (R5)+,R3                ;GET EXP'D UBRK ADDR
011772  104416                  ZAPHFP                          ;RE-INIT HFP
011774  170003                  LDUB                            ;SETUP EXP'D UBRK ADDR
                        ;
011776  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                        ;--------ERROR-LOOP-ENTERS-HERE--------
                        ;
012000  172437  002646          LDF     MFAC0,AC0               ;GET E[SF] OPERAND
012004  170127  040060          LDFPS   #040060                 ;SET FMM=1
012010  170600          63$:    ABSF    AC0                     ;E[SF=0] INTO ER
012012  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
012016  001374                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                        ;
012020  076600  000036          MFD     ,RFEC                   ;GET FPSHI#FEC AFTER
012024  020027  140016          CMP     R0,#140016              ;DID HFP UBREAK?
012030  001401                  BEQ     20$                     ;BR IF YES
012032  104076                  ERROR   76                      ;HFP DIDN'T UBRK
                        ;"EXPNT ER=0 DATAPATH ERR"
```

```
3658                                            ;       E-UBRK = EXP'D HFP UBRK TARGET
3659                                            ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER EXEC
3660                                            ;       EA[SF] = EXPNT SF OPERAND
3661                                            ;
3662                                            ;*NEXT DATA PATTERN*
3663  012034  077431                   20$:     SOB     R4,10$                  ;COUNT & LOOP
3664  012036  104416                            ZAPHFP                          ;ON LEAVING TEST
3665  012040  000430                            BR      TST27           ;;      ;NEXT TEST WHEN DONE
3666                                            ;
3667                                            ;
3668                                            ;////////////////////////////////////////////////////////////////////////
3669                                            ;
3670                                            ;       DATA FOR ABOVE TEST:
3671                                            ;
3672                                            ;       E[SF]   "ABSF"
3673                                            ;       (LDF)   UBRK
3674
3675  012042  000000  000033           40$:     000*S,  033     ;ZERO
3676
3677  012046  000200  000032                    001*S,  032     ;
3678  012052  000400  000032                    002*S,  032     ;
3679  012056  001000  000032                    004*S,  032     ;
3680
3681  012062  000000  000033                    000*S,  033     ;ZERO
3682
3683  012066  002000  000032                    010*S,  032     ;
3684  012072  004000  000032                    020*S,  032     ;
3685  012076  010000  000032                    040*S,  032     ;
3686
3687  012102  000000  000033                    000*S,  033     ;ZERO
3688
3689  012106  020000  000032                    100*S,  032     ;
3690  012112  040000  000032                    200*S,  032     ;
3691
3692  012116  000000  000033                    000*S,  033     ;ZERO
3693
3694
3695
3696                                            ;;**************************************************************
3697                                            ;*TEST 27        EXPNT, EALU ADD/CARRY LOGIC W/"MULF"
3698                                            ;
3699                                            ;       THIS TEST RUNS DATA PATTERNS THRU THE EXPONENT ALJ
3700                                            ;       TO CHECK ITS ABILITY TO PERFORM THE "ADD" FUNCTION.
3701                                            ;       THE "MULF" INSTRUCTION FLOW IS UTILIZED, BUT IT IS ABORTED
3702                                            ;       DURING MICROSTATE "MULFZ.04", SO THE STORED EXPONENT SUM
3703                                            ;       IS **NOT** RE-COMPENSATED BY -(200) AFTER THE ADDITION.
3704                                            ;
3705                                            ;         THE RESULT  "E[DF] <- EA[DF]-PLUS-EB[SF]"
3706                                            ;
3707                                            ;       IS THE ACTUAL VALUED STORED IN THE DESTINATION ACC.
3708                                            ;
3709                                            ;       ----------
3710                                            ;       REGISTER/LOCATION USE:
3711                                            ;
3712                                            ;       MFAC0+  -EA[DF] EXPNT OPERAND-A
3713                                            ;       MFAC1+  -EB[SF] EXPNT OPERAND-B
```

```
                        ;       MFAC2+  -EXPECTED EA-PLUS-EB EXPNT SUM
                        ;       MFAC3+  -RECEIVED EA-PLUS-EB EXPNT SUM FROM HFP
                        ;
                        ;       AC0     -EA[DF] EXPNT, RECEIVED SUM
                        ;       AC1     -EA[DF] TEMP.
                        ;       AC3     -EB[SF] EXPNT OPERAND
                        ;
                        ;       R0      -RCV'D FPSHI#FEC
                        ;       R3      -HFP UBRK ADDRESS, "MULFZ.04"=(200)
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, ~IR.SF/DF, F.BUS.E/ENABLES/DRIVERS
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.A/B,
                        ;         ESPAD.A/B.ADDR, EALU.DATA/CNTL(A-PLUS-B/CARRY)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
012122  000004          TST27:  SCOPE
                        ;
012124  170127  040000          LDFPS   #040000                 ;INTR-DISAB/F-MODE
012130  012705  012310          MOV     #40$,R5                 ;DATA TABLE PTR
012134  105037  002624          CLRB    FPLENF                  ;F-MODE KEY
012140  005037  002650          CLR     MFAC0+2                 ;WORD-B IS ZEROES, FOR TYPEOUT
012144  005037  002660          CLR     MFAC1+2                 ;
012150  005037  002670          CLR     MFAC2+2                 ;
                        ;
                        ;*DATA LOOP ENTERS HERE*
012154  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
012160  012537  002646          MOV     (R5)+,MFAC0+0           ;GET EA[DF]
012164  100447                  BMI     39$                     ;IF <0, DONE
012166  012537  002656          MOV     (R5)+,MFAC1+0           ;GET EB[SF]
012172  012537  002666          MOV     (R5)+,MFAC2+0           ;GET EXP'D EA[DF]+EB[SF]
012176  172537  002646          LDF     MFAC0,AC1               ;SETUP EA[DF], TEMP
012202  172737  002656          LDF     MFAC1,AC3               ;SETUP EB[SF]
                        ;
012206  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                        ;--------ERROR-LOOP-ENTERS-HERE--------
                        ;
012210  104416                  ZAPHFP                          ;INIT HFP, SET FEC=(377)
012212  012703  000200          MOV     #200,R3                 ;SETUP MULFZ.04 MICROADDR
012216  170003                  LDUB                            ;RESET UBRK IF ALTERED
012220  170127  040020          LDFPS   #040020                 ;INTR-DISAB/F-MODE/FMM=1
012224  174100          63$:    STF     AC1,AC0                 ;COPY TEMP. TO EA[DF]
012226  171003                  MULF    AC3,AC0                 ;FORM  E[DF] <- EA[DF]-PLUS-EB[SF]
                        ;                                       ;  [ABORT @ MULFZ.04]
```

```
3770  012230  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3771  012234  001373                            BNE     63$                     ;  WITH/ LINE CLOCK OFF, & FPS(FID=1/FMM=0)
3772                                                                            ;
3773  012236  076600  000036                    MED     ,RFEC                   ;GET R0=FPSHI#FEC AFTER INSTR
3774  012242  174037  002676                    STF     AC0,MFAC3               ;STORE ANSWER
3775  012246  042737  000177  002676            BIC     #177,MFAC3+0            ;IGNORE FRACTION PORTION
3776  012254  005037  002700                    CLR     MFAC3+2                 ;
3777                                                                            ;
3778  012260  020027  140016                    CMP     R0,#140016              ;DID HFP ABORT VIA UBRK ??
3779  012264  001401                            BEQ     15$                     ;BR IF YES
3780  012266  104053                            ERROR   53                      ;NO - HFP SEQUENCING ERROR
3781                                            ;"EXPNT EALU EA+EB MULF/SEQ ERR"
3782                                            ;       E-UBRK = EXP'D HFP UBRK TARGET
3783                                            ;       R-FPSHIFEC = RCV'D FPSHI/FEC, EXP'D TO BE (140016)
3784                                            ;       EA[DF] = EXPNT DF OPERAND
3785                                            ;       EB[SF] = EXPNT SF OPERAND
3786                                            ;       EXPD EA+EB = EXPECTED EA+EB EXPNT
3787                                            ;       RCVD EA+EB = RECEIVED EA+EB EXPNT
3788                                                                            ;
3789  012270  023737  002666  002676  15$:      CMP     MFAC2+0,MFAC3+0         ;(EXP'D A+B) = (RCV'D A+B) ???
3790  012276  001726                            BEQ     10$                     ;BR IF OK
3791  012300  104077                            ERROR   77                      ;ELSE EXPNT-ALU/A+B ERROR
3792                                            ;"EXPNT EALU EA-PLUS-EB RESULT ERR"
3793                                            ;       E-UBRK = EXP'D HFP UBRK TARGET
3794                                            ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER EXEC
3795                                            ;       EA[DF] = EXPNT DF OPERAND
3796                                            ;       EB[SF] = EXPNT SF OPERAND
3797                                            ;       EXPD EA+EB = EXPECTED EA+EB EXPNT
3798                                            ;       RCVD EA+EB = RECEIVED EA+EB EXPNT
3799                                                                            ;
3800  012302  000724                            BR      10$                     ;MORE
3801                                                                            ;
3802                                            ;*DONE WITH THIS TEST*
3803  012304  104416                    39$:    ZAPHFP                          ;CLEAR THE WORLD
3804  012306  000445                            BR      TST30           ;;      ;NEXT TEST
3805
3806                                            ;
3807                                            ;////////////////////////////////////////////////////////////////////////
3808
3809                                            ;       DATA FOR ABOVE TEST:
3810                                            ;
3811                                            ;       OPND-A  OPND-B  ANSWER  ; ESPAD.A[DF]  +  ESPAD.B[SF] =  ESPAD[DF]
3812                                            ;       ------  ------  ------  ;-------------- -------------- -------------
3813
3814  012310  025200  025200  052400  40$:      025200, 025200, 052400  ;(00.0101.0101)+(00.0101.0101)=(00.1010.1010)
3815
3816  012316  052400  052400  025000            052400, 052400, 025000  ;(00.1010.1010)+(00.1010.1010)=(01.0101.0100)
3817
3818  012324  052400  025200  077600            052400, 025200, 077600  ;(00.1010.1010)+(00.0101.0101)=(00.1111.1111)
3819
3820  012332  025200  052400  077600            025200, 052400, 077600  ;(00.0101.0101)+(00.1010.1010)=(00.1111.1111)
3821
3822  012340  041600  041600  003400            041600, 041600, 003400  ;(00.1000.0111)+(00.1000.0111)=(01.0000.1110)
3823
3824  012346  036000  036000  074000            036000, 036000, 074000  ;(00.0111.1000)+(00.0111.1000)=(00.1111.0000)
3825
```

```
3826  012354  022600  060600  003400          022600, 060600, 003400  ;(00.0100.1011)+(00.1100.0011)=(01.0000.1110)
3827
3828  012362  055000  017000  074000          055000, 017000, 074000  ;(00.1011.0100)+(00.0011.1100)=(00.1111.0000)
3829
3830  012370  051200  032200  003400          051200, 032200, 003400  ;(00.1010.0101)+(00.0110.1001)=(01.0000.1110)
3831
3832  012376  026400  045400  074000          026400, 045400, 074000  ;(00.0101.1010)+(00.1001.0110)=(00.1111.0000)
3833
3834  012404  026400  055000  003400          026400, 055000, 003400  ;(00.0101.1010)+(00.1011.0100)=(01.0000.1110)
3835
3836  012412  051200  022600  074000          051200, 022600, 074000  ;(00.1010.0101)+(00.0100.1011)=(00.1111.0000)
3837
3838  012420  100000                          100000  ;<END>
3839
3840
3841
3842                                  ;;*****************************************************************
3843                                  ;*TEST 30       EXPNT, COUNTER/PRE-SHFT-QUOT WITH "MAS"
3844                                  ;
3845                                  ;       THIS TEST VERIFIES THE EXPONENT:
3846                                  ;
3847                                  ;               PRESHIFT-QUOTIENT ROM (OUTPUT TO COUNTER)
3848                                  ;
3849                                  ;               AND "COUNTER" AND BUT(COUNT)
3850                                  ;
3851                                  ;       FOR VALUES IN THE "ER" OF (000) TO (077).
3852                                  ;
3853                                  ;       THE TEST DOES THE FOLLOWING, BY USING THE "MAS" INSTRUCTION:
3854                                  ;
3855                                  ;         1) E[AC1] <- (000)
3856                                  ;
3857                                  ;         2) CNTR <- PRESHIFT-QUOT-ROM( ER<5:0> )
3858                                  ;
3859                                  ;         3) BUT(CNT) UNTIL CNTR=(17), LOOP: E[AC1] <- E[AC1]-PLUS-(001)
3860                                  ;
3861                                  ;       ---------
3862                                  ;       REGISTER/LOCATION USE:
3863                                  ;
3864                                  ;       AC0     -"MAS" ER<5:0> INPUT VALUE IN EXPNT
3865                                  ;       AC2     -"MAS" CNTR/PRESHFT-CNTR ROM OUTPUT IN EXPNT
3866                                  ;
3867                                  ;       R0      -EXP'D EXPNT (AFTER STEXP) IN AC2/EXPNT
3868                                  ;       R2      -RCV'D EXPNT (AFTER STEXP) FROM AC2/EXPNT
3869                                  ;       R3      -ER<5:0> INPUT CNTR, (077)->(000)
3870                                  ;
3871                                  ;       ----------
3872                                  ;       MODULE/ERROR INFO:
3873                                  ;
3874                                  ;       FNUA/K8
3875                                  ;         CROM/LATCHES, JREG/BUA
3876                                  ;
3877                                  ;       FEXF/K9
3878                                  ;         CROM/LATCHES, BUT<2:0>.LOGIC, ESPAD.A/B,
3879                                  ;         ESPAD.A/B.ADDR, EALU.DATA/CNTL, QUOT.ROM/CNTR
3880                                  ;
3881                                  ;       FMUL/K10
```

```
3882                                    ;         [PREVIOUSLY VERIFIED]
3883                                    ;
3884                                    ;       FALU/K11
3885                                    ;         [PREVIOUSLY VERIFIED]
3886                                    ;
3887                                    ;*****************************************************************
3888  012422  000004                    TST30:  SCOPE
3889                                                                    ;
3890  012424  170127  040240                    LDFPS   #040240       ;INTR-DISAB/D-MODE/TRUNC
3891  012430  012703  000077                    MOV     #077,R3       ;ER<5:0> CNTR, (77)->(00)
3892                                                                    ;
3893                                            ;*DATA LOOP ENTERS HERE*
3894  012434  005237  002640            10$:    INC     DWLOOP        ;BUMP CLOCK IN.A.LOOP COUNT
3895  012440  176403                            LDEXP   R3,AC0        ;R3<5:0> -> E[AC0]<5=0>
3896  012442  010301                            MOV     R3,R1         ;
3897  012444  005000                            CLR     R0            ;GET R0=EXP'D E[AC2]
3898  012446  071027  000013                    DIV     #11.,R0       ;
3899  012452  005200                            INC     R0            ;
3900                                                                    ;
3901  012454  104406                            ERRPNT                ;DONT CHANGE DATA IN ERROR LOOP
3902                                            ;--------ERROR-LOOP-ENTERS-HERE--------
3903                                                                    ;
3904  012456  170402                            CLRD    AC2           ;ZAP EXPNT BEFORE
3905  012460  170007                    63$:    MAS                   ;E[AC0] -> PRE-SHFT-QUOT-ROM
3906                                                                    ;  -> CNTR -> INC(E[AC2])
3907  012462  105737  002645                    TSTB    LPTITE        ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3908  012466  001374                            BNE     63$           ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
3909                                                                    ;
3910  012470  175202                            STEXP   AC2,R2        ;GET EXPNT AFTER
3911  012472  062702  000200                    ADD     #200,R2       ;COMPENSATE FOR STEXP ADJUSTMENT
3912  012476  020002                            CMP     R0,R2         ;(EXP'D) = (RCV'D)??
3913  012500  001401                            BEQ     20$           ;BR IF AGREE
3914  012502  104100                            ERROR   100           ;ELSE ERROR
3915                                            ;"EXPNT CNTR/PRE-SHFT-QUOT-ROM ERR"
3916                                            ;       ER<5:0> = ER VALUE, INPUT TO QUOT-ROM
3917                                            ;       E-CNTR/EXPNT = EXP'D VALUE OUTPUT FROM CNTR LOOP
3918                                            ;       R-CNTR/EXPNT = RCV'D VALUE, FROM ABOVE
3919                                                                    ;
3920                                            ;*NEXT ER VALUE*
3921  012504  005303                    20$:    DEC     R3            ;COUNT DOWN
3922  012506  002352                            BGE     10$           ;LOOP ON (77)->(00)
3923
3924
3925
```

```
3926                    ;;*****************************************************************
3927                    ;*TEST 31       IFORK[(ADD+SUB)*MO], SUMPATH/MO*R(6+7) DECODE
3928                    ;
3929                    ;       THIS TEST EXERCISES THE IFORK[(ADD+SUB)*MODEO] LOGIC OF THE
3930                    ;       HFP INSTRUCTION DECODE.  SPECIFICALLY TESTED IS THE "SUMPATH"
3931                    ;       DECODE LOGIC, WHICH IS BIT<4> OF THE TARGET MICROADDRESS.
3932                    ;       MODE-0*REG(6/7) IS ALSO EMPLOYED, TO CHECK THAT TARGET BITS<2:0>
3933                    ;       ARE FORCED TO "111".
3934                    ;
3935                    ;       A SUMMARY OF THE TESTS IS AS FOLLOWS.
3936                    ;
3937                    ;       UBRK    SUMPATH    ADD[L]/SUB[H]    SS-XOR-SD
3938                    ;       ----    -------    -------------    ---------
3939                    ;       147        L          ADD  L         H [L,H]
3940                    ;       167        H          ADD  L         L [L,L]
3941                    ;       147        L          SUB  H         L [H,H]
3942                    ;       167        H          SUB  H         H [H,L]
3943                    ;
3944                    ;       NOTE THAT BOTH EA[DF] AND EB[SF] = (000), AND MODE-0*REG(6)
3945                    ;       IS EMPLOYED.  THIS EFFECTIVELY DISABLES THE RANGE CODE ROM FOR THIS
3946                    ;       TEST [IE, TARGET BITS<3:0>="0111"].
3947                    ;
3948                    ;
3949                    ;       ----------
3950                    ;       REGISTER/LOCATION USE:
3951                    ;
3952                    ;       MFAC0+  -COPY OF AC0/AC1
3953                    ;       MFAC1+  -COPY OF AC3/AC6
3954                    ;
3955                    ;       AC0     -(TEMP) OF AC1
3956                    ;       AC1     -SD SIGN BIT, EXPNT=(000)
3957                    ;       AC3     -SS SIGN BIT, EXPNT=(000)
3958                    ;       AC6     -COPY OF AC3
3959                    ;
3960                    ;       R0      -RCV'D FPSHI/FEC AFTER "ADDF/SUBF" INSTR
3961                    ;       R3      -TARGET MICROADDRESS EXP'D (147/167)
3962                    ;
3963                    ;       ----------
3964                    ;       MODULE/ERROR INFO:
3965                    ;
3966                    ;       FNUA/K8
3967                    ;         CROM/LATCHES, JREG/BUA, SUMPATH, IFORK.DECODE
3968                    ;
3969                    ;       FEXP/K9
3970                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, SSPAD.A/B, SS/SD.LOGIC,
3971                    ;         EXPNT.RANGE.CODE-LOGIC
3972                    ;
3973                    ;       FMUL/K10
3974                    ;         [PREVIOUSLY VERIFIED]
3975                    ;
3976                    ;       FALU/K11
3977                    ;         [PREVIOUSLY VERIFIED]
3978                    ;
3979                    ;*****************************************************************
3980  012510  000004    TST31:  SCOPE
3981                    ;
```

```
3982  012512  105037  002624                CLRB    FPLENF                  ;F-MODE KEY
3983                                        ;
3984                                        ;-------SUMPATH[L]--UBRK(147)--"ADDF"[L]--SS[L].XOR.SD[H]=[H]-------
3985  012516  104416                        ZAPHFP                          ;INIT TO HFP, SET FEC=(377)
3986  012520  012703  000147                MOV     #147,R3                 ;HFP TARGET ADDRESS
3987  012524  170003                        LDUB                            ;INTO UBRK
3988  012526  172537  003002                LDF     FPMOAP,AC1              ;E[1,6] <- (000), S[1,6] <- SD <- 1
3989  012532  174137  002646                STF     AC1,MFAC0               ;SAVE IN MEMORY
3990  012536  172737  003060                LDF     FPZERO,AC3              ;E[3,6] <- (000), S[3,6] <- SS <- 0
3991  012542  174337  002656                STF     AC3,MFAC1               ;SAVE IN MEMORY
3992  012546  170127  040020                LDFPS   #040020                 ;INTR-DISABL/F-MODE/FMM=1
3993  012552  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
3994                                        ;--------ERROR-LOOP-ENTERS-HERE--------
3995  012554  174100              60$:      STF     AC1,AC0                 ;COPY DEST. ACC
3996  012556  172006                        ADDF    AC6,AC0                 ;SF=MO*R6, ER <- EA[DF]-EB[SF]
3997                                                                        ;SD <- S[DF], SS <- S[SF]
3998  012560  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
3999  012564  001373                        BNE     60$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4000                                        ;
4001  012566  076600  000036                MED     ,RFEC                   ;GET FPSHI#FEC AFTER
4002  012572  020027  140016                CMP     R0,#140016              ;DID HFP UBREAK ??
4003  012576  001401                        BEQ     .+4                     ;BR IF YES
4004  012600  104101                        ERROR   101                     ;ELSE SIGNAL ERROR, WRONG PATH
4005                                        ;"IFORK/(ADD+SUB)*MO SUMPATH/MO*R6 ERR"
4006                                        ;       E-UBRK = EXP'D HFP UBREAK TARGET
4007                                        ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4008                                        ;       SD/EA[DF] = SD SIGN BIT, EA=(000)
4009                                        ;       SS/EB[SF] = SS SIGN BIT, EB=(000)
4010                                        ;
4011                                        ;
4012                                        ;-------SUMPATH[H]--UBRK(167)--"ADDF"[L]--SS[L].XOR.SD[L]=[L]-------
4013  012602  104416                        ZAPHFP                          ;INIT TO HFP, SET FEC=(377)
4014  012604  012703  000167                MOV     #167,R3                 ;HFP TARGET ADDRESS
4015  012610  170003                        LDUB                            ;INTO UBRK
4016  012612  172537  003060                LDF     FPZERO,AC1              ;E[1,6] <- (000), S[1,6] <- SD <- 0
4017  012616  174137  002646                STF     AC1,MFAC0               ;SAVE IN MEMORY
4018  012622  172737  003060                LDF     FPZERO,AC3              ;E[3,6] <- (000), S[3,6] <- SS <- 0
4019  012626  174337  002656                STF     AC3,MFAC1               ;SAVE IN MEMORY
4020  012632  170127  040020                LDFPS   #040020                 ;INTR-DISABL/F-MODE/FMM=1
4021  012636  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4022                                        ;--------ERROR-LOOP-ENTERS-HERE--------
4023  012640  174100              61$:      STF     AC1,AC0                 ;COPY DEST. ACC
4024  012642  172006                        ADDF    AC6,AC0                 ;SF=MO*R6, ER <- EA[DF]-EB[SF]
4025                                                                        ;SD <- S[DF], SS <- S[SF]
4026  012644  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4027  012650  001373                        BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4028                                        ;
4029  012652  076600  000036                MED     ,RFEC                   ;GET FPSHI#FEC AFTER
4030  012656  020027  140016                CMP     R0,#140016              ;DID HFP UBREAK ??
4031  012662  001401                        BEQ     .+4                     ;BR IF YES
4032  012664  104101                        ERROR   101                     ;ELSE SIGNAL ERROR, WRONG PATH
4033                                        ;"IFORK/(ADD+SUB)*MO SUMPATH/MO*R6 ERR"
4034                                        ;       E-UBRK = EXP'D HFP UBREAK TARGET
4035                                        ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4036                                        ;       SD/EA[DF] = SD SIGN BIT, EA=(000)
4037                                        ;       SS/EB[SF] = SS SIGN BIT, EB=(000)
```

```
4038                                                    ;
4039                                                    ;
4040                                    ;-------SUMPATH[L]--UBRK(147)--"SUBF"[H]--SS[H].XOR.SD[H]=[L]-------
4041  012666  104416                            ZAPHFP                  ;INIT TO HFP, SET FEC=(377)
4042  012670  012703  000147                    MOV     #147,R3         ;HFP TARGET ADDRESS
4043  012674  170003                            LDUB                    ;INTO UBRK
4044  012676  172537  003002                    LDF     FPMOAP,AC1      ;E[1,6] <- (000), S[1,6] <- SD <- 1
4045  012702  174137  002646                    STF     AC1,MFAC0       ;SAVE IN MEMORY
4046  012706  172737  003002                    LDF     FPMOAP,AC3      ;E[3,6] <- (000), S[3,6] <- SS <- 1
4047  012712  174337  002656                    STF     AC3,MFAC1       ;SAVE IN MEMORY
4048  012716  170127  040020                    LDFPS   #040020         ;INTR-DISABL/F-MODE/FMM=1
4049  012722  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
4050                                            ;--------ERROR-LOOP-ENTERS-HERE--------
4051  012724  174100                    62$:    STF     AC1,AC0         ;COPY DEST. ACC
4052  012726  173006                            SUBF    AC6,AC0         ;SF=MO*R6, ER <- EA[DF]-EB[SF]
4053                                                                    ;SD <- S[DF], SS <- S[SF]
4054  012730  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4055  012734  001373                            BNE     62$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4056                                                                    ;
4057  012736  076600  000036                    MED     ,RFEC           ;GET FPSHI*FEC AFTER
4058  012742  020027  140016                    CMP     R0,#140016      ;DID HFP UBREAK ??
4059  012746  001401                            BEQ     .+4             ;BR IF YES
4060  012750  104101                            ERROR   101             ;ELSE SIGNAL ERROR, WRONG PATH
4061                                            ;"IFORK/(ADD+SUB)*MO SUMPATH/MO*R6 ERR"
4062                                            ;       E-UBRK = EXP'D HFP UBREAK TARGET
4063                                            ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4064                                            ;       SD/EA[DF] = SD SIGN BIT, EA=(000)
4065                                            ;       SS/EB[SF] = SS SIGN BIT, EB=(000)
4066                                                    ;
4067                                                    ;
4068                                    ;-------SUMPATH[H]--UBRK(167)--"SUBF"[H]--SS[H].XOR.SD[L]=[H]-------
4069  012752  104416                            ZAPHFP                  ;INIT TO HFP, SET FEC=(377)
4070  012754  012703  000167                    MOV     #167,R3         ;HFP TARGET ADDRESS
4071  012760  170003                            LDUB                    ;INTO UBRK
4072  012762  172537  003060                    LDF     FPZERO,AC1      ;E[1,6] <- (000), S[1,6] <- SD <- 0
4073  012766  174137  002646                    STF     AC1,MFAC0       ;SAVE IN MEMORY
4074  012772  172737  003002                    LDF     FPMOAP,AC3      ;E[3,6] <- (000), S[3,6] <- SS <- 1
4075  012776  174337  002656                    STF     AC3,MFAC1       ;SAVE IN MEMORY
4076  013002  170127  040020                    LDFPS   #040020         ;INTR-DISABL/F-MODE/FMM=1
4077  013006  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
4078                                            ;--------ERROR-LOOP-ENTERS-HERE--------
4079  013010  174100                    63$:    STF     AC1,AC0         ;COPY DEST. ACC
4080  013012  173006                            SUBF    AC6,AC0         ;SF=MO*R6, ER <- EA[DF]-EB[SF]
4081                                                                    ;SD <- S[DF], SS <- S[SF]
4082  013014  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4083  013020  001373                            BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4084                                                                    ;
4085  013022  076600  000036                    MED     ,RFEC           ;GET FPSHI*FEC AFTER
4086  013026  020027  140016                    CMP     R0,#140016      ;DID HFP UBREAK ??
4087  013032  001401                            BEQ     .+4             ;BR IF YES
4088  013034  104101                            ERROR   101             ;ELSE SIGNAL ERROR, WRONG PATH
4089                                            ;"IFORK/(ADD+SUB)*MO SUMPATH/MO*R6 ERR"
4090                                            ;       E-UBRK = EXP'D HFP UBREAK TARGET
4091                                            ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4092                                            ;       SD/EA[DF] = SD SIGN BIT, EA=(000)
4093                                            ;       SS/EB[SF] = SS SIGN BIT, EB=(000)
```

```
4094
4095  013036  104416                    ZAPHFP                          ;INIT HFP
4096
4097
4098
4099                                ;;*****************************************************************
4100                                ;*TEST 32       IFORK[(ADD+SUB)*MO], EXPNT(A+B)=ZERO DECODE
4101                                ;
4102                                ;       THIS TEST CHECKS THE EA[XX]=ZERO & EB[XX]=ZERO DETECTION
4103                                ;       LOGIC OF THE IFORK/RITE DECODE.  THIS LOGIC FORCES THE
4104                                ;       RANGE.CODE ROM TO BE DISABLED, AND OUTPUT CODE="000".
4105                                ;
4106                                ;       ----------
4107                                ;       REGISTER/LOCATION USE:
4108                                ;
4109                                ;       AC0     -EA[DF] EXPNT VALUE, (000) & (377)
4110                                ;       AC1     -(TEMP) OF AC0
4111                                ;       AC2     -EB[SF] EXPNT VALUE, (000) & (377), APPROX. AC5
4112                                ;       AC4     -EB[SF] EXPNT VALUE, (000) & (377), APPROX. AC6
4113                                ;
4114                                ;       R0      -RCV'D FPSHI/FEC AFTER EXEC
4115                                ;       R3      -TARGET MICROADDRESS EXP'D, (160/170)
4116                                ;
4117                                ;       ----------
4118                                ;       MODULE/ERROR INFO:
4119                                ;
4120                                ;       FNUA/K8
4121                                ;         CROM/LATCHES, JREG/BUA, IFORK.DECODE
4122                                ;
4123                                ;       FEXP/K9
4124                                ;         CROM/LATCHES, BUT<2:0>.LOGIC,
4125                                ;         EXPNT.RANGE.CODE-LOGIC, ECR(EA=0/EB=0)
4126                                ;
4127                                ;       FMUL/K10
4128                                ;         [PREVIOUSLY VERIFIED]
4129                                ;
4130                                ;       FALU/K11
4131                                ;         [PREVIOUSLY VERIFIED]
4132                                ;
4133                                ;*****************************************************************
4134  013040  000004                TST32:  SCOPE
4135                                ;
4136                                ;-------EA[DF]=(000)--EB[SF]=(377)--ER(8)="1"-------
4137  013042  104416                        ZAPHFP                          ;INIT TO HFP, SET FEC=(377)
4138  013044  012703  000170                MOV     #170,R3                 ;HFP TARGET ADDRESS
4139  013050  170003                        LDUB                            ;INTO UBRK
4140  013052  172527  000000                LDF     #000000,AC1             ;EA[DF] <- (000)
4141  013056  172627  044230                LDF     #077600,AC2             ;EB[SF] <- (377)
4142  013062  170127  040020                LDFPS   #040020                 ;INTR-DISABL/F-MODE/FMM=1
4143  013066  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4144                                        ;--------ERROR-LOOP-ENTERS-HERE--------
4145  013070  174100                60$:    STF     AC1,AC0                 ;COPY DEST. ACC
4146  013072  172002                        ADDF    AC2,AC0                 ;ER <- EA[DF]=(000) - EB[SF]=(377)
4147  013074  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4148  013100  001373                        BNE     60$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4149                                ;
```

```
4150  013102  076600  000036          MFD     @RFEC                   ;GET FPSHI#FEC AFTER
4151  013106  020027  140016          CMP     R0,#140016              ;DID HFP UBREAK ??
4152  013112  001401                  BEQ     .+4                     ;BR IF YES
4153  013114  104102                  ERROR   102                     ;ELSE SIGNAL ERROR, WRONG PATH
4154                                  ;"IFORK/(ADD+SUB)*MO [EA+EB]=0/MO*R6 ERR"
4155                                  ;       E-UBRK = EXP'D HFP UBREAK TARGET
4156                                  ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4157                                  ;
4158
4159                          ;-------EA[DF]=(377)--EB[SF]=(000)--ER(8)="0"-------
4160  013116  104416                  ZAPHFP                          ;INIT TO HFP, SET FEC=(377)
4161  013120  012703  000160          MOV     #160,R3                 ;HFP TARGET ADDRESS
4162  013124  170003                  LDUB                            ;INTO UBRK
4163  013126  172527  044230          LDF     #077600,AC1             ;EA[DF] <- (377)
4164  013132  172627  000000          LDF     #000000,AC2             ;EB[SF] <- (000)
4165  013136  174204                  STF     AC2,AC4                 ;ACTUAL EB[SF]
4166  013140  170127  040020          LDFPS   #040020                 ;INTR-DISABL/F-MODE/FMM=1
4167  013144  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4168                                  ;--------ERROR-LOOP-ENTERS-HERE--------
4169  013146  174100          61$:    STF     AC1,AC0                 ;COPY DEST. ACC
4170  013150  172004                  ADDF    AC4,AC0                 ;ER <- EA[DF]=(377) - EB[SF]=(000)
4171  013152  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4172  013156  001373                  BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4173                                  ;
4174  013160  076600  000036          MFD     @RFEC                   ;GET FPSHI#FEC AFTER
4175  013164  020027  140016          CMP     R0,#140016              ;DID HFP UBREAK ??
4176  013170  001401                  BEQ     .+4                     ;BR IF YES
4177  013172  104102                  ERROR   102                     ;ELSE SIGNAL ERROR, WRONG PATH
4178                                  ;"IFORK/(ADD+SUB)*MO [EA+EB]=0/MO*R6 ERR"
4179                                  ;       E-UBRK = EXP'D HFP UBREAK TARGET
4180                                  ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
4181                                  ;
4182
4183
4184
4185                  ;;*****************************************************************
4186                  ;*TEST 33        IFORK[(ADD+SUB)*MO], EXPNT.RANGE.CODE ROM CONTENTS
4187                  ;
4188                  ;       THIS TEST VARIES THE EXPNT/"ER"<8:0> VALUE THRU ITS FULL
4189                  ;       RANGE TO VERIFY THE CONTENTS OF THE EXPNT.RANGE.CODE ROM, AND
4190                  ;       ITS ASSOCIATED GATING LOGIC.  "SUMPATH-H" IS HELD CONSTANT AT "H".
4191                  ;
4192                  ;       ----------
4193                  ;       REGISTER/LOCATION USE:
4194                  ;
4195                  ;       MFAC0+  -EB[SF] IN MEMORY
4196                  ;       MFAC1+  -EA[DF] IN MEMORY
4197                  ;
4198                  ;       AC0     -EB[SF] IN ACC, TO GENERATE ER-VALUE
4199                  ;       AC1     -EA[DF] IN ACC, TO GENERATE ER-VALUE
4200                  ;       AC3     -(TEMP) FOR TEST, COPY OF AC1
4201                  ;
4202                  ;       R0      -RCV'D "FPSHI#FEC" AFTER "ADDF" INSTR. EXEC
4203                  ;       R1      -ER<8:0> CNTR, (000)->(777)
4204                  ;       R2      -FPS DURING TEST (F/D-MODES, FID=1, FMM=1)
4205                  ;       R3      -EXPECTED TARGET MICROADDRESS, FOR LDUB
```

```
4206
4207                                    ;       FPLENF  -(000)=F-MODE/(377)=D-MODE FLAG
4208                                    ;
4209                                    ;       ----------
4210                                    ;       MODULF/ERROR INFO:
4211                                    ;
4212                                    ;       FNUA/K8
4213                                    ;         CROM/LATCHES, JREG/BUA, IFORK.DECODE
4214                                    ;
4215                                    ;       FEXP/K9
4216                                    ;         CROM/LATCHES, BUT<2:0>.LOGIC, EALU.DATA/CNTL(A-MINUS-B),
4217                                    ;         EXPNT.RANGE.CODE-LOGIC
4218                                    ;
4219                                    ;       FMUL/K10
4220                                    ;         [PREVIOUSLY VERIFIED]
4221                                    ;
4222                                    ;       FALU/K11
4223                                    ;         [PREVIOUSLY VERIFIED]
4224                                    ;
4225                                    ;.*****************************************************************
4226  013174  000004                    TST33:  SCOPE
4227                                                                    ;
4228  013176  012737  000036  001342            MOV     #30.,$TIMES     ;30. ITER. OF THIS TEST
4229  013204  105037  002624                    CLRB    FPLENF          ;SET F-MODE KEY
4230  013210  012702  040020                    MOV     #040020,R2      ;F-MODE & FMM=1 FPS
4231                                                                    ;
4232                                            ;*LOOP ON F/D-MODE ENTERS HERE*
4233  013214  005001                    1$:     CLR     R1              ;SET ER<8:0>=000 TO START LOOP
4234                                                                    ;
4235                                            ;*DATA LOOP ON ER<8:0> ENTERS HERE*
4236  013216  005237  002640            2$:     INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
4237  013222  004737  013334                    JSR     PC,GTEAEB       ;CALCULATE "EA" & "EB" REQ'D TO GENERATE
4238                                                                    ;  THIS "ER" FROM "ER=EA-MINUS-EB"
4239  013226  103426                            BCS     35$             ;IF SET, CAN'T DO THIS "ER" VALUE
4240                                                                    ;
4241  013230  004737  013440                    JSR     PC,RNGCOD       ;GET R3 = EXP'D UBRK ADDR FOR
4242                                                                    ;  THIS "ER"/RANGE CODE
4243  013234  104416                            ZAPHFP                  ;INIT TO HFP, SET FEC=(377)
4244  013236  170001                            SETF                    ;F-MODE
4245  013240  170003                            LDUB                    ;LOAD HFP W/ EXP'D TARGET MICROADDR.
4246  013242  172437  002646                    LDF     MFAC0,AC0       ;EB[SF] FROM MFAC0
4247  013246  172537  002656                    LDF     MFAC1,AC1       ;EA[DF] FROM MFAC1
4248  013252  170102                            LDFPS   R2              ;F/D-MODE & FMM=1
4249                                                                    ;
4250  013254  104406                            ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
4251                                            ;--------ERROR-LOOP-ENTERS-HERE--------
4252                                                                    ;
4253  013256  174103                    63$:    STF     AC1,AC3         ;COPY DEST DATA
4254  013260  172300                            ADDF    AC0,AC3         ;IFORK(RITE), ER = EA[DF=3] - EB[SF=0]
4255  013262  105737  002645                    TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4256  013266  001373                            BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4257                                                                    ;
4258  013270  076600  000036                    MED     ,RFEC           ;GET FPSHI#FEC AFTER
4259  013274  020027  140016                    CMP     R0,#140016      ;DID HFP UBREAK ?
4260  013300  001401                            BEQ     35$             ;YES - ON TO NEXT DATA SET
4261  013302  104103                            ERROR   103             ;HFP MISSED THE TARGET
```

```
                        ;"IFORK/(ADD+SUB)*MO EXPNT RANGE.CODE ROM ERR"
                        ;       E-UBRK = EXP'D HFP UBREAK TARGET
                        ;       -FPS-- = HFP FPS STATUS WORD BEFORE UBRK (F/D-MODE)
                        ;       ER<8:0> = VALUE IN ER DURING TEST
                        ;       R-FPSHI/FEC = RCV'D FPSHI/FEC AFTER, EXP'D (140016)
                        ;       EB[SF] = EXPNT VALUE IN EB, TO GENERATE ER VALUE
                        ;       EA[DF] = EXPNT VALUE IN EA, TO GENERATE ER VALUE
                        ;                                               ;
                        ;*NEXT "ER" VALUE"
013304  005201  35$:    INC     R1                      ;BUMP "ER" LOOP CNTR
013306  020127  000777  CMP     R1,#777                 ;AT UPPER LIMIT ?
013312  003741          BLE     2$                      ;NOT YET
                                                        ;
                        ;*NEXT FPS(F/D) VALUE*
013314  105137  002624  COMB    FPLENF                  ;INVERT SENSE OF F/D KEY
013320  001403          BEQ     39$                     ;DONE IF (0) AGAIN
013322  052702  000200  BIS     #BIT7,R2                ;ELSE SET D-MODE IN FPS
013326  000732          BR      1$                      ;AND LOOP ON "ER" AGAIN
                                                        ;
                        ;*DONE WITH TESTING*
013330  104416  39$:    ZAPHFP                          ;CLEAN UP HFP AFTER
013332  000524          BR      TST34           ;;      ;ON TO NEXT TEST
                ;
                ;
                ;////////////////////////////////////////////////////////////////////
                ;
                ;       SUBR TO GENERATE NECESSARY EA[DF](MFAC1) AND EB[SF](MFAC0)
                ;       EXPNT'S TO FORM ER<8:0> VALUE REQ'D
                ;
                ;               R1 = INPUT ER<8:0> VALUE, (000)->(777)
                ;               MFAC0 = EB[SF] EXPNT VALUE
                ;               MFAC1 = EA[DF] EXPNT VALUE
                ;
013334  020127  000402  GTEAEB:  CMP     R1,#402                 ;ER >= 402 ?
013340  002005          BGE     10$                     ;BR IF YES
013342  020127  000376          CMP     R1,#376                 ;ER <= 376 ?
013346  003420          BLE     20$                     ;BR IF YES
013350  000261          SEC                             ;(377)-(401) RANGE CAN'T DO
013352  000207          RTS     PC                      ;AND EXIT
                                                        ;
013354  012737  000200  002656  10$:    MOV     #200,MFAC1+0            ;EA[DF] = 001<14:07>
013362  020127  000402          CMP     R1,#402                 ;AT ER=402 ?
013366  003003          BGT     11$                     ;BR IF PAST
013370  012737  100000  002646          MOV     #100000,MFAC0+0         ;RESET EB[SF] CNTR
013376  162737  000200  002646  11$:    SUB     #200,MFAC0+0            ;COUNT DOWN
013404  000241          CLC                             ;OK RETURN
013406  000207          RTS     PC                      ;AND EXIT
                                                        ;
013410  012737  000200  002646  20$:    MOV     #200,MFAC0+0            ;EB[SF] = 001<14:07>
013416  005701          TST     R1                      ;AT ER=000 ?
013420  003002          BGT     21$                     ;BR IF PAST
013422  005037  002656          CLR     MFAC1+0                 ;RESET EA[DF] CNTR
013426  062737  000200  002656  21$:    ADD     #200,MFAC1+0            ;COUNT UP
013434  000241          CLC                             ;OK RETURN
013436  000207          RTS     PC                      ;AND EXIT
                        ;
```

```
                        ;       METHOD:
                        ;
                        ;       ER = EA[DF] - EB[SF]
                        ;
                        ;       000   001      001       !
                        ;       ...   ...      001       !   UP COUNT
                        ;       376   377      001      \!/
                        ;       377 \
                        ;       400  >-CAN'T BE DONE
                        ;       401 /
                        ;       402   001      377       !
                        ;       ...   001      ...       !   DOWN COUNT
                        ;       777   001      002      \!/
                        ;
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;
                        ;       SUBR TO GET CONTENTS OF RANGE.CODE.ROM, AND
                        ;       FORM INTO A MICROBREAK TARGET ADDRESS
                        ;
                        ;               R1 = INPUT ER<8:0> VALUE, UNTOUCHED
                        ;               R3 = OUTPUT TARGET MICROADDRESS FOR LDUB
                        ;               FPLENF = 000=F/377=D HFP MODES
                        ;
013440  012703  013500  RNGCOD: MOV     #40$,R3                 ;DATA TABLE PTR
013444  020123          1$:     CMP     R1,(R3)+                ;(INPUT-VALUE) : (TABLE-VALUE)
013446  002402                  BLT     2$                      ;STOP @ NEXT LARGEST
013450  022323                  CMP     (R3)+,(R3)+             ;BUMP PAST 2 ENTRIES
013452  000774                  BR      1$                      ;TRY AGAIN
                                ;POINTING AT NEXT LARGEST - BACK UP 1 ENTRY
013454  016346  177774  2$:     MOV     -4(R3),-(SP)            ;SAVE BASE VALUE
013460  016303  177772          MOV     -6(R3),R3               ;GET D-MODE BIT<0> ALTER CODE
013464  105737  002624          TSTB    FPLENF                  ;F-OR-D MODE ?
013470  001001                  BNE     3$                      ;BR IF D
013472  005003                  CLR     R3                      ;CODE=0 IF F-MODE
013474  052603          3$:     BIS     (SP)+,R3                ;FORM COMPOSITE ADDR.
013476  000207                  RTS     PC                      ;AND DONE
                        ;
                        ;       TABLE FOR ABOVE:
                        ;
                        ;       LOW-ER  ALTER   IFORK[(ADD+SUB)*MO]
                        ;       VALUE   CODE    MICROADDRESS
                        ;                               CODE    ER<8:0>-VALUE
                        ;
013500  000000  000000  000161  40$:     000,    0,      161     ;EQ     000
013506  000001  000000  000162          001,    0,      162     ;GT1    001
013514  000002  000000  000163          002,    0,      163     ;GT2    002-013
013522  000014  000000  000165          014,    0,      165     ;GT3    014-030
013530  000031  000001  000164          031,    001,    164     ;GT3/MGT 031-070  (D/F-MODE)
013536  000071  000000  000164          071,    0,      164     ;MGT    071-377
013544  000400  000000  000174          400,    0,      174     ;MGT    400-707
```

```
4374
4375  013552  000710  000001  000174          710,    001,    174     ;MGT/GT3 710-747  (F/D-MODE)
4376
4377  013560  000750  000000  000175          750,    0,      175     ;GT3    750-764
4378
4379  013566  000765  000000  000173          765,    0,      173     ;GT2    765-776
4380
4381  013574  000777  000000  000172          777,    0,      172     ;GT1    777
4382
4383  013602  007777                          7777    ;<END>
4384
4385                                          ;////////////////////////////////////////////////////////////
4386
4387
4388
```

```
                        ;*****************************************************************
                        ;*TEST 34       FRACTION, FPINMUX-INBUF-FSPADMUX-FSPAD-FPOUTMUX DATAPATH, VIA "LDF/STF"
                        ;
                        ;       THIS TEST USES A FLOATING POINT "LDF/STF" SEQUENCE, RUNNING
                        ;       A SERIES OF DATA PATTERNS THRU THE FP11-E DATAPATH.  THE
                        ;       PATH INVOLVED HERE IS:
                        ;
                        ;         LDF:  BM(DMUX) -> FPINMUX(DMUX) -> INBUF(A&B) ->
                        ;                       -> FSPADMUX(INBUF-A&B) -> FSPAD(A&B)
                        ;
                        ;         STF:  FSPAD(A&B) -> FBUSA(A&B) -> FPOUTMUX(A&B) -> BM(BUSDIN)
                        ;
                        ;       NOTE THAT PREVIOUS TESTS VERIFIED THE EXPONENT DATAPATH USING
                        ;       THE "LDEXP/STEXP" SEQUENCE.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -EXPECTED F-MODE DATA
                        ;       MFAC1+  -RECEIVED F-MODE DATA
                        ;
                        ;       AC0     -ACC REFERENCE
                        ;
                        ;       R3      -(TEMP)
                        ;       R4      -COUNTER FOR LOOPS
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A), F.BUS.A-ENABLES, INBUF.A/B
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, MULNET.ALU.CNTL(A.SELECT),
                        ;         AR.CLK, FSPAD.WRITE/ENABLE(F)
                        ;
                        ;       FMUL/K10
                        ;         MULNET-ALU(A-SELECT), FPOUT.MUX(PORT-0,1)
                        ;
                        ;       FALU/K11
                        ;         F.BUS.A(F.MODE), FSPAD(F.MODE), FALU.DATA/CNTL(A), AR(F.MODE),
                        ;         ROUND.BITS(F.MODE), FSPAD.IN.MUX
                        ;
                        ;*****************************************************************
013604  000004         TST34:  SCOPE
                        ;
013606  170127  040040          LDFPS   #040040         ;F-MODE/INTR-DISABL/TRUNCATE
013612  012704  000006          MOV     #6,R4           ;6 ENTRIES IN DATA TABLE
013616  012705  013700          MOV     #40$,R5         ;PTR TO DATA
                        ;
                        ;*DATA LOOP ENTERS HERE*
013622  005237  002640  10$:    INC     @WLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
013626  012703  002646          MOV     #MFAC0,R3       ;INITIAL DATA
013632  012523                  MOV     (R5)+,(R3)+     ;GET IT FROM TABLE
013634  012523                  MOV     (R5)+,(R3)+     ;INTO MFAC0
```

```
                                                        ;
013636  104406                        ERRPNT          ;DONT CHANGE DATA IN ERROR LOOP
                                      ;--------ERROR-LOOP-ENTERS-HERE--------
                                                        ;
013640  172437  003060        63$:    LDF     FPZERO,AC0      ;(0.0) INTO SIGN/EXP/FRAC AC0
013644  172437  002646                LDF     MFAC0,AC0       ;DATA THRU INBUF TO SPAD'S
013650  174037  002656                STF     AC0,MFAC1       ;DATA FROM SPAD'S THRU FPOUTMUX
013654  105737  002645                TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
013660  001367                        BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                        ;
                                      ;NOW CHECK THE DATA
013662  104426  002646  002656        CMP32M  ,MFAC0,MFAC1    ;(EXPECTED) = (RECEIVED)?
013670  001401                        BEQ     30$             ;YES - BR
013672  104055                        ERROR   55              ;NO- SIGNAL ERROR
                                      ;"LDF/STF FRAC DATAPATH ERR"
                                      ;       EXPD AC0 = 32.BIT DATA LOADED/EXPECTED
                                      ;       RCVD AC0 = 32.BIT DATA STORED
                                      ;
013674  077426                30$:    SOB     R4,10$          ;COUNT ENTRIES
013676  000414                        BR      TST35           ;;      ;ON TO NEXT TEST WHEN DONE
                                      ;
                                      ;---DATA FOR ABOVE TEST---

013700  177777  000000        40$:    .WORD   177777,000000           ;WORD-A, ALL 1'S

013704  125252  000000                .WORD   125252,000000           ;       1/0'S

013710  052525  000000                .WORD   052525,000000           ;       0/1'S

013714  000000  177777                .WORD   000000,177777           ;WORD-B, ALL 1'S

013720  000000  125252                .WORD   000000,125252           ;       1/0'S

013724  000000  052525                .WORD   000000,052525           ;       0/1'S


                                      ;;*****************************************************************
                                      ;*TEST 35       FRACTION, 60. BIT DATAPATH, VIA "LDD/STD"
                                      ;
                                      ;       THIS TEST USES A FLOATING POINT "LDD/STD" SEQUENCE, RUNNING
                                      ;       A SERIES OF DATA PATTERNS THRU THE FP11-E DATAPATH.  THE
                                      ;       PATH INVOLVED HERE IS:
                                      ;
                                      ;         LDD:  BM(DMUX) -> FPINMUX(DMUX) -> INBUF(A&B) -> FSPADMUX(INBUF-A&B) ->
                                      ;                       -> FSPAD(A&B) -> AR(A&B)/FBUSA(INBUF-C&D) -> FSPADMUX(AR) ->
                                      ;                       -> FSPAD(A&B&C&D)
                                      ;
                                      ;         STD:  FSPAD(A&B&C&D) -> FBUSA(A&B&C&D) -> FPOUTMUX(A&B&C&D) -> BM(BUSDIN)
                                      ;
                                      ;       NOTE THAT PREVIOUS TESTS VERIFIED THE EXPONENT DATAPATH USING
                                      ;       THE "LDX/STX" SEQUENCE; AND THE "LDF/STF" PATH, DIRECTLY ABOVE.
                                      ;
                                      ;       ----------
                                      ;       REGISTER/LOCATION USE:
                                      ;
```

```
4501                            ;       MFAC0+  -EXPECTED D-MODE DATA
4502                            ;       MFAC1+  -RECEIVED D-MODE DATA
4503                            ;
4504                            ;       AC0     -ACC REFERENCE
4505                            ;
4506                            ;       R3      -(TEMP)
4507                            ;       R4      -COUNTER FOR LOOPS
4508                            ;       R5      -DATA TABLE PTR
4509                            ;
4510                            ;       ----------
4511                            ;       MODULE/ERROR INFO:
4512                            ;
4513                            ;       FNUA/K8
4514                            ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A), F.BUS.A-ENABLES
4515                            ;
4516                            ;       FEXP/K9
4517                            ;         CROM/LATCHES, BUT<2:0>-LOGIC, MULNET.ALU.CNTL(A.SELECT),
4518                            ;         AR.CLK, FSPAD.WRITE/ENABLE(D.MODE)
4519                            ;
4520                            ;       FMUL/K10
4521                            ;         MULNET-ALU(A-SELECT), FPOUTMUX(PORT-2,3)
4522                            ;
4523                            ;       FALU/K11
4524                            ;         F.BUS.A(D.MODE), FSPAD(D.MODE), FALU.DATA/CNTL(A), AR(D.MODE),
4525                            ;         ROUND.BITS(D.MODE)
4526                            ;
4527                            ;*****************************************************************
4528  013730  000004            T35:    SCOPE
4529                                                                    ;
4530  013732  170127  040240            LDFPS   #040240               ;D-MODE, INTR-DISABL/TRUNC
4531  013736  012704  000014            MOV     #12.,R4               ;12. DATA TABLE ENTRIES
4532  013742  012705  014030            MOV     #40$,R5               ;PTR TO DATA
4533                                                                    ;
4534                                    ;*DATA LOOP ENTERS HERE*
4535  013746  005237  002640    10$:    INC     DWLOOP                ;BUMP CLOCK IN.A.LOOP COUNT
4536  013752  012703  002646            MOV     #MFAC0,R3             ;INITIAL DATA
4537  013756  012523                    MOV     (R5)+,(R3)+           ;FROM TABLE TO MFAC0
4538  013760  012523                    MOV     (R5)+,(R3)+           ;
4539  013762  012523                    MOV     (R5)+,(R3)+           ;
4540  013764  012523                    MOV     (R5)+,(R3)+           ;
4541                                                                    ;
4542  013766  104406                    ERRPNT                        ;DONT CHANGE DATA IN ERROR LOOP
4543                                    ;--------ERROR-LOOP-ENTERS-HERE--------
4544                                                                    ;
4545  013770  172437  003060    63$:    LDD     FPZERO,AC0            ;INIT AC0 S/E/F=(0,0,0,0)
4546  013774  172437  002646            LDD     MFAC0,AC0             ;DATA THRU FULL DATAPATH TO SPAD'S
4547  014000  174037  002656            STD     AC0,MFAC1             ;DATA FROM SPAD'S THRU FPOUTMUX
4548  014004  105737  002645            TSTB    LPTITE                ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4549  014010  001367                    BNE     63$                   ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4550                                                                    ;
4551                                    ;NOW CHECK THE DATA
4552  014012  104425  002646  002656    CMP64M  ,MFAC0,MFAC1          ;(EXPECTED) = (RECEIVED)?
4553  014020  001401                    BEQ     30$                   ;YES - BR
4554  014022  104054                    ERROR   54                    ;NO - SIGNAL ERROR
4555                                    ;"LDD/STD FRAC DATAPATH ERR"
4556                                    ;       EXPD AC0 = 64.BIT DATA LOADED/EXPECTED
```

```
                                        ;       RCVD AC0 = 64.BIT DATA STORED
                                        ;
014024  077430                  30$:    SOB     R4,10$                  ;COUNT ENTRIES
014026  000460                          BR      TST36           ;;      ;ON TO NEXT TEST WHEN DONE
                                        ;
                                        ; --DATA FOR ABOVE TEST---
014030  177777  000000  000000  40$:    .WORD   M1,0,0,0                ;WORD-A, ALL 1'S
014036  000000
014040  125252  000000  000000          .WORD   AN,0,0,0                ;       1/0'S
014046  000000
014050  052525  000000  000000          .WORD   AP,0.0,0                ;       0/1'S
014056  000000
014060  000000  177777  000000          .WORD   0,M1,0,0                ;WORD-B, ALL 1'S
014066  000000
014070  000000  125252  000000          .WORD   0,AN,0,0                ;       1/0'S
014076  000000
014100  000000  052525  000000          .WORD   0,AP,0,0                ;       0/1'S
014106  000000
014110  000000  000000  177777          .WORD   0,0,M1,0                ;WORD-C, ALL 1'S
014116  000000
014120  000000  000000  125252          .WORD   0,0,AN,0                ;       1/0'S
014126  000000
014130  000000  000000  052525          .WORD   0,0,AP,0                ;       0/1'S
014136  000000
014140  000000  000000  000000          .WORD   0,0,0,M1                ;WORD-D,  ALL 1'S
014146  177777
014150  000000  000000  000000          .WORD   0,0,0,AN                ;       1/0'S
014156  125252
014160  000000  000000  000000          .WORD   0,0,0,AP                ;       0/1'S
014166  052525


                                ;*****************************************************************
                                ;*TEST 36       FRACTION, FSPAD DATA PATTERNS, AC0-AC5
                                ;
                                ;       THIS TEST RUNS A SERIES OF DATA PATTERNS THRU EACH OF THE FRACTION
                                ;       SCRATCHPADS [FULL 60. BITS, D-MODE] AC0 - AC5.
                                ;
                                ;       ----------
                                ;       REGISTER/LOCATION USE:
                                ;
                                ;       MFAC0+  -INPUT/EXPECTED DATA
                                ;       MFAC1+  -OUTPUT/RECEIVED DATA
```

```
                        ;
                        ;       AC0...AC5 -FOR DATA
                        ;
                        ;       R0      -ACC. NUMBER, 0-5
                        ;       R4      -(PTR)
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A), F.BUS.A-ENABLES/EMIT.F
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, MULNET.ALU.CNTL(A.SELECT),
                        ;         FSPAD.WRITE/ENABLE(D.MODE)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         F.BUS.A(D.MODE), FSPAD(D.MODE), FALU.DATA/CNTL(A), AR(D.MODE)
                        ;
                        ;*****************************************************************
014170  000004          TST36:  SCOPE
                                                        ;
014172  170127  040240          LDFPS   #040240         ;INTR-DISAB/D-MODE/TRUNC
                                                        ;
                        ;---------------DATA PATTERNS IN "AC0"----------------
014176  012705  014702  10$:    MOV     #40$,R5         ;PTR TO DATA
014202  005000                  CLR     R0              ;ACC NUMBER CNTR
                                                        ;
                                ;*DATA LOOP ENTERS HERE*
014204  005237  002640  20$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
014210  012704  002646          MOV     #MFAC0+0,R4     ;INITIAL DATA IN MFAC0
014214  012524                  MOV     (R5)+,(R4)+     ;GET WORD-A
014216  100421                  BMI     11$             ;IF -, DONE FOR NOW
014220  012524                  MOV     (R5)+,(R4)+     ;GET WORD-B
014222  012524                  MOV     (R5)+,(R4)+     ;GET WORD-C
014224  012524                  MOV     (R5)+,(R4)+     ;GET WORD-D
                                                        ;
014226  104406                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                        ;
014230  172437  002646  60$:    LDD     MFAC0,AC0       ;DATA-PATTERN -> ACC
014234  174037  002656          STD     AC0,MFAC1       ;ACC -> MEMORY
014240  105737  002645          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
014244  001371                  BNE     60$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                        ;
014246  104425  002646  002656  CMP64M  ,MFAC0,MFAC1    ;(LOADED) = (STORED) ??
014254  001753                  BEQ     20$             ;BR IF AGREE
014256  104056                  ERROR   56              ;ELSE FSPAD DATA ERROR
                                ;"FSPAD DATA ERROR"
                                ;       ACC###   = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
                                ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
                                ;       RCVD ACC = RECEIVED 64.BIT DATA
```

```
4669
4670  014260  000751                          BR      20$                     ;NEXT DATA PATTERN
4671
4672                                  ;----------------DATA PATTERNS IN "AC1"----------------
4673  014262  012705  014702          11$:    MOV     #40$,R5                 ;PTR TO DATA
4674  014266  005200                          INC     R0                      ;BUMP ACC NUMBER CNTR
4675                                          ;
4676                                  ;*DATA LOOP ENTERS HERE*
4677  014270  005237  002640          21$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
4678  014274  012704  002646                  MOV     #MFAC0+0,R4             ;INITIAL DATA IN MFAC0
4679  014300  012524                          MOV     (R5)+,(R4)+             ;GET WORD-A
4680  014302  100421                          BMI     12$                     ;IF -, DONE FOR NOW
4681  014304  012524                          MOV     (R5)+,(R4)+             ;GET WORD-B
4682  014306  012524                          MOV     (R5)+,(R4)+             ;GET WORD-C
4683  014310  012524                          MOV     (R5)+,(R4)+             ;GET WORD-D
4684                                          ;
4685  014312  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4686                                          ;--------ERROR-LOOP-ENTERS-HERE--------
4687                                          ;
4688  014314  172537  002646          61$:    LDD     MFAC0,AC1               ;DATA-PATTERN -> ACC
4689  014320  174137  002656                  STD     AC1,MFAC1               ;ACC -> MEMORY
4690  014324  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4691  014330  001371                          BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4692                                          ;
4693  014332  104425  002646  002656          CMP64M  ,MFAC0,MFAC1            ;(LOADED) = (STORED) ??
4694  014340  001753                          BEQ     21$                     ;BR IF AGREE
4695  014342  104056                          ERROR   56                      ;ELSE FSPAD DATA ERROR
4696                                          ;"FSPAD DATA ERROR"
4697                                          ;       ACC###   = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
4698                                          ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
4699                                          ;       RCVD ACC = RECEIVED 64.BIT DATA
4700                                          ;
4701  014344  000751                          BR      21$                     ;NEXT DATA PATTERN
4702                                          ;
4703                                  ;----------------DATA PATTERNS IN "AC2"----------------
4704  014346  012705  014702          12$:    MOV     #40$,R5                 ;PTR TO DATA
4705  014352  005200                          INC     R0                      ;BUMP ACC NUMBER CNTR
4706                                          ;
4707                                  ;*DATA LOOP ENTERS HERE*
4708  014354  005237  002640          22$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
4709  014360  012704  002646                  MOV     #MFAC0+0,R4             ;INITIAL DATA IN MFAC0
4710  014364  012524                          MOV     (R5)+,(R4)+             ;GET WORD-A
4711  014366  100421                          BMI     13$                     ;IF -, DONE FOR NOW
4712  014370  012524                          MOV     (R5)+,(R4)+             ;GET WORD-B
4713  014372  012524                          MOV     (R5)+,(R4)+             ;GET WORD-C
4714  014374  012524                          MOV     (R5)+,(R4)+             ;GET WORD-D
4715                                          ;
4716  014376  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4717                                          ;--------ERROR-LOOP-ENTERS-HERE--------
4716                                          ;
4719  014400  172637  002646          62$:    LDD     MFAC0,AC2               ;DATA-PATTERN -> ACC
4720  014404  174237  002656                  STD     AC2,MFAC1               ;ACC -> MEMORY
4721  014410  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4722  014414  001371                          BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4723                                          ;
4724  014416  104425  002646  002656          CMP64M  ,MFAC0,MFAC1            ;(LOADED) = (STORED) ??
```

```
4725  014424  001753                  BEQ     22$                     ;BR IF AGREE
4726  014426  104056                  ERROR   56                      ;ELSE FSPAD DATA ERROR
4727                                  ;"FSPAD DATA ERROR"
4728                                  ;       ACC###    = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
4729                                  ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
4730                                  ;       RCVD ACC = RECEIVED 64.BIT DATA
4731                                  ;
4732  014430  000751                  BR      22$                     ;NEXT DATA PATTERN
4733                                  ;
4734                          ;----------------DATA PATTERNS IN "AC3"----------------
4735  014432  012705  014702  13$:    MOV     #40$,R5                 ;PTR TO DATA
4736  014436  005200                  INC     R0                      ;BUMP ACC NUMBER CNTR
4737                                  ;
4738                                  ;*DATA LOOP ENTERS HERE*
4739  014440  005237  002640  23$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
4740  014444  012704  002646          MOV     #MFAC0+0,R4             ;INITIAL DATA IN MFAC0
4741  014450  012524                  MOV     (R5)+,(R4)+             ;GET WORD-A
4742  014452  100421                  BMI     14$                     ;IF -, DONE FOR NOW
4743  014454  012524                  MOV     (R5)+,(R4)+             ;GET WORD-B
4744  014456  012524                  MOV     (R5)+,(R4)+             ;GET WORD-C
4745  014460  012524                  MOV     (R5)+,(R4)+             ;GET WORD-D
4746                                  ;
4747  014462  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4748                                  ;--------ERROR-LOOP-ENTERS-HERE--------
4749                                  ;
4750  014464  172737  002646  63$:    LDD     MFAC0,AC3               ;DATA-PATTERN -> ACC
4751  014470  174337  002656          STD     AC3,MFAC1               ;ACC -> MEMORY
4752  014474  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4753  014500  001371                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4754                                  ;
4755  014502  104425  002646  002656  CMP64M  ,MFAC0,MFAC1            ;(LOADED) = (STORED) ??
4756  014510  001753                  BEQ     23$                     ;BR IF AGREE
4757  014512  104056                  ERROR   56                      ;ELSE FSPAD DATA ERROR
4758                                  ;"FSPAD DATA ERROR"
4759                                  ;       ACC###    = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
4760                                  ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
4761                                  ;       RCVD ACC = RECEIVED 64.BIT DATA
4762                                  ;
4763  014514  000751                  BR      23$                     ;NEXT DATA PATTERN
4764                                  ;
4765                          ;----------------DATA PATTERNS IN "AC4"----------------
4766  014516  012705  014702  14$:    MOV     #40$,R5                 ;PTR TO DATA
4767  014522  005200                  INC     R0                      ;BUMP ACC NUMBER CNTR
4768                                  ;
4769                                  ;*DATA LOOP ENTERS HERE*
4770  014524  005237  002640  24$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
4771  014530  012704  002646          MOV     #MFAC0+0,R4             ;INITIAL DATA IN MFAC0
4772  014534  012524                  MOV     (R5)+,(R4)+             ;GET WORD-A
4773  014536  100423                  BMI     15$                     ;IF -, DONE FOR NOW
4774  014540  012524                  MOV     (R5)+,(R4)+             ;GET WORD-B
4775  014542  012524                  MOV     (R5)+,(R4)+             ;GET WORD-C
4776  014544  012524                  MOV     (R5)+,(R4)+             ;GET WORD-D
4777                                  ;
4778  014546  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4779                                  ;--------ERROR-LOOP-ENTERS-HERE--------
4780                                  ;
```

```
4781  014550  172737  002646          64$:    LDD     MFAC0,AC3               ;DATA-PATTERN -> TEMP
4782  014554  174304                          STD     AC3,AC4                 ;TEMP -> ACC
4783  014556  172704                          LDD     AC4,AC3                 ;ACC -> TEMP
4784  014560  174337  002656                  STD     AC3,MFAC1               ;TEMP -> MEMORY
4785  014564  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4786  014570  001367                          BNE     64$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4787                                                                          ;
4788  014572  104425  002646  002656          CMP64M  ,MFAC0,MFAC1            ;(LOADED) = (STORED) ??
4789  014600  001751                          BEQ     24$                     ;BR IF AGREE
4790  014602  104056                          ERROR   56                      ;ELSE FSPAD DATA ERROR
4791                                          ;"FSPAD DATA ERROR"
4792                                          ;       ACC###    = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
4793                                          ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
4794                                          ;       RCVD ACC = RECEIVED 64.BIT DATA
4795                                                                          ;
4796  014604  000747                          BR      24$                     ;NEXT DATA PATTERN
4797                                                                          ;
4798                                  ;----------------DATA PATTERNS IN "AC5"----------------
4799  014606  012705  014702          15$:    MOV     #40$,R5                 ;PTR TO DATA
4800  014612  005200                          INC     R0                      ;BUMP ACC NUMBER CNTR
4801                                                                          ;
4802                                          ;*DATA LOOP ENTERS HERE*
4803  014614  005237  002640          25$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
4804  014620  012704  002646                  MOV     #MFAC0+0,R4             ;INITIAL DATA IN MFAC0
4805  014624  012524                          MOV     (R5)+,(R4)+             ;GET WORD-A
4806  014626  100423                          BMI     16$                     ;IF -, DONE WITH THIS TEST
4807  014630  012524                          MOV     (R5)+,(R4)+             ;GET WORD-B
4808  014632  012524                          MOV     (R5)+,(R4)+             ;GET WORD-C
4809  014634  012524                          MOV     (R5)+,(R4)+             ;GET WORD-D
4810                                                                          ;
4811  014636  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
4812                                          ;--------ERROR-LOOP-ENTERS-HERE--------
4813                                                                          ;
4814  014640  172637  002646          65$:    LDD     MFAC0,AC2               ;DATA-PATTERN -> TEMP
4815  014644  174205                          STD     AC2,AC5                 ;TEMP -> ACC
4816  014646  172605                          LDD     AC5,AC2                 ;ACC -> TEMP
4817  014650  174237  002656                  STD     AC2,MFAC1               ;TEMP -> MEMORY
4818  014654  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
4819  014660  001367                          BNE     65$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
4820                                                                          ;
4821  014662  104425  002646  002656          CMP64M  ,MFAC0,MFAC1            ;(LOADED) = (STORED) ??
4822  014670  001751                          BEQ     25$                     ;BR IF AGREE
4823  014672  104056                          ERROR   56                      ;ELSE FSPAD DATA ERROR
4824                                          ;"FSPAD DATA ERROR"
4825                                          ;       ACC###    = ACCUMULATOR NUMBER (0) -> (5) UNDER TEST
4826                                          ;       EXPD ACC = LOADED/EXPECTED 64.BIT DATA
4827                                          ;       RCVD ACC = RECEIVED 64.BIT DATA
4828                                                                          ;
4829  014674  000747                          BR      25$                     ;NEXT DATA PATTERN
4830                                                                          ;
4831  014676  000137  015254          16$:    JMP     FSPD01                  ;EXIT THE HARD WAY
4832                                          ;
4833                                          ;////////////////////////////////////////////////////////////
4834                                          ;
4835                                          ;       DATA FOR ABOVE TEST:
4836                                          ;
```

```
 4837
 4838  014702  000000  000000  000000  40$:    ↑B0000C000,↑B0000000000000000,↑B0000000000000000,↑B0000C0000000000
 4839  014710  000000
 4840
 4841  014712  000120  000000  000000          ↑B01010000,↑B0000000000000000,↑B0000C0000000000,↑B0000000000000000
 4842  014720  000000
 4843
 4844  014722  000040  000000  000000          ↑B00100000,↑B0000000000000000,↑B0000000000000000,↑B0000000000000000
 4845  014730  000000
 4846
 4847  014732  000005  000000  000000          ↑B00000101,↑B0000000000000000,↑B0000000000000000,↑B0000000000000000
 4848  014740  000000
 4849
 4850  014742  000012  000000  000000          ↑B00001010,↑B0000000000000000,↑B0000000000000000,↑B0000000000000000
 4851  014750  000000
 4852
 4853  014752  000000  050000  000000          ↑B00000000,↑B0101000000000000,↑B0000000000000000,↑B0000000C0000000
 4854  014760  000000
 4855
 4856  014762  000000  120000  000000          ↑B00000000,↑B1010000000000000,↑B0000000000000000,↑B0000000000000000
 4857  014770  000000
 4858
 4859  014772  000000  002400  000000          ↑B00000000,↑B000001010000000,↑B0000000000000000,↑B0000000000000000
 4860  015000  000000
 4861
 4862  015002  000000  005000  000000          ↑B00000000,↑B0000101000000000,↑B0000000000000000,↑B0000000000000000
 4863  015010  000000
 4864
 4865  015012  000000  000120  000000          ↑B00000000,↑B0000000001010000,↑B0000000000000000,↑B0000000000000000
 4866  015020  000000
 4867
 4868  015022  000000  000240  000000          ↑B00000000,↑B0000000010100000,↑B0000000000000000,↑B0000000000000000
 4869  015030  000000
 4870
 4871  015032  000000  000005  000000          ↑B00000000,↑B0000000000000101,↑B0000000000000000,↑B0000000000000000
 4872  015040  000000
 4873
 4874  015042  000000  000012  000000          ↑B00000000,↑B0000000000001010,↑B0000000000000000,↑B0000000000000000
 4875  015050  000000
 4876
 4877  015052  000000  000000  050000          ↑B00000000,↑B0000000000000000,↑B0101000000000000,↑B0000000000000000
 4878  015060  000000
 4879
 4880  015062  000000  000000  120000          ↑B00000000,↑B0000000000000000,↑B1010000000000000,↑B0000000000000000
 4881  015070  000000
 4882
 4883  015072  000000  000000  002400          ↑B00000000,↑B0000000000000000,↑B0000010100000000,↑B0000000000000000
 4884  015100  000000
 4885
 4886  015102  000000  000000  005000          ↑B00000000,↑B0000000000000000,↑B0000101000000000,↑B0000000000000000
 4887  015110  000000
 4888
 4889  015112  000000  000000  000120          ↑B00000000,↑B0000000000000000,↑B0000000001010000,↑B0000000000000000
 4890  015120  000000
 4891
 4892  015122  000000  000000  000240          ↑B00000000,↑B0000000000000000,↑B0000000010100000,↑B0000000000000000
```

```
4893  015130  000000
4894
4895  015132  000000  000000  000005          ↑B00000000,↑B0000000000000000,↑B0000000000000101,↑B0000000000000000
4896  015140  000000
4897
4898  015142  000000  000000  000012          ↑B00000000,↑B0000000000000000,↑B0000000000001010,↑B0000000000000000
4899  015150  000000
4900
4901  015152  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0101000000000000
4902  015160  050000
4903
4904  015162  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B1010000000000000
4905  015170  120000
4906
4907  015172  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000010100000000
4908  015200  002400
4909
4910  015202  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000101000000000
4911  015210  005000
4912
4913  015212  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000000001010000
4914  015220  000120
4915
4916  015222  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000000010100000
4917  015230  000240
4918
4919  015232  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000000000000101
4920  015240  000005
4921
4922  015242  000000  000000  000000          ↑B00000000,↑B0000000000000000,↑B0000000000000000,↑B0000000000001010
4923  015250  000012
4924
4925  015252  177777                          -1      ;<DONE>
4926
4927  015254                              FSPD01:
4928  015254  000400                              BR      TST37           ;;      ;NEXT TEST, THE HARD WAY
4929
4930                                      ;////////////////////////////////////////////////////////////////////////
4931
4932
4933
4934                                      ;;*****************************************************************
4935                                      ;*TEST 37       FPINIT/FP.EMIT.(E&F)/FSPAD EXACT.ZERO
4936                                      ;
4937                                      ;       THIS TEST VERIFIES THAT THE "FP.INIT" FLOW CORRECTLY STORED
4938                                      ;       A (200,000000,000000,000000) IN THE "EXACT.ZERO" (FSPAD[17])
4939                                      ;       ACCUMULATOR.  NOTE THAT ONLY BITS <57:03> ARE CHECKED HERE, THE
4940                                      ;       OTHERS <59:58>="01" WILL BE VERIFIED LATER; THEY ARE NOT
4941                                      ;       DIRECTLY VISIBLE.  THIS TEST ALSO INDIRECTLY VERIFIES THE
4942                                      ;       "FP.EMIT.F" LOGIC, WHICH WAS USED TO GENERATE THE ZEROES.
4943                                      ;       THE EXPONENT AND SIGN FIELDS ARE ALSO VERIFIED TO BE "0" AND
4944                                      ;       (000).
4945                                      ;
4946                                      ;       ----------
4947                                      ;       REGISTER/LOCATION USE:
4948                                      ;
```

```
                                ;       MFACO+  -OUTPUT D-MODE DATA, SHOULD BE "EXACT.ZERO"
                                ;
                                ;       ACO     -(TEMP)
                                ;
                                ;       ----------
                                ;       MODULE/ERROR INFO:
                                ;
                                ;       FNUA/K8
                                ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(ZERO,A), FSPAD.ADDR.MUX/REG,
                                ;         F.BUS.A-ENABLES/EMIT.F
                                ;
                                ;       FEXP/K9
                                ;         CROM/LATCHES, BUT<2:0>-LOGIC,
                                ;         FSPAD.WRITE/ENABLE(D), FP.EMIT.F
                                ;
                                ;       FMUL/K10
                                ;         [PREVIOUSLY VERIFIED]
                                ;
                                ;       FALU/K11
                                ;         F.BUS.A(D.MODE), FSPAD(D.MODE), FALU.DATA/CNTL(A,ZERO),
                                ;
                                ;;*****************************************************************
015256  000004                  TST37:  SCOPE
                                ;                                               ;
015260  170127  040240                  LDFPS   #040240                         ;INTR-DISAB/D-MODE/TRUNC
                                ;                                               ;
015264  104406                  ERRPNT                                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                ;                                               ;
015266  172437  003004                  LDD     FPALTP,AC0                      ;PRESET OUTPUT TO 4*(052525)
015272  174000                  63$:    STD     AC0,AC0                         ;COPY AC[DF] -> AC[SF]
015274  170400                          CLRD    AC0                             ;READ FSPAD[17] "EXACT.ZERO"
015276  105737  002645                  TSTB    LPTITE                          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
015302  001374                          BNE     63$                             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                ;                                               ;
015304  172400                          LDD     AC0,AC0                         ;COPY AC[SF] -> AC[DF]
015306  174037  002646                  STD     AC0,MFAC0                       ;AC[DF] -> MEMORY
                                ;                                               ;
015312  104425  003060  002646          CMP64M  ,FPZERO,MFAC0                   ;ANSWER = (0,0,0,0) ???
015320  001401                          BEQ     TST40           ;;              ;NEXT TEST IF AGREE
015322  104107                          ERROR   107                             ;ELSE ERROR
                                ;"FRACTION/CLRD-EXEC/FPEMIT.F DATA ERR"
                                ;       RCVD AC0 = RECEIVED DATA, EXP'D (000000,000000,000000,000000)
                                ;                                               ;
```

```
                                ;;*****************************************************************
                                ;*TEST 40        FRACTION, FSPAD ADDRESSING VIA R[DF] AND R[SF]
                                ;
                                ;       THIS TEST VERIFIES THE SCRATCHPAD ADDRESSING FUNCTIONS:
                                ;
                                ;         R[SF]<3:0> == "0"*FIRB<2:0>  AND
                                ;
                                ;         R[DF]<3:0> == "00"*FIRB<7:6>
                                ;
```

```
                        ;       ADDRESS MODES IN THE FRACTION SCRATCHPADS.
                        ;
                        ;       THIS TEST LOOKS FOR STUCK 0/1 CONDITIONS IN THE 3 LOW ORDER
                        ;       BITS OF THE SCRATCHPAD ADDRESS PATH, FROM FIR -> THE SPAD.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -INPUT DATA PATTERN
                        ;       MFAC1+  -OUTPUT, AFTER ADDRESSING CHECK
                        ;
                        ;       AC0     -(TEMP)
                        ;       ...
                        ;       AC5     -(TEMP)
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL, FSPAD.ADDR.MUX/REG,
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         F.BUS.A(D.MODE), FSPAD.ADDR(D.MODE)
                        ;
                        ;;*****************************************************************
015324  000004          TST40:  SCOPE
                                ;
015326  170127  040240          LDFPS   #040240                 ;INTR-DISAB/D-MODE/TRUNC
015332  012704  177600          MOV     #177600,R4              ;MASK TO ZAP SIGN/EXPNT
                        ;
                        ;------------R[SF]=FIRB<2:0> ADDRESSING, CHECK FOR STUCK-0'S, BITS<2:0>------------
015336  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
015340  172737  003070  059$:   LDD     FPZEAP,AC3              ;(FPZEAP) -> FIRB<7:6>="11"
015344  174300                  STD     AC3,AC0                 ;FIRB<7:6>="11" -> FIRB<2:0>="000"
015346  170401                  CLRD    AC1                     ;ZEROES -> FIRB<2:0>="001"
015350  170402                  CLRD    AC2                     ;ZEROES -> FIRB<2:0>="010"
015352  170404                  CLRD    AC4                     ;ZEROES -> FIRB<2:0>="100"
015354  172700                  LDD     AC0,AC3                 ;FIRB<2:0>="000" -> FIRB<7:6>="11"
015356  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
015362  001366                  BNE     59$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                ;
015364  174337  002656          STD     AC3,MFAC1               ;FIRB<7:6>="11" -> MFAC1
015370  040437  002656          BIC     R4,MFAC1                ;ZERO UNWANTED BITS OF RESULT,
015374  104425  003070  002656  CMP64M  ,FPZEAP,MFAC1           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
015402  001401                  BEQ     .+4                     ;BR IF AGREE
015404  104007                  ERROR   07                      ;SIGNAL ERROR ... IF NOT
                                ;"FSPAD ADDRS ERROR; R[SF]"
                                ;       RCVD ACC = RECEIVED ACC AFTER SEQ, EXP'D TO BE (125.AP.AP.AP)
                                ;                       NOT (0,0,0,0)
```

```
5061                                                    ;
5062                                    ;------------R[SF]=FIRB<2:0> ADDRESSING,'CHECK FOR STUCK-1'S, BITS<1:0>------------
5063  015406  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5064                                            ;--------ERROR-LOOP-ENTERS-HERE--------
5065  015410  172437  003070          60$:      LDD     FPZEAP,AC0              ;(FPZEAP) -> FIRB<7:6>="00"
5066  015414  174003                            STD     AC0,AC3                 ;FIRB<7:6>="00" -> FIRB<2:0>="011"
5067  015416  170401                            CLRD    AC1                     ;ZEROES -> FIRB<2:0>="001"
5068  015420  170402                            CLRD    AC2                     ;ZEROES -> FIRB<2:0>="010"
5069  015422  172403                            LDD     AC3,AC0                 ;FIRB<2:0>="011" -> FIRB<7:6>="00"
5070  015424  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5071  015430  001367                            BNE     60$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5072                                                    ;
5073  015432  174037  002656                    STD     AC0,MFAC1               ;FIRB<7:6>="00" -> MFAC1
5074  015436  040437  002656                    BIC     R4,MFAC1                ;ZERO UNWANTED BITS OF RESULT,
5075  015442  104425  003070  002656            CMP64M  ,FPZEAP,MFAC1           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
5076  015450  001401                            BEQ     .+4                     ;BR IF AGREE
5077  015452  104007                            ERROR   07                      ;SIGNAL ERROR ... IF NOT
5078                                            ;"FSPAD ADDRS ERROR; R[SF]"
5079                                            ;       RCVD ACC = RECEIVED ACC AFTER SEQ, EXP'D TO BE (125,AP,AP,AP)
5080                                            ;                       NOT (0,0,0,0)
5081                                                    ;
5082                                    ;------------R[SF]=FIRB<2:0> ADDRESSING,'CHECK FOR STUCK-1, BIT<2>------------
5083  015454  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5084                                            ;--------ERROR-LOOP-ENTERS-HERE--------
5085  015456  172737  003070          61$:      LDD     FPZEAP,AC3              ;(FPZEAP) -> FIRB<7:6>="11"
5086  015462  174304                            STD     AC3,AC4                 ;FIRB<7:6>="11" -> FIRB<2:0>="100"
5087  015464  170400                            CLRD    AC0                     ;ZEROES -> FIRB<2:0>="000"
5088  015466  172704                            LDD     AC4,AC3                 ;FIRB<2:0>="100" -> FIRB<7:6>="11"
5089  015470  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5090  015474  001370                            BNE     61$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5091                                                    ;
5092  015476  174337  002656                    STD     AC3,MFAC1               ;FIRB<7:6>="11" -> MFAC1
5093  015502  040437  002656                    BIC     R4,MFAC1                ;ZERO UNWANTED BITS OF RESULT,
5094  015506  104425  003070  002656            CMP64M  ,FPZEAP,MFAC1           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
5095  015514  001401                            BEQ     .+4                     ;BR IF AGREE
5096  015516  104007                            ERROR   07                      ;SIGNAL ERROR ... IF NOT
5097                                            ;"FSPAD ADDRS ERROR; R[SF]"
5098                                            ;       RCVD ACC = RECEIVED ACC AFTER SEQ, EXP'D TO BE (125,AP,AP,AP)
5099                                            ;                       NOT (0,0,0,0)
5100                                                    ;
5101                                    ;------------R[DF]=FIRB<7:6> ADDRESSING,'CHECK FOR STUCK-0'S, BITS<7:6>------------
5102  015520  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5103                                            ;--------ERROR-LOOP-ENTERS-HERE--------
5104  015522  172437  003070          62$:      LDD     FPZEAP,AC0              ;(FPZEAP) -> FIRB<7:6>="00"
5105  015526  172537  003060                    LDD     FPZERO,AC1              ;ZEROES -> FIRB<7:6>="01"
5106  015532  172637  003060                    LDD     FPZERO,AC2              ;ZEROES -> FIRB<7:6>="10"
5107  015536  174037  002656                    STD     AC0,MFAC1               ;FIRB<7:6>="00" -> MFAC1
5108  015542  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5109  015546  001365                            BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5110                                                    ;
5111  015550  040437  002656                    BIC     R4,MFAC1                ;ZERO UNWANTED BITS OF RESULT,
5112  015554  104425  003070  002656            CMP64M  ,FPZEAP,MFAC1           ;COMPARE (EXPD):(RCVD), FOR EQ/NE
5113  015562  001401                            BEQ     .+4                     ;BR IF AGREE
5114  015564  104006                            ERROR   06                      ;SIGNAL ERROR ... IF NOT
5115                                            ;"FSPAD ADDRS ERROR; R[DF]"
5116                                            ;       RCVD ACC = RECEIVED ACC AFTER SEQ, EXP'D TO BE (125,AP,AP,AP)
```

```
                                        ;                       NOT (0,0,0,0)
                                                                ;
                                        ;-----------R[DF]=FIRB<7:6> ADDRESSING, CHECK FOR STUCK-1'S, BITS<7:6>-----------
015566  104406                                ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                        ;--------ERROR-LOOP-ENTERS-HERE--------
015570  172737  003070          63$:   LDD     FPZEAP,AC3                      ;(FPZEAP) -> FIRB<7:6>="11"
015574  172537  003060                 LDD     FPZERO,AC1                      ;ZEROES -> FIRB<7:6>="01"
015600  172637  003060                 LDD     FPZERO,AC2                      ;ZEROES -> FIRB<7:6>="10"
015604  174337  002656                 STD     AC3,MFAC1                       ;FIRB<7:6>="11" -> MFAC1
015610  105737  002645                 TSTB    LPTITE                          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
015614  001365                         BNE     63$                             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                                               ;
015616  040437  002656                 BIC     R4,MFAC1                        ;ZERO UNWANTED BITS OF RESULT,
015622  104425  003070  002656         CMP64M  ,FPZEAP,MFAC1                   ;COMPARE (EXPD):(RCVD), FOR EQ/NE
015630  001401                         BEQ     .+4                             ;BR IF AGREE
015632  104006                         ERROR   06                              ;SIGNAL ERROR ... IF NOT
                                        ;"FSPAD ADDRS ERROR; R[DF]"
                                        ;       RCVD ACC = RECEIVED ACC AFTER SEQ, EXP'D TO BE (125,AP,AP,AP)
                                        ;                       NOT (0,0,0,0)
                                                                ;


                                        ;;*****************************************************************
                                        ;*TEST 41       FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDF"
                                        ;
                                        ;       THIS TEST CHECKS THE F/D-MODE WRITE.SECT.FSPAD[CD] LOGIC, AND
                                        ;       THE FSPAD.SECT[CD] FORCE-ADDRESS-17 LOGIC, USING THE FOLLOWING:
                                        ;
                                        ;       LDF:     (-CONVSP) * (F.MODE) -> (FORCE.ADRS.17).FSPAD.SECT[CD]
                                        ;              (-UCONVSP) * (F.MODE) -> (-WRITE).FSPAD.SECT[CD]
                                        ;
                                        ;       ----------
                                        ;       REGISTER/LOCATION USE:
                                        ;
                                        ;       AC0     -INPUT DATA, F/D-MODE
                                        ;       AC3     -OUTPUT DATA, F/D-MODE
                                        ;
                                        ;       MFAC0+  -AC0 IN MEMORY
                                        ;       MFAC1+  -AC3 IN MEMORY
                                        ;
                                        ;       ----------
                                        ;       MODULE/ERROR INFO:
                                        ;
                                        ;       FNUA/K8
                                        ;         CROM/LATCHES, JREG/FUA
                                        ;
                                        ;       FEXP/K9
                                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, FSPAD.WRITE/ENABLE(F/D)
                                        ;
                                        ;       FMUL/K10
                                        ;         [PREVIOUSLY VERIFIED]
                                        ;
                                        ;       FALU/K11
                                        ;         FSPAD.WRITE.ENB/ADDR(F/D.MODE)
                                        ;
```

```
                                ;
                                ;.*****************************************************************
015634  000004                TST41:  SCOPE
                                ;
015636  170127  040240                LDFPS   #040240                 ;INTR-DISABLE/D-MODE/TRUNCATE
015642  172437  003004                LDD     FPAL1P,AC0              ;(052525,052525,052525,052525) -> AC0[F,B,C,D]
                                ;
015646  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                        ;--------ERROR-LOOP-ENTERS-HERE--------
                                ;
015650  172737  003024                LDD     FPALTN,AC3              ;(125252,125252,125252,125252) -> AC3[A,B,C,D]
015654  170001                        SETF                            ;ENTER F-MODE
015656  172700                63$:    LDF     AC0,AC3                 ;AC0[A,B,0,0] -> AC3[A,B,-,-]
015660  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
015664  001374                        BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                ;
015666  170011                        SETD                            ;BACK TO D-MODE
015670  174037  002646                STD     AC0,MFAC0               ;GET AC0[A,B,C,D] -> MFAC0
015674  174337  002656                STD     AC3,MFAC1               ;GET AC3[A,B,C,D] -> MFAC1
                                ;
                                        ;*SEE THAT AC0[A,B,C,D] WAS KEPT INTACT*
015700  104425  002646  003004        CMP64M  MFAC0,FPALTP            ;= (052525,052525,052525,052525) ??
015706  001401                        BEQ     10$                     ;BR IF OK
015710  104057                        ERROR   57                      ;ELSE ERROR
                                        ;"FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
                                        ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                        ;       RCVD AC3 = 64.BIT DATA STORED AFTER F(->)D MODE CONVERT
                                ;
                                        ;*SEE THAT OUTPUT ACC WAS WRITTEN AS SPECIFIED*
015712  104425  002656  003074  10$:    CMP64M  MFAC1,FPAPAN            ;= (052525,052525,125252,125252) ??
015720  001401                        BEQ     TST42           ;;      ;NEXT TEST IF ALL OK
015722  104060                        ERROR   60                      ;ELSE ERROR
                                        ;"FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
                                        ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                        ;       RCVD AC3 = 64.BIT DATA STORED AFTER F(->)D MODE CONVERT
```

```
                                ;;*****************************************************************
                                ;*TEST 42        FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCFD"
                                ;
                                ;       THIS TEST CHECKS THE F/D-MODE WRITE.SECT.FSPAD[CD] LOGIC, AND
                                ;       THE FSPAD.SECT[CD] FORCE-ADDRESS-17 LOGIC, USING THE FOLLOWING:
                                ;
                                ;       STCFD:    (-CONVSP) * (F.MODE) -> (FORCE.ADRS.17).FSPAD.SECT[CD]
                                ;                 (UCONVSP) * (F.MODE) -> (WRITE).FSPAD.SECT[CD]
                                ;
                                ;       ----------
                                ;       REGISTER/LOCATION USE:
                                ;
                                ;       AC0     -INPUT DATA, F/D-MODE
                                ;       AC3     -OUTPUT DATA, F/D-MODE
                                ;
                                ;       MFAC0+  -AC0 IN MEMORY
                                ;       MFAC1+  -AC3 IN MEMORY
                                ;
```

```
                                ;       ----------
                                ;       MODULE/ERROR INFO:
                                ;
                                ;       FNUA/K8
                                ;         CROM/LATCHES, JREG/BUA
                                ;
                                ;       FEXP/K9
                                ;         CROM/LATCHES, BUT<2:0>-LOGIC, FSPAD.WRITE/ENABLE(F/D)
                                ;
                                ;       FMUL/K10
                                ;         [PREVIOUSLY VERIFIED]
                                ;
                                ;       FALU/K11
                                ;         FSPAD.WRITE.ENB/ADDR(F/D.MODE)
                                ;
                                ;
                                ;;*****************************************************************
015724  000004                  TST42:  SCOPE
                                ;
015726  170127  040240                  LDFPS   #040240                 ;INTR-DISABLE/D-MODE/TRUNCATE
015732  172437  003004                  LDD     FPALTP,AC0              ;(052525,052525,052525,052525) -> AC0[A,B,C,D]
                                ;
015736  104406                  ERRPNT                                  ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                ;
015740  172737  003024                  LDD     FPALTN,AC3              ;(125252,125252,125252,125252) -> AC3[A,B,C,D]
015744  170001                  SETF                                    ;ENTER F-MODE
015746  176003          63$:    STCFD   AC0,AC3                         ;AC0[A,B,0,0] -> AC3[A,B,C,D]
015750  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
015754  001374                  BNE     63$                             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                ;
015756  170011                  SETD                                    ;BACK TO D-MODE
015760  174037  002646                  STD     AC0,MFAC0               ;GET AC0[A,B,C,D] -> MFAC0
015764  174337  002656                  STD     AC3,MFAC1               ;GET AC3[A,B,C,D] -> MFAC1
                                ;
                                ;*SEE THAT AC0[A,B,C,D] WAS KEPT INTACT*
015770  104425  002646  003004          CMP64M  MFAC0,FPALTP            ;= (052525,052525,052525,052525) ??
015776  001401                  BEQ     10$                             ;BR IF OK
016000  104057                  ERROR   57                              ;ELSE ERROR
                                ;"FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
                                ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
                                ;
                                ;*SEE THAT OUTPUT ACC WAS WRITTEN AS SPECIFIED*
016002  104425  002656  003010  10$:    CMP64M  MFAC1,FPAP00            ;= (052525,052525,000000,000000) ??
016010  001401                  BEQ     TST43           ;;              ;NEXT TEST IF ALL OK
016012  104060                  ERROR   60                              ;ELSE ERROR
                                ;"FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
                                ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
```

```
                                ;;*****************************************************************
                                ;*TEST 43        FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "STCDF"
                                ;
```

```
5285                    ;       THIS TEST CHECKS THE F/D-MODE WRITE.SECT.FSPAD[CD] LOGIC, AND
5286                    ;       THE FSPAD.SECT[CD] FORCE-ADDRESS-17 LOGIC, USING THE FOLLOWING:
5287                    ;
5288                    ;       STCDF:   (-CONVSP) * (D.MODE) -> (-FORCE.ADRS.17).FSPAD.SECT[CD]
5289                    ;                (UCONVSP) * (D.MODE) -> (-WRITE).FSPAD.SECT[CD]
5290                    ;
5291                    ;       ----------
5292                    ;       REGISTER/LOCATION USE:
5293                    ;
5294                    ;       AC0     -INPUT DATA, F/D-MODE
5295                    ;       AC3     -OUTPUT DATA, F/D-MODE
5296                    ;
5297                    ;       MFAC0+  -AC0 IN MEMORY
5298                    ;       MFAC1+  -AC3 IN MEMORY
5299                    ;
5300                    ;       ----------
5301                    ;       MODULE/ERROR INFO:
5302                    ;
5303                    ;       FNUA/K8
5304                    ;         CROM/LATCHES, JREG/BUA
5305                    ;
5306                    ;       FEXP/K9
5307                    ;         CROM/LATCHES, BUT<2:0>-LOGIC, FSPAD.WRITE/ENABLE(F/D)
5308                    ;
5309                    ;       FMUL/K10
5310                    ;         [PREVIOUSLY VERIFIED]
5311                    ;
5312                    ;       FALU/K11
5313                    ;         FSPAD.WRITE.ENB/ADDR(F/D MODE)
5314                    ;
5315                    ;
5316                    ;*****************************************************************
5317  016014  000004    TST43:  SCOPE
5318                    ;
5319  016016  170127  040240    LDFPS   #040240                 ;INTR-DISABLE/D-MODE/TRUNCATE
5320  016022  172437  003004    LDD     FPALTP,AC0              ;(052525,052525,052525,052525) -> AC0[A,B,C,D]
5321                    ;
5322  016026  104406            ERRPT                           ;DONT CHANGE DATA IN ERROR LOOP
5323                    ;       ---- ----ERROR-LOOP-ENTERS-HERE--------
5324                    ;
5325  016030  172737  003024    LDD     FPALTN,AC3              ;(125252,125252,125252,125252) -> AC3[A,B,C,D]
5326  016034  176003    63$:    STCDF   AC0,AC3                 ;AC0[A,B,C,D] -> AC3[A,B,-,-]
5327  016036  105737  002645    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5328  016042  001374            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5329                    ;
5330  016044  174037  002646    STD     AC0,MFAC0               ;GET AC0[A,B,C,D] -> MFAC0
5331  016050  174337  002656    STD     AC3,MFAC1               ;GET AC3[A,B,C,D] -> MFAC1
5332                    ;
5333                    ;       ;*SEE THAT AC0[A,B,C,D] WAS KEPT INTACT*
5334  016054  104425  002646  003004    CMP64M  MFAC0,FPALTP    ;= (052525,052525,052525,052525) ??
5335  016062  001401            BEQ     10$                     ;BR IF OK
5336  016064  104057            ERROR   57                      ;ELSE ERROR
5337                    ;       :"FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
5338                    ;               HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
5339                    ;               RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
5340                    ;
```

```
                                        ;*SEE THAT OUTPUT ACC WAS WRITTEN AS SPECIFIED*
016066  104425  002656  003074  10$:    CMP64M  ,MFAC1,FPAPAN           ;= (052525,052525,125252,125252) ??
016074  001401                          BEQ     TST44           ;;      ;NEXT TEST IF ALL OK
016076  104060                          ERROR   60                      ;ELSE ERROR
                                        ;"FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
                                        ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                        ;       RCVD AC3 = 64.BIT DATA STORED AFTER F(->)D MODE CONVERT



                                ;;*****************************************************************
                                ;*TEST 44        FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCDF"
                                ;
                                ;       THIS TEST CHECKS THE F/D-MODE WRITE.SECT.FSPAD[CD] LOGIC, AND
                                ;       THE FSPAD.SECT[CD] FORCE-ADDRESS-17 LOGIC, USING THE FOLLOWING:
                                ;
                                ;       LDCDF:    (CONVSP) * (F.MODE) -> (-FORCE.ADRS.17).FSPAD.SECT[CD]
                                ;                (-UCONVSP) * (F.MODE) -> (-WRITE).FSPAD.SECT[CD]
                                ;
                                ;       ----------
                                ;       REGISTER/LOCATION USE:
                                ;
                                ;       AC0     -INPUT DATA, F/D-MODE
                                ;       AC3     -OUTPUT DATA, F/D-MODE
                                ;
                                ;       MFAC0+  -AC0 IN MEMORY
                                ;       MFAC1+  -AC3 IN MEMORY
                                ;
                                ;       ----------
                                ;       MODULE/ERROR INFO:
                                ;
                                ;       FNUA/K8
                                ;         CROM/LATCHES, JREG/BUA
                                ;
                                ;       FEXP/K9
                                ;         CROM/LATCHES, BUT<2:0>-LOGIC, FSPAD.WRITE/ENABLE(F/D)
                                ;
                                ;       FMUL/K10
                                ;         [PREVIOUSLY VERIFIED]
                                ;
                                ;       FALU/K11
                                ;         FSPAD.WRITE.ENB/ADDR(F/D.MODE)
                                ;
                                ;*****************************************************************
016100  000004                  TST44:  SCOPE
                                        ;
016102  170127  040240                  LDFPS   #040240                 ;INTR-DISABLE/D-MODE/TRUNCATE
016106  172437  003004                  LDD     FPALTP,AC0              ;(052525,052525,052525,052525) -> AC0[A,B,C,D]
                                        ;
016112  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                        ;--------ERROR-LOOP-ENTERS-HERE--------
                                        ;
016114  172737  003024                  LDD     FPALTN,AC3              ;(125252,125252,125252,125252) -  AC3[A,B,C,D]
016120  170001                          SETF                            ;ENTER F-MODE
016122  177700                  63$:    LDCDF   AC0,AC3                 ;AC0[A,B,C,D] -> AC3[A,B,-,-]
```

```
5397  016124  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5398  016130  001374                  BNE     .-4                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5399                                                                  ;
5400  016132  170011                  SETD                            ;BACK TO D-MODE
5401  016134  174037  002646          STD     AC0,MFAC0               ;GET AC0[A,B,C,D] -> MFAC0
5402  016140  174337  002656          STD     AC3,MFAC1               ;GET AC3[A,B,C,D] -> MFAC1
5403                                                                  ;
5404                                  ;*SEE THAT AC0[A,B,C,D] WAS KEPT INTACT*
5405  016144  104425  002646  003004  CMP64M   ,MFAC0,FPALTP          ;= (052525,052525,052525,052525) ??
5406  016152  001401                  BEQ     10$                     ;BR IF OK
5407  016154  104057                  ERROR   57                      ;ELSE ERROR
5408                                  ;"FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
5409                                  ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
5410                                  ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
5411                                                                  ;
5412                                  ;*SEE THAT OUTPUT ACC WAS WRITTEN AS SPECIFIED*
5413  016156  104425  002656  003074  10$:    CMP64M   ,MFAC1,FPAPAN          ;= (052525,052525,125252,125252) ??
5414  016164  001401                  BEQ     TST45           ;;      ;NEXT TEST IF ALL OK
5415  016166  104060                  ERROR   60                      ;ELSE ERROR
5416                                  ;"FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
5417                                  ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
5418                                  ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
5419
5420
5421
5422                          ;;**************************************************************
5423                          ;*TEST 45       FRACTION, FSPAD[CD].WRITE/ADDRS-FORCE: USING "LDCFD"
5424                          ;
5425                          ;       THIS TEST CHECKS THE F/D-MODE WRITE.SECT.FSPAD[CD] LOGIC, AND
5426                          ;       THE FSPAD.SECT[CD] FORCE-ADDRESS-17 LOGIC, USING THE FOLLOWING:
5427                          ;
5428                          ;       LDCFD:    (CONVSP) * (D.MODE) -> (FORCE.ADRS.17).FSPAD.SECT[CD]
5429                          ;              (-UCONVSP) * (D.MODE) -> (WRITE).FSPAD.SECT[CD]
5430                          ;
5431                          ;       ----------
5432                          ;       REGISTER/LOCATION USE:
5433                          ;
5434                          ;       AC0     -INPUT DATA, F/D-MODE
5435                          ;       AC3     -OUTPUT DATA, F/D-MODE
5436                          ;
5437                          ;       MFAC0+  -AC0 IN MEMORY
5438                          ;       MFAC1+  -AC3 IN MEMORY
5439                          ;
5440                          ;       ----------
5441                          ;       MODULE/ERROR INFO:
5442                          ;
5443                          ;       FNUA/K8
5444                          ;         CROM/LATCHES, JREG/BUA
5445                          ;
5446                          ;       FEXP/K9
5447                          ;         CROM/LATCHES, BUS<2:0>-LOGIC, FSPAD.WRITE/ENABLE(F/D)
5448                          ;
5449                          ;       FMUL/K10
5450                          ;         [PREVIOUSLY VERIFIED]
5451                          ;
5452                          ;       FALU/K11
```

```
                                        ;         FSPAD.WRITE.ENB/ADDR(F/D.MODE)
                                        ;
                                        ;
                                        ;.*****************************************************************
016170  000004                          TST45:  SCOPE
                                                                        ;
016172  170127  040240                          LDFPS   #040240         ;INTR-DISABLE/D-MODE/TRUNCATE
016176  172437  003004                          LDD     FPALTP,AC0      ;(052525,052525,052525,052525) -> AC0[A,B,C,D]
                                                                        ;
016202  104406                                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                                        ;
016204  172737  003024                          LDD     FPALTN,AC3      ;(125252,125252,125252,125252) -> AC3[A,B,C,D]
016210  177700                          63$:    LDCFD   AC0,AC3         ;AC0[A,B,C,D] -> AC3[A,B,C,D]
016212  105737  002645                          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
016216  001374                                  BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                                        ;
016220  174037  002646                          STD     AC0,MFAC0       ;GET AC0[A,B,C,D] -> MFAC0
016224  174337  002656                          STD     AC3,MFAC1       ;GET AC3[A,B,C,D] -> MFAC1
                                                                        ;
                                                ;*SEE THAT AC0[A,B,C,D] WAS KEPT INTACT*
016230  104425  002646  003004                  CMP64M  MFAC0,FPALTP    ;= (052525,052525,052525,052525) ??
016236  001401                                  BEQ     10$             ;BR IF OK
016240  104057                                  ERROR   57              ;ELSE ERROR
                                                ;"FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
                                                ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                                ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
                                                                        ;
                                                ;*SEE THAT OUTPUT ACC WAS WRITTEN AS SPECIFIED*
016242  104425  002656  003010  10$:             CMP64M  MFAC1,FPAP00    ;= (052525,052525,000000,000000) ??
016250  001401                                  BEQ     TST46           ;;      ;NEXT TEST IF ALL OK
016252  104060                                  ERROR   60              ;ELSE ERROR
                                                ;"FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
                                                ;       HFPP AC0 = SRC 64.BIT DATA FOR OPERATION = (AP,AP,AP,AP)
                                                ;       RCVD AC3 = 64.BIT DATA STORED AFTER F<->D MODE CONVERT
```

```
5491                    ;*****************************************************************
5492                    ;*TEST 46       SHIFTER/NORMK, WITH "MNS"
5493                    ;
5494                    ;       THIS TEST VERIFIES THE "NORMK" AR<59:51> ENCODER, AND SOME OF THE
5495                    ;       "SHIFTER" LOGIC.  THE "MNS" INSTRUCTION IS USED TO PLACE A
5496                    ;       VALUE IN THE AR<59:51> BITS, AND THEN THE FOLLOWING IS PERFORMED:
5497                    ;
5498                    ;         1) ESPAD[AC1] = EADJ/NORMK( AR<59:51> ) PLUS ESPAD[AC0]
5499                    ;            THIS FUNCTION CHECKS THE NORMK/EADJ ENCODING.
5500                    ;
5501                    ;         2) FSPAD[AC1] = NORM.SHIFTED( FSPAD[AC0] )
5502                    ;            THIS FUNCTION CHECKS THE SHIFTER/NORMK.CONTROL, AND SOME
5503                    ;            BASIC SHIFTING (SHIFTER, UPPER 16. BITS ONLY)
5504                    ;
5505                    ;       ----------
5506                    ;       REGISTER/LOCATION USE:
5507                    ;
5508                    ;       AC0     -MNS/ SHIFT INPUT DATA AC
5509                    ;       AC1     -MNS/ SHIFT OUTPUT DATA (EXPNT, FRAC) AC
5510                    ;
5511                    ;       MFAC0+  -MNS/ SHIFT INPUT DATA (AC0 COPY)
5512                    ;       MFAC1+  -MNS/ EXP'D SHIFT OUTPUT DATA (EXPNT, FRAC) (AC1 COPY)
5513                    ;       MFAC2+  -MNS/ RCV'D SHIFT OUTPUT DATA
5514                    ;
5515                    ;       R5      -DATA TABLE PTR
5516                    ;
5517                    ;       ----------
5518                    ;       MODULE/ERROR INFO:
5519                    ;
5520                    ;       FNUA/K8
5521                    ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ,
5522                    ;         F.BUS.E-DRIVERS/ENABLES
5523                    ;
5524                    ;       FEXP/K9
5525                    ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(NORMK/RIF)
5526                    ;
5527                    ;       FMUL/K10
5528                    ;         [PREVIOUSLY VERIFIED]
5529                    ;
5530                    ;       FALU/K11
5531                    ;         SHIFTER(A/B-LEVELS)-DATA/CNTL, FALU(B.SIDE.DATA),
5532                    ;         FSPAD/FALU/AR(<59:58>), NORMK.ENCODER
5533                    ;
5534                    ;*****************************************************************
5535  016254  000004    TST46:  SCOPE
5536                    ;
5537  016256  170127  040040       LDFPS   #040040                 ;F-MODE/TRUNCATE
5538  016262  012705  016362       MOV     #40$,R5                 ;PTR TO DATA TABLE
5539  016266  005037  002650       CLR     MFAC0+2                 ;WORD-B INIT/EXP'D ARE
5540  016272  005037  002660       CLR     MFAC1+2                 ;  ZEROES
5541  016276  105037  002624       CLRB    FPLENF                  ;F-MODE TYPEOUTS
5542                    ;
5543                           ;*DATA TABLE LOOP ENTERS HERE*
5544  016302  005237  002640  10$: INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
5545  016306  012537  002646       MOV     (R5)+,MFAC0+0           ;GET WORD-A OF MNS/AC0
5546  016312  100442               BMI     TST47           ::      ;NEXT TEST IF ALL DONE
```

```
5547  016314  012537  002656                MOV     (R5)+,MFAC1+0           ;GET WORD-A OF MNS/AC1
5548  016320  172437  002646                LDF     MFAC0,AC0               ;GET EXP=0, FRAC BITS INTO AC0
5549                                                                          ;
5550  016324  104406                       ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5551                                        ;--------ERROR-LOOP-ENTERS-HERE--------
5552                                                                          ;
5553  016326  172537  003004                LDF     FPALTP,AC1              ;INIT AC1 TO (AP,AP,-,-)
5554  016332  170004                63$:    MNS                             ;DO F[AC1]<-NORMK(F[AC0]-LEFT2)
5555                                                                          ;   E[AC1]<-0-PLUS-EADJ(F[AC0]-L2)
5556  016334  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5557  016340  001374                       BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5558                                                                          ;
5559  016342  174137  002666                STF     AC1,MFAC2               ;STORE RESULT IN MEMORY
5560                                                                          ;
5561  016346  023737  002656  002666        CMP     MFAC1+0,MFAC2+0         ;(EXP'D-WORDA)=(RCV'D-WORDA)?
5562  016354  001752                       BEQ     10$                     ;YES - GO FOR NEXT LOOP
5563  016356  104061                       ERROR   61                      ;ELSE ERROR IN EXPNT OR FRAC
5564                                        :"NORMK-EADJ/SHFTR ERR"
5565                                        ;       HFPP AC0 = INITIAL 32.BIT AC0 FOR MNS (FROM TABLE)
5566                                        ;       EXPD AC1 = EXPECTED AC1 AFTER MNS, EXPNT=EADJ/FRAC=(200,0)
5567                                        ;       RCVD AC1 = RECEIVED AC1 AFTER MNS
5568                                        ;
5569  016360  000750                       BR      10$                     ;TRY AGAIN
5570                                  ;
5571                                  ;////////////////////////////////////////////////////////////////////////
5572                                  ;
5573                                  ;DATA FOR ABOVE TEST:
5574                                  ;
5575                                  ;       -INITIAL-AC0-   -EXP'D-AC1-     EXPNT   FRACTION
5576                                  ;       EXPNT/FRAC      EXPNT/FRAC      ;EADJ   -SHIFT-
5577
5578  016362  000100  000200          40$:    00000+100,      00200+000       ;+1     RITE-1
5579
5580  016366  000040  000000                  00000+040,      00000+000       ; 0     NONE-0
5581
5582  016372  000020  077600                  00000+020,      77600+000       ;-1     LEFT-1
5583
5584  016376  000010  077400                  00000+010,      77400+000       ;-2     LEFT-2
5585
5586  016402  000004  077200                  00000+004,      77200+000       ;-3     LEFT-3
5587
5588  016406  000002  077000                  00000+002,      77000+000       ;-4     LEFT-4
5589
5590  016412  000001  077100                  00000+001,      77000+100       ;-4     LEFT-4&NORMK-OVF
5591
5592  016416  177777                        -1      ;DONE
5593
5594
5595
5596                                  ;;*****************************************************************
5597                                  ;*TEST 47        SHIFTER, LEFT(2+4) OF (200,0,0,0)
5598                                  ;
5599                                  ;       THIS TEST PERFORMS A FULL 64. BIT "LEFT/NORMALIZATION" SHIFT THRU THE
5600                                  ;       SHIFT TREE, USING THE "MNS" INSTRUCTION.  THE DATA EMPLOYED IS:
5601                                  ;
5602                                  ;         (200.000000.000000.000000)
```

```
5603
5604                    ;       WHICH SHOULD COME OUT TO BE ALL ZEROES, AFTER SHIFTING.  THIS TEST IS
5605                    ;       LOOKING FOR ANY FLOATING LINES IN THE SHIFT TREE.
5606                    ;
5607                    ;       ----------
5608                    ;       REGISTER/LOCATION USE:
5609                    ;
5610                    ;       AC0     -MNS/ SHIFT INPUT DATA AC
5611                    ;       AC1     -MNS/ SHIFT OUTPUT DATA (EXPNT, FRAC) AC
5612                    ;
5613                    ;       MFAC0+  -MNS/ RCV'D SHIFT OUTPUT DATA
5614                    ;
5615                    ;       ----------
5616                    ;       MODULE/ERROR INFO:
5617                    ;
5618                    ;       FNUA/K8
5619                    ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ
5620                    ;
5621                    ;       FEXP/K9
5622                    ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(RES.ROM/NORMK/RIF)
5623                    ;
5624                    ;       FMUL/K10
5625                    ;         [PREVIOUSLY VERIFIED]
5626                    ;
5627                    ;       FALU/K11
5628                    ;         SHIFTER(A/B-LEVELS)-DATA/CNT_, FALU(B.SIDE.DATA),
5629                    ;         FSPAD/FALU/AR(<59:58>,<2:0>', NORMK.ENCODER
5630                    ;
5631                    ;;*****************************************************************
5632 016420  000004     TST47:  SCOPE
5633                    ;
5634 016422  170127  040240       LDFPS   #040240                 ;INTR-DISABL/D-MODE/TRUNC
5635 016426  170400               CLRD    AC0                     ;INPUT = (200,0,0,0)
5636                    ;                                       ;
5637 016430  104406               ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5638                              ;--------ERROR-LOOP-ENTERS-HERE--------
5639                    ;                                       ;
5640 016432  172537  003004       LDD     FPALTP,AC1              ;OUTPUT = (4*052525) @ START
5641 016436  170004     63$:      MNS                             ;((AC0-L2)-L4) -> AC1
5642 016440  105737  002645       TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5643 016444  001374               BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5644                    ;                                       ;
5645 016446  174137  002646       STD     AC1,MFAC0               ;GET RESULT TO MEMORY
5646 016452  104425  002646  003154  CMP64M  ,MFAC0,FPEM4Z       ;=(077000,0,0,0)??
5647 016460  001401               BEQ     TST50           ;;      ;NEXT TEST IF OK
5648 016462  104062               ERROR   62                      ;ELSE ERROR - FLOATING DATA LINE
5649                              ;"SHFTR L(2+4) OF ZERO ERR"
5650                              ;       RCVD AC1 = RECEIVED AC1 AFTER MNS OF (0,0,0,0) IN FRAC
5651                    ;                                       ;
5652
5653
5654
5655                    ;;*****************************************************************
5656                    ;*TEST 50        SHIFTER, RITE(1.-11.) OF (0,0,0,0)
5657                    ;
5659                    ;       THIS TEST PERFORMS A FULL 64. BIT "RIGHT/ALIGNMENT" SHIFT THRU THE
```

```
                        ;       SHIFT TREE, USING THE "MAS" INSTRUCTION.  THE DATA EMPLOYED IS:
                        ;
                        ;         (000.000000.000000.000000)
                        ;
                        ;       WHICH SHOULD COME OUT TO BE ALL ZEROES, AFTER SHIFTING.  THIS TEST IS
                        ;       LOOKING FOR ANY FLOATING LINES IN THE SHIFT TREE.
                        ;
                        ;       ----------
                        ;       REGISTEP/LOCATION USE:
                        ;
                        ;       AC0     -MAS/ SHIFT INPUT DATA AC
                        ;       AC1     -MAS/ SHIFT OUTPUT DATA (EXPNT, FRAC) AC
                        ;
                        ;       MFAC0+  -MAS/ RCV'D SHIFT OUTPUT DATA
                        ;
                        ;       R1      -SHIFT CNTR, 1.->11.
                        ;       R2      -MAS/EXPNT SHIFT CODE (= SHIFT.CNTR-1+4)
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(RES.ROM)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         SHIFTER(A/B-LEVELS)-DATA/CNTL, FALU(B.SIDE.DATA),
                        ;         FSPAD/FALU/AR(<59:58>,<2:0>), NORMK.ENCODER
                        ;
                        ;*****************************************************************
016464  000004          TST50:  SCOPE
                                                                ;
016466  170127  040240          LDFPS   #040240                 ;INTR-DISABL/D-MODE/TRUNC
016472  012701  000013          MOV     #11.,R1                 ;SHIFT CTR, R11.->R1.
016476  012702  000016          MOV     #14.,R2                 ;EXPNT VALUE = (SHIFT-1)+4
                                                                ;
                                ;*DATA LOOP ENTERS HERE*
016502  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
016506  170400                  CLRD    AC0                     ;(200,0,0,0) -> FSPAD[0]
016510  170401                  CLRD    AC1                     ;ZAP SIGN[1] TO (0)
016512  176402                  LDEXP   R2,AC0                  ;SET FSPAD[0]<5:0> = SHIFT-CODE+4
016514  170004                  MNS                             ;(FSPAD[0]-L6) -> FSPAD[1], E[0]-4 -> E[1]
016516  174100                  STD     AC1,AC0                 ;SET FSPAD[0]<59:00> = ZEROES
                                                                ;
016520  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                                ;
016522  172537  003004          LDD     FPALTP,AC1              ;PRESET AC1=(4*052525)
016526  170007          63$:    MAS                             ;(AC0-R11./R1.) -> AC1
016530  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
016534  001374                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
```

```
5715
5716  016536  174137  002646                  STD     AC1,MFAC0               ;GET RESULT IN MEMORY
5717  016542  104425  002646  003154          CMP64M  ,MFAC0,FPEM4Z           ;=(077000,0,0,0)??
5718  016550  001401                          BEQ     20$                     ;BR IF OK
5719  016552  104063                          ERROR   63                      ;ELSE ERROR - FLOATING DATA LINE
5720                                          ;"SHFTR R(11.-1.) OF ZERO ERR"
5721                                          ;       SHIFT    = SHIFT VALUE EMPLOYED, 1.->11., 01->13
5722                                          ;       RCVD AC1 = RECEIVED AC1 AFTER MAS OF (0,0,0,0) WITH ABOVE SHIFT
5723                                                                              ;
5724  016554  005302                  20$:    DEC     R2                      ;NEXT SHIFT VALUE CODE
5725  016556  077127                          SOB     R1,10$                  ;COUNT LOOP R11.-R1.
5726
5727
5728
5729                                  ;;*****************************************************************
5730                                  ;*TEST 51        FRACTION, FALU FSPAD.IN.MUX BIT(59:58)
5731                                  ;
5732                                  ;       THIS TEST CHECKS THAT BITS<59:58> OF THE FRACTION DATAPATH
5733                                  ;       ARE SET TO "01" ON A "LDF" INSTRUCTION.  THIS SHOULD BE DONE
5734                                  ;       AUTOMATICALLY BY THE "FSPAD.IN.MUX" ON THE "FALU" BOARD.
5735                                  ;
5736                                  ;       THE "HIDDEN BITS" <59:58> ARE EXAMINED VIA USING THE "MAS"
5737                                  ;       INSTRUCTION AND THE SHIFTER TO SHIFT/RIGHT.3.  NOTE THAT AN
5738                                  ;       ERROR IN THIS TEST MOST LIKELY IS DUE TO A FAULT IN
5739                                  ;       BITS<59:58> OF THE FSPAD.IN.MUX; HOWEVER, A FAULT IN
5740                                  ;       THE UPPER BITS OF THE SHIFTER COULD ALSO GENERATE SUCH AN ERROR.
5741                                  ;
5742                                  ;       ----------
5743                                  ;       REGISTER/LOCATION USE:
5744                                  ;
5745                                  ;       MFAC0+  -INITIAL DATA PATTERN
5746                                  ;       MFAC1+  -EXPECTED RESULT, AFTER "MAS"
5747                                  ;       MFAC2+  -RECEIVED RESULT (AC1) AFTER "MAS"
5748                                  ;
5749                                  ;       AC0     -INPUT "LDF" ACCUM
5750                                  ;       AC1     -OUTPUT ACCUM FOR RESULT, AFTER SHIFT
5751                                  ;
5752                                  ;       ----------
5753                                  ;       MODULE/ERROR INFO:
5754                                  ;
5755                                  ;       FNUA/K8
5756                                  ;         CR   LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ
5757                                  ;
5758                                  ;       FEXP/K9
5759                                  ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(RES.ROM)
5760                                  ;
5761                                  ;       FMUL/K10
5762                                  ;         [PREVIOUSLY VERIFIED]
5763                                  ;
5764                                  ;       FALU/K11
5765                                  ;         FSPAD.IN.MUX(<59:58>), SHIFTER(A/B-LEVELS)-DATA/CNTL,
5766                                  ;         FALU(B.SIDE.DATA),
5767                                  ;         FSPAD/FALU/AR(<59:58>,<2:0>), NORMK.ENCODER
5768
5769                                  ;*****************************************************************
5770  016560  000004                  TST51:  SCOPE
```

```
5771                                                                    ;
5772  016562  170127  040040                LDFPS   #040040          ;INIT-DISAB/E-MODE/TRUNC
5773  016566  012737  000400  002646        MOV     #000400+000,MFAC0+0 ;INIT DATA, EXPNT FOR "MAS"-SHIFT/RITE-3
5774  016574  005037  002650                CLR     MFAC0+2             ;
5775  016600  012737  077220  002656        MOV     #077200+020,MFAC1+0 ;EXP'D DATA, AFTER MAS; EXPNT/EADJ=(-3)
5776  016606  005037  002660                CLR     MFAC1+2             ;
5777                                                                    ;
5778  016612  104406                        ERRPNT                      ;DONT CHANGE DATA IN ERROR LOOP
5779                                        ;--------ERROR-LOOP-ENTERS-HERE--------
5780                                                                    ;
5781  016614  172437  002646        63$:    LDF     MFAC0,AC0           ;INITIAL DATA LOAD THRU FSPAD.IN.MUX
5782  016620  105737  002645                TSTB    LPTITE              ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5783  016624  001373                        BNE     63$                 ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5784                                                                    ;
5785  016626  170401                        CLRF    AC1                 ;ZAP OUTPUT AC
5786  016630  170007                        MAS                         ;DO THE (AC0)-RITE-3 -> AC1
5787  016632  174137  002666                STF     AC1,MFAC2           ;SAVE IN MEMORY
5788                                                                    ;
5789  016636  104426  002656  002666        CMP32M  ,MFAC1,MFAC2        ;(RECEIVED) = (EXPECTED) ??
5790  016644  001401                        BEQ     TST52       ;;      ;NEXT TEST IF AGREE
5791  016646  104071                        ERROR   71                  ;HIDDEN BITS<59:58> INSERT ERROR
5792                                        ;"FSPAD.IN.MUX INBUF-PORT ERR"
5793                                        ;       HFPP AC0 = LOADED DATA, 2W
5794                                        ;       EXPD AC1 = EXPECTED DATA FROM AC1 AFTER MAS/RITE-3
5795                                        ;       RCVD AC1 = RECEIVED DATA FROM AC1 AFTER MAS
5796
5797
5798
5799                            ;*****************************************************************
5800                            ;*TEST 52       FPINIT/FP.EMIT.F/CLRX EXACT.ZERO "01" IN BIT(59:58)
5801                            ;
5802                                  THIS TEST CHECKS THAT BITS<59:58> OF THE FSPAD[17] "EXACT.ZERO"
5803                                  WERE SET TO "01" IN THE "FPINIT" FLOWS.  THIS SHOULD BE DONE
5804                                  VIA THE "FP.EMIT.F" FACILITY.
5805                            ;
5806                                  THE "HIDDEN BITS" <59:58> ARE EXAMINED VIA USING THE "MAS"
5807                                  INSTRUCTION AND THE SHIFTER TO SHIFT/RIGHT.3.  NOTE THAT AN
5808                                  ERROR IN THIS TEST MOST LIKELY IS DUE TO A FAULT IN
5809                                  BITS<59:58> OF THE FSPAD.IN.MUX; HOWEVER, A FAULT IN
5810                                  THE UPPER BITS OF THE SHIFTER COULD ALSO GENERATE SUCH AN ERROR.
5811                            ;
5812                                  ----------
5813                                  REGISTER/LOCATION USE:
5814                            ;
5815                                  MFAC0+  -EXPECTED RESULT, AFTER "MAS"
5816                                  MFAC1+  -RECEIVED RESULT (AC1) AFTER "MAS"
5817                            ;
5818                                  AC0     -INPUT "CLRF" ACCUM
5819                                  AC1     -OUTPUT ACCUM FOR RESULT, AFTER SHIFT
5820                            ;
5821                                  ----------
5822                                  MODULE/ERROR INFO:
5823                            ;
5824                                  FNUA/K8
5825                                    CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ, FP.EMIT.F
5826                            ;
```

```
5827                                    ;       FEXP/K9
5828                                    ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(RES.ROM)
5829                                    ;
5830                                    ;       FMUL/K10
5831                                    ;         [PREVIOUSLY VERIFIED]
5832                                    ;
5833                                    ;       FALU/K11
5834                                    ;         FSPAD.IN.MUX(<59:58>), SHIFTER(A/B-LEVELS)-DATA/CNTL,
5835                                    ;         FALU(B.SIDE.DATA), FSPAD/FALU/AR(<59:58>,<2:0>), NORMK.ENCODER
5836                                    ;
5837                                    ;;*****************************************************************
5838  016650  000004                    TST52:  SCOPE
5839                                    ;
5840  016652  170127  040240           ;       LDFPS   #040240                 ;INTR-DISAB/D-MODE/TRUNC
5841  016656  012737  077220  002646   ;       MOV     #077200+020,MFAC0+0     ;EXP'D DATA, AFTER MAS; EXPNT/EADJ=(-3)
5842  016664  005037  002650           ;       CLR     MFAC0+2                 ;
5843  016670  005037  002652           ;       CLR     MFAC0+4                 ;
5844  016674  005037  002654           ;       CLR     MFAC0+6                 ;
5845                                    ;
5846  016700  104406                    ;       ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5847                                    ;       ;--------ERROR-LOOP-ENTERS-HERE--------
5848                                    ;
5849  016702  170400                    ;63$:   CLRD    AC0                     ;READ EXACT.ZERO
5850  016704  176427  177602           ;       LDEXP   #<2-200>,AC0            ;MAS/EXPNT FOR RITE.3
5851                                    ;                                       ; FSPAD[AC0].OR.EXACT-ZERO -> FSPAD[AC0]
5852  016710  105737  002645           ;       TSTB    LPTITE                  ;IF TIGHT LOOP-ON ERROR SET, THEN HANG IN LOOP
5853  016714  001372                    ;       BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5854                                    ;
5855  016716  170007                    ;       MAS                             ;DO THE (AC0)-RITE-3 -> AC1
5856  016720  174137  002656           ;       STD     AC1,MFAC1               ;SAVE IN MEMORY
5857                                    ;
5858  016724  104425  002646  002656   ;       CMP64M  ,MFAC0,MFAC1            ;(RECEIVED) = (EXPECTED) ??
5859  016732  001401                    ;       BEQ     TST53           ;;      ;NEXT TEST IF AGREE
5860  016734  104110                    ;       ERROR   110                     ;HIDDEN BITS<59:58> INSERT ERROR
5861                                    ;"FPINIT/CLRD/LDEXP BIT<59:58> INSERT ERR"
5862                                    ;       EXPD AC1 = EXPECTED DATA FROM AC1 AFTER MAS/RITE-3
5863                                    ;       RCVD AC1 = RECEIVED DATA FROM AC1 AFTER MAS
5864
5865
5866
5867                                    ;;*****************************************************************
5868                                    ;*TEST 53       SHIFTER, RITE(4,5,6,7/1,9)[MAS] RIPPLE-A-1
5869                                    ;
5870                                    ;       THIS TEST RIPPLES A "1" THRU THE SHIFT TREE (USING THE "MAS"
5871                                    ;       INSTRUCTION) TESTING FOR THE CORRECT SHIFT VALUE DECODE (VIA
5872                                    ;       PRE.SHIFT.RESIDUE.ROM), AND ANY STUCK LOW/HIGH DATA LINES.
5873                                    ;
5874                                    ;       FIRST THE A.SHIFT.LEVEL IS HELD CONSTANT (AT +4), AND
5875                                    ;       THE B.SHIFT.LEVEL VARIED FROM +0/+1/+2/+3.  PART TWO
5876                                    ;       HOLDS THE B.SHIFT.LEVEL CONSTANT (AT +1), AND VARIES THE
5877                                    ;       A.SHIFT.LEVEL AS +0/+8.
5878                                    ;
5879                                    ;       ----------
5880                                    ;       REGISTER/LOCATION USE:
5881                                    ;
5882                                    ;       AC0     -MAS/ SHIFT INPUT DATA AC
```

```
                                ;       AC1     -MAS/ SHIFT OUTPUT DATA (EXPNT, FRAC) AC
                                ;
                                ;       MFAC0+  -MAS/ SHIFT INPUT DATA (AC0 COPY)
                                ;       MFAC1+  -MAS/ EXP'D SHIFT OUTPUT DATA (EXPNT, FRAC) (AC1 COPY)
                                ;       MFAC2+  -MAS/ RCV'D SHIFT OUTPUT DATA
                                ;
                                ;       R1      -MAS/ EXPNT.SHIFT.CODE (= SHIFT.VALUE-1)
                                ;       R2      -HIDDEN.BIT.MASK, WORD.A
                                ;       R3      -   "       "     WORD.B
                                ;       R4      -SHIFT VALUE, (RIGHT.SHIFT)
                                ;       R5      -DATA TABLE PTR
                                ;
                                ;       ----------
                                ;       MODULE/ERROR INFO:
                                ;
                                ;       FNUA/K8
                                ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ
                                ;
                                ;       FEXP/K9
                                ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(RES.ROM)
                                ;
                                ;       FMUL/K10
                                ;         [PREVIOUSLY VERIFIED]
                                ;
                                ;       FALU/K11
                                ;         SHIFTER(A/B-LEVELS)-DATA/CNTL, FALU(B.SIDE.DATA),
                                ;         FSPAD/FALU/AR(<59:58>,<2:0>), NORMK.ENCODER
                                ;
                                ;*****************************************************************
016736  000004                TST53:  SCOPE
                                                                        ;
016740  012737  000062  001342          MOV     #50.,$TIMES             ;50. ITER. OF THIS TEST
016746  170127  040240                  LDFPS   #040240                 ;INTR-DISABL/D-MODE/TRUNC
016752  012705  017144                  MOV     #40$,R5                 ;PTR TO DATA TABLE
                                                                        ;
                                        ;*DATA TABLE LOOP ENTERS HERE*
016756  012504                10$:      MOV     (R5)+,R4                ;GET NEXT SHIFT VALUE
016760  100530                          BMI     TST54           ;;      ;DONE WHEN = -1
016762  012502                          MOV     (R5)+,R2                ;GET WORD (A,B) HIDDEN-BIT
016764  012503                          MOV     (R5)+,R3                ;  MASK (AFTER SHIFT)
016766  172427                          LDD     (PC)+,AC0               ;MFAC0 IS INITIAL DATA:
016770  000100                          .WORD   100                     ;  FRAC(300,0,0,0)
016772  174037  002646                  STD     AC0,MFAC0               ;
016776  170437  002656                  CLRD    MFAC1                   ;MFAC1 IS EXPECTED RESULT:
017002  012537  002656                  MOV     (R5)+,MFAC1+0           ;  FRAC(A-TABLE,B-TABLE,0,0)
017006  012537  002660                  MOV     (R5)+,MFAC1+2           ;
017012  010401                          MOV     R4,R1                   ;EXPNT SHIFT VALUE
017014  005301                          DEC     R1                      ;  = (SHIFT-1)
                                                                        ;
                                        ;*LOOP ON THIS SHIFT VALUE, RIPPLE-A-1 IN FRAC<59:00>*
017016  005237  002640          11$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
017022  172437  002646                  LDD     MFAC0,AC0               ;FRAC<59:00>=01*MEM<57:03>*000
017026  176401                          LDEXP   R1,AC0                  ;EXP<5:0>=SHIFT CODE
017030  050237  002656                  BIS     R2,MFAC1+0              ;INSERT SHIFTED-HIDDEN-BIT
017034  050337  002660                  BIS     R3,MFAC1+2              ;  IN EXPECTED FRAC, WORD-A/B
                                                                        ;
```

```
5939  017040  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
5940                                          ;--------ERROR-LOOP-ENTERS-HERE--------
5941                                          ;
5942  017042  172537  003004                  LDD     FPALTP,AC1              ;PRESET OUTPUT=(4*052525)
5943  017046  170007                  63$:    MAS                             ;(AC0-SHIFTED)->AC1
5944  017050  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
5945  017054  001374                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
5946                                          ;
5947  017056  174137  002666                  STD     AC1,MFAC2               ;GET RESULT IN MEMORY
5948  017062  104423  002666                  FIXFRA  ,MFAC2                  ;(000) -> SIGN/EXPNT
5949                                          ;                               ;  INSERT FRAC<59:58> FROM "EADJ"
5950  017066  104425  002656  002666          CMP64M  ,MFAC1,MFAC2            ;(EXP'D) = (RCV'D)??
5951  017074  001401                          BEQ     20$                     ;BR IF OK
5952  017076  104064                          ERROR   64                      ;ELSE MAS-SHIFT RIPPLE-A-1 ERR
5953                                          ;"SHFTR, MAS-RITE RIPPLE-A-1 ERR"
5954                                          ;       SHIFT    = SHIFT VALUE EMPLOYED; ALL ARE RIGHT SHIFTS
5955                                          ;       HFPP AC0 = 64.BIT INITIAL AC0 FOR MAS SHIFT
5956                                          ;       EXPD AC1 = 64.BIT EXPEC'D DATA AFTER ABOVE MAS SHIFT EMPLOYED
5957                                          ;       RCVD AC1 = 64.BIT RECEIVED SHIFTED VALUE
5958                                          ;
5959                                          ;*NEXT SHIFTED VALUE*
5960  017100  032737  000001  002654  20$:    BIT     #BIT0,MFAC0+6           ;ANY MORE?
5961  017106  001323                          BNE     10$                     ;BR IF NOT - NEXT TABLE ENTRY
5962  017110  042737  177600  002656          BIC     #177600,MFAC1+0         ;ZAP SIGN/EXPNT OF EXP'D
5963  017116  040237  002656                  BIC     R2,MFAC1+0              ;  AND SHIFTED-HIDDEN-BIT
5964  017122  040337  002660                  BIC     R3,MFAC1+2              ;
5965  017126  104430  177777  002646          ASH64I  ,-1,MFAC0               ;INIT DATA RITE-1 (64 BITS)
5966  017134  104430  177777  002656          ASH64I  ,-1,MFAC1               ;EXP'D DATA RITE-1 (64 BITS)
5967  017142  000725                          BR      11$                     ;NEXT
5968                                  ;
5969                                  ;////////////////////////////////////////////////////////////////////////////////
5970                                  ;       DATA FOR ABOVE TEST:
5971                                  ;
5972                                  ;       RIGHT   FRAC-MASK       EXP'D-DATA      ;   A   ;   B   ;
5973                                  ;       SHIFT   WORD(A,B)       WORD(A,B)       ;SHFTR;SHFTR;
5974
5975  017144  000004  000010  000000  40$:    4,      010,000000,     004,000000      ;  +4   ;  +0   ;
5976  017152  000004  000000
5977
5978  017156  000005  000004  000000          5,      004,000000,     002,000000      ;  +4   ;  +1   ;
5979  017164  000002  000000
5980
5981  017170  000006  000002  000000          6.      002,000000,     001,000000      ;  +4   ;  +2   ;
5982  017176  000001  000000
5983
5984  017202  000007  000001  000000          7.      001.000000,     000,100000      ;  +4   ;  +3   ;
5985  017210  000000  100000
5986
5987  017214  000001  000100  000000          1.      100,000000.     040,000000      ;  +0   ;  +1   ;
5988  017222  000040  000000
5989
5990  017226  000011  000000  040000          9.,     000,040000,     000,020000      ;  +8   ;  +1   ;
5991  017234  000000  020000
5992
5993  017240  177777                          -1      ;<DONE>
5994
```

```
5995
5996
5997                    ;;*****************************************************
5998                    ;*TEST 54       SHIFTER, LEFT(2+(1,2,3,4))[MNS] RIPPLE-A-1
5999                    ;
6000                    ;       THIS TEST RIPPLES A "1" THRU THE SHIFT TREE (USING THE "MNS"
6001                    ;       INSTRUCTION) TESTING FOR THE CORRECT SHIFT VALUE DECODE (VIA
6002                    ;       NORMK/NORMALIZE-SHIFT ENCODING), AND ANY STUCK LOW/HIGH DATA LINES.
6003                    ;
6004                    ;       FIRST THE A.SHIFT.LEVEL IS HELD CONSTANT (AT +0), AND
6005                    ;       THE B.SHIFT.LEVEL VARIED FROM +0/+1/+2/+3.  PART TWO
6006                    ;       HOLDS THE B.SHIFT.LEVEL CONSTANT (AT +0/+3), AND VARIES THE
6007                    ;       A.SHIFT.LEVEL AS +4/-4.  THIS RANGES THRU THE FULL NORMALIZATION
6008                    ;       SHIFT VALUES OF +1/0/-1/-2/-3/-4.
6009                    ;
6010                    ;       ----------
6011                    ;       REGISTER/LOCATION USE:
6012                    ;
6013                    ;       AC0     -MNS/ SHIFT INPUT DATA AC
6014                    ;       AC1     -MNS/ SHIFT OUTPUT DATA (EXPNT, FRAC) AC
6015                    ;
6016                    ;       MFAC0+  -MNS/ SHIFT INPUT DATA (AC0 COPY)
6017                    ;       MFAC1+  -MNS/ EXP'D SHIFT OUTPUT DATA (EXPNT, FRAC) (AC1 COPY)
6018                    ;       MFAC2+  -MNS/ RCV'D SHIFT OUTPUT DATA
6019                    ;
6020                    ;       R2      -NORMK SHIFT SELECT BIT<5:1>
6021                    ;       R4      -SHIFT VALUE, (LEFT[+]/RIGHT[-])
6022                    ;       R5      -DATA TABLE PTR
6023                    ;
6024                    ;       ----------
6025                    ;       MODULE/ERROR INFO:
6026                    ;
6027                    ;       FNUA/K8
6028                    ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A,B-SELECT), EADJ
6029                    ;
6030                    ;       FEXP/K9
6031                    ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHIFTER.CNTL(NORMK/RIF)
6032                    ;
6033                    ;       FMUL/K10
6034                    ;         [PREVIOUSLY VERIFIED]
6035                    ;
6036                    ;       FALU/K11
6037                    ;         SHIFTER(A/B-LEVELS)-DATA/CNTL, FALU(B.SIDE.DATA),
6038                    ;         FSPAD/FALU/AR(<59:58>,<2:0>), NORMK.ENCODER
6039                    ;
6040                    ;;*****************************************************************
6041  017242  000004    TST54:  SCOPE
6042                    ;
6043  017244  012737  000062  001342        MOV     #50.,$TIMES             ;50. ITER. OF THIS TEST
6044  017252  170127  040240              LDFPS   #040240                 ;INTR-DISAB/D-MODE/TRUNC
6045  017256  012705  017424              MOV     #40$,R5                 ;PRT TO DATA TABLE
6046                                                                        ;
6047                                      ;*DATA TABLE LOOP ENTERS HERE*
6048  017262  012504    10$:    MOV     (R5)+,R4                ;GET NEXT SHIFT VALUE
6049  017264  020427  052525              CMP     R4,#AP                  ;END OF TABLE?
6050  017270  001500                      BEQ     TST55           ;;      ;DONE WHEN = 052525
```

```
6051  017272  172427                          LDD     (PC)+,AC0               ;MFAC1 IS EXPECTED RESULT:
6052  017274  000100                          .WORD   100                     ;  FRAC(300,0,0,0)
6053  017276  174037  002656                  STD     AC0,MFAC1               ;
6054  017302  170437  002646                  CLRD    MFAC0                   ;MFAC0 IS INITIAL DATA:
6055  017306  012537  002646                  MOV     (R5)+,MFAC0+0           ;  FRAC (TABLE,0,0,0)
6056  017312  012502                          MOV     (R5)+,R2                ;NORMK BIT FOR MNS SHIFT SELECT
6057                                                                          ;
6058                                          ;*LOOP ON THIS SHIFT VALUE, RIPPLE-A-1 IN FRAC<59:00>*
6059  017314  005237  002640          11$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
6060  017320  050237  002646                  BIS     R2,MFAC0+0              ;INSERT NORMK SHFT SELECT BIT
6061  017324  172437  002646                  LDD     MFAC0,AC0               ;INTO AC0
6062                                                                          ;
6063  017330  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
6064                                          ;--------ERROR-LOOP-ENTERS-HERE--------
6065                                                                          ;
6066  017332  172537  003004                  LDD     FPALTP,AC1              ;PRESET OUTPUT=(4*052525)
6067  017336  170004                  63$:    MNS                             ;(AC0-SHIFTED)->AC1
6068  017340  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
6069  017344  001374                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
6070                                                                          ;
6071  017346  174137  002666                  STD     AC1,MFAC2               ;GET RESULT IN MEMORY
6072  017352  042737  177600  002666          BIC     #^C177,MFAC2+0          ;ZAP SIGN/EXPNT OF RCV'D
6073                                                                          ;
6074  017360  104425  002656  002666          CMP64M  MFAC1,MFAC2             ;(EXP'D) = (RCV'D)??
6075  017366  001401                          BEQ     20$                     ;BR IF OK
6076  017370  104065                          ERROR   65                      ;ELSE MNS-SHIFT RIPPLE-A-1 ERR
6077                                          ;"SHFTR, MNS-LEFT/RITE RIPPLE-A-1 ERR"
6078                                          ;       SHIFT    = SHIFT VALUE EMPLOYED; LEFT(+)/RIGHT(-) SHIFTS
6079                                          ;       HFPP AC0 = 64.BIT INITIAL AC0 FOR MNS SHIFT
6080                                          ;       EXPD AC1 = 64.BIT EXPEC'D DATA AFTER ABOVE MNS SHIFT EMPLOYED
6081                                          ;       RCVD AC1 = 64.BIT RECEIVED SHIFTED VALUE
6082                                                                          ;
6083                                          ;*GET NEXT SHIFTED VALUE*
6084  017372  032737  000001  002654  20$:     BIT     #BIT0,MFAC0+6           ;ANY MORE?
6085  017400  001330                          BNE     10$                     ;BR IF NOT - NEXT TABLE ENTRY
6086  017402  040237  002646                  BIC     R2,MFAC0+0              ;ZAP NORMK SELECT BIT
6087  017406  104430  177777  002646          ASH64I  .-1,MFAC0               ;INIT DATA RITE-1 (64 BITS)
6088  017414  104430  177777  002656          ASH64I  .-1,MFAC1               ;EXP'D DATA RITE-1 (64 BITS)
6089  017422  000734                          BR      11$                     ;NEXT
6090                                                                          ;
6091                                          ;////////////////  . ////// ////////// .//// /// ///// ////
6092                                          ;DATA FOR ABOVE TEST:
6093                                          ;
6094                                          ;SHIFT   INIT-DATA       NORMK-SHIFT
6095                                          ;-> R-   F WORD-A)       SELECT BIT
6096                                          ;
6097  017424  000000  000020  000040  40$:     0.      020.            040
6098                                          ;
6099  017432  000002  000004  000010          2.      004.            010
6100                                          ;
6101  017440  000003  000002  000004          3.      002.            004
6102                                          ;
6103  017446  000001  000010  000020          1.      010.            020
6104                                          ;
6105  017454  000004  000001  000002          4.      001.            002
6106
```

```
 6107  017462  177777  000040  000100          -1,      040,            100
 6108
 6109  017470  052525                          AP       ;<DONE>
 6110
 6111
 6112
```

```
                        ;;*****************************************************************
                        ;*TEST 55       FALU/FEXP, F/D-R/T MODE SELECT
                        ;
                        ;       THIS TEST USES THE "MNS" INSTRUCTION TO EXERCISE THE F(32.)/D(64.)
                        ;       BIT ROUND/TRUNCATE LOGIC ON THE FALU/FEXP MODULES.
                        ;
                        ;       THE TEST FIRST PRELOADS AC0<59:03> WITH 4*(052525) AS BACKGROUND
                        ;       DATA PATTERN.  AN F.MODE/D.MODE "CLEAR" IS THEN DONE, TO ZERO THE
                        ;       APPROPRIATE SECTION OF THE ACC.  THE "MNS" INSTRUCTION IS THEN USED
                        ;       TO SHIFT LEFT (2+4)=(6) THE RESULT OF :
                        ;
                        ;         AR<59:35/03> <- FSPAD[AC0]<59:35/03>-PLUS-0(R/T) [LEFT-6]
                        ;
                        ;       THIS BRINGS THE ROUND BITS (F AND D) INTO VIEW.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       AC0     -MNS/ F/D & R/T INPUT DATA AC
                        ;       AC1     -MNS/ F/D & R/T OUTPUT DATA (EXPNT, FRAC) AC
                        ;
                        ;       MFAC0+  -MNS/ EXP'D F/D & R/T OUTPUT DATA (EXPNT, FRAC) :AC1 COPY)
                        ;       MFAC1+  -MNS/ RCV'D F/D & R/T OUTPUT DATA
                        ;
                        ;       R1      -FPS BEFORE MNS, F/D MODES, R/T MODES
                        ;       R4      -(TEMP)
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, ROUND/TRUNC.CNTL,
                        ;         SHIFTER(PRE.SHFT.L3/CNTL,RES.ROM)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         FALU(F/D.MODE-ROUND.BIT.LOGIC), SHIFTER(LEFT3)
                        ;
                        ;*****************************************************************
017472  000004          TST55:  SCOPE
                        ;
017474  012705  017600          MOV     #40$,R5                 ;PTR TO DATA TABLE
                        ;
                        ;*DATA TABLE LOOP ENTERS HERE*
017500  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
017504  012501                  MOV     (R5)+,R1                ;GET NEXT F/D, R/T FPS VALUE
017506  100461                  BMI     TST56           ::      ;DONE WHEN = -1
017510  012704  002646          MOV     #MFAC0+0,R4             ;PTR
017514  012524                  MOV     (R5)+,(R4)+             ;GET EXP'D AC1/AFTER
017516  012524                  MOV     (R5)+,(R4)+             ;
```

```
                        MOV     (R5)+,(R4)+             ;
                        MOV     (R5)+,(R4)+             ;
017520  012524          LDFPS   #040240                 ;INTR.DISAB/D-MODE/TRUNC
017522  012524          LDD     FPALTN,AC0              ;INPUT ACC IS 4*(125252) BEFORE CLRF/D
017524  170127  040240  LDD     FPALTP,AC1              ;OUTPUT AC IS 4*(052525) BEFORE MNS-F/D
017530  172437  003024  LDFPS   R1                      ;SETUP F/D-MODE, R/T-MODE
017534  172537  003004                                  ;
017540  170101          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
                        ;--------ERROR-LOOP-ENTERS-HERE--------
                                                        ;
017542  104406  63$:    CLRD    AC0                     ;F.OR.D MODE
017544  170400          MNS                             ;DO AR <- AC0[F/D],[R/T]-LEFT-2
017546  170004          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
017550  105737  002645  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
017554  001373                                          ;
                                                        ;
017556  170011          SETD                            ;STORE 64. BITS
017560  174137  002656  STD     AC1,MFAC1               ;GET OUTPUT
017564  104425  002646  002656  CMP64M  ,MFAC0,MFAC1    ;(EXP'D) = (RCV'D) ???
017572  001742          BEQ     10$                     ;BR IF AGREE
017574  104106          ERROR   106                     ;ELSE ERROR
                        ;"FEXP/FALU FPS F-D &/+ R-T MODE ERR"
                        ;       -FPS-- = FPS, WITH F/D MODE, R/T MODE
                        ;       EXPD AC1 = EXPECTED AC1, AFTER F/D, R/T
                        ;       RCVD AC1 = RECEIVED AC1
                                                        ;
017576  000740          BR      10$                     ;NEXT
                                                        ;
                        ;
                        ;////////////////////////////////////////////////////////////////////////
                        ;
                        ;       DATA FOR ABOVE TEST
                        ;
        000000          F=000   ;BIT7=0
        000200          D=200   ;BIT7=1
                                ;
        000000          R=000   ;BIT5=0
        000040          T=040   ;BIT5=1
                        ;
                        ;
                        ;       ---FPS----  ----EXP'D--AC1--AFTER------
017600  040040  077000  000000  40$:    040000+F+T, 077000,000000,052525,052525 ;32. BITS AND TRUNCATE
017606  052525  052525
                                                        ;
017612  040240  077000  000000          040000+D+T, 077000,000000,000000,000000 ;64. BITS AND TRUNCATE
017620  000000  000000
                                                        ;
017624  040000  077000  000040          040000+F+R, 077000,000040,052525,052525 ;32. BITS AND F.ROUND.BIT
017632  052525  052525
                                                        ;
017636  040200  077000  000000          040000+D+R, 077000,000000,000000,000040 ;64. BITS AND D.ROUND.BIT
017644  000000  000040
                                                        ;
017650  177777                          -1      ;<DONE>
```

```
6225
6226
```

```
                        ;;*****************************************************************
                        ;*TEST 56       FRACTION, FALU ADD/CARRY LOGIC WITH "ADDD"
                        ;
                        ;       THIS TEST CHECKS THE FRACTION ALU ("FALU") AND ITS ASSOCIATED
                        ;       CARRY LOOKAHEAD LOGIC.  THE TEST IS PERFORMED USING "ADDD" (64.BITS)
                        ;       WITH EXPONENTS ALWAYS EQUAL (THUS NO PRE.SHIFT ALIGNMENT IS REQUIRED).
                        ;
                        ;       EITHER "DIFFPRCO" (SUBTRACT) OR "SUMPRCO" (ADD) PATHS ARE
                        ;       FOLLOWED THRU THE FP11-E ADD/SUBTRACT FLOWS.
                        ;
                        ;       BOTH THE ADD AND SUBTRACT FUNCTIONS ARE CHECKED VIA USING OPERANDS
                        ;       WITH LIKE (ADD) AND UNLIKE (SUBTRACT) SIGNS.
                        ;
                        ;       THERE IS ALWAYS A ONE STEP (RIGHT.1) NORMALIZATION SHIFT
                        ;       PERFORMED AFTER THE OPERATION (DUE TO CHOICE OF OPERANDS).
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -INITIAL OPERAND "A", FROM TABLE
                        ;       MFAC1+  -INITIAL OPERAND "B", FROM TABLE
                        ;       MFAC2+  -EXPECTED SUM, FROM TABLE
                        ;       MFAC3+  -RECEIVED SUM FROM HFP
                        ;
                        ;       AC0     -COPY OF OPERAND "A"
                        ;       AC1     -COPY OF OPERAND "B"
                        ;       AC2     -HFP SUM
                        ;
                        ;       R3      -(TEMP)
                        ;       R4      -(PTR)
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A.PLUS.B,A.MINUS.B)
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         FSPAD[TEMP'S], FALU(ADD/SUB,CARRY.LOOKAHEAD)
                        ;
                        ;
                        ;*****************************************************************
017652  000004          TST56:  SCOPE
                                                                ;
017654  170127  040240          LDFPS   #040240                 ;INTR-DISAB/D-MODE/TRUNC
017660  012705  017760          MOV     #40$,R5                 ;DATA TABLE PTR
                                                                ;
                        ;*DATA LOOP ENTERS HERE*
017664  005237  002640  10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
```

```
6283  017670  012704  002646          MOV     #MFAC0+0,R4             ;PTR TO RESULT AREA
6284  017674  012703  000013          MOV     #11.,R3                 ;WORD CNTR
6285  017700  012524                  MOV     (R5)+,(R4)+             ;GET A WORD
6286  017702  001424                  BEQ     30$                     ;IF ALL ZERO, WE'RE DONE
6287  017704  012524          11$:    MOV     (R5)+,(R4)+             ;MOVE THE REST OF THE DATA
6288  017706  077302                  SOB     R3,11$                  ;LOOP
6289                                                                  ;
6290  017710  172437  002646          LDD     MFAC0,AC0               ;GET OPERAND "A"
6291  017714  172537  002656          LDD     MFAC1,AC1               ;GET OPERAND "B"
6292                                                                  ;
6293  017720  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
6294                                  ;--------ERROR-LOOP-ENTERS-HERE--------
6295                                                                  ;
6296  017722  174102          63$:    STD     AC1,AC2                 ;COPY QUICKLY
6297  017724  172200                  ADDD    AC0,AC2                 ;(AC0)-PLUS-(AC2) -> AC2
6298  017726  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
6299  017732  001373                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
6300                                                                  ;
6301  017734  174237  002676          STD     AC2,MFAC3               ;COPY TO MEMORY
6302                                                                  ;
6303                                  ;*NOW CHECK ANSWERS*
6304  017740  104425  002666  002676  CMP64M   MFAC2,MFAC3            ;(EXPECTED) = (RECEIVED) ??
6305  017746  001746                  BEQ     10$                     ;BR IF AGREE
6306  017750  104070                  ERROR   70                      ;ELSE FALU/ARITH ERROR
6307                                  ;"FALU ADD/CARRY RESULT ERR"
6308                                  ;       HFPP AC0 = OPERAND "A" FROM AC0
6309                                  ;       HFPP AC1 = OPERAND "B" FROM AC1
6310                                  ;       EXPD AC2 = EXPECTED SUM OF A-PLUS-B
6311                                  ;       RCVD AC2 = RECEIVED SUM OF A-PLUS-B
6312                                                                  ;
6313  017752  000744                  BR      10$                     ;NEXT LOOP
6314                                  ;
6315  017754  000137  020532  30$:     JMP     FALU01                  ;EXIT THE HARD WAY
6316                                  ;
6317                                  ;
6318                                  ;-------------------------------- ----------------------------------------
6319                                  ;
6320                                  ;       DATA TABLE FOR ABOVE TEST:
6321                                  ;
6322  017760                  40$:
6323                                  ;
6324                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6325                                  ;<010000000.00000000000000000.00000000000000000.00000000000000000>      OPERAND A
6326  017760  040200  000000  000000          .WORD   040200+0,0,0,0  ;(NO - PRE.SHIFT)
6327  017766  000000
6328                                  ;
6329                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6330                                  ;<010000000.00000000000000000.00000000000000000.00000000000000000>      OPERAND B
6331  017770  040200  000000  000000          .WORD   040200+0,0,0,0  ;(NO - PRE.SHIFT)
6332  017776  000000
6333                                  ;
6334                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6335                                  ;<010000000.00000000000000000.00000000000000000.00000000000000000>      RESULT
6336  020000  040400  000000  000000          .WORD   040400+0,0,0,0  ;(RIGHT-1 - NORM.SHIFT)
6337  020006  000000
6338                                  ;
```

```
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND A
020010  040252  125252  125252                .WORD   040200+52,125252,125252,125252   ;(NO - PRE.SHIFT)
020016  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND B
020020  040252  125252  125252                .WORD   040200+52,125252,125252,125252   ;(NO - PRE.SHIFT)
020026  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      RESULT
020030  040452  125252  125252                .WORD   040400+52,125252,125252,125252   ;(RIGHT-1 - NORM.SHIFT)
020036  125252
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND A
020040  040325  052525  052525                .WORD   040200+125,52525,52525,52525   ;(NO - PRE.SHIFT)
020046  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND B
020050  040325  052525  052525                .WORD   040200+125,52525,52525,52525   ;(NO - PRE.SHIFT)
020056  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      RESULT
020060  040525  052525  052525                .WORD   040400+125,52525,52525,52525   ;(RIGHT-1 - NORM.SHIFT)
020066  052525
                                        ;
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND A
020070  040325  052525  052525                .WORD   040200+125,52525,52525,52525   ;(NO - PRE.SHIFT)
020076  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND B
020100  040252  125252  125252                .WORD   040200+52,125252,125252,125252   ;(NO - PRE.SHIFT)
020106  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010111111.111111111111111111.111111111111111111.111111111111111111>      RESULT
020110  040477  177777  177777                .WORD   040400+77,177777,177777,177777   ;(RIGHT-1 - NORM.SHIFT)
020116  177777
```

N11


```
                                        ;
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND A
020120  040252  125252  125252           .WORD   040200+52,125252,125252,125252   ;(NO - PRE.SHIFT)
020126  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND B
020130  040325  052525  052525           .WORD   040200+125,52525,52525,52525   ;(NO - PRE.SHIFT)
020136  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010111111.111111111111111111.111111111111111111.111111111111111111>      RESULT
020140  040477  177777  177777           .WORD   040400+77,177777,177777,177777   ;(RIGHT-1 - NORM.SHIFT)
020146  177777
                                        ;
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND A
020150  040325  052525  052525           .WORD   040200+125,52525,52525,52525   ;(NO - PRE.SHIFT)
020156  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<011010101.010101010101010101.010101010101010101.010101010101010101>      OPERAND B
020160  140325  052525  052525           .WORD   140200+125,52525,52525,52525   ;(RIGHT-1 - NORM.SHIFT)
020166  052525
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<000000000.000000000000000000.000000000000000000.000000000000000000>      RESULT
020170  000000  000000  000000           .WORD   000000+0,0,0,0   ;(RIGHT-1 - NORM.SHIFT)
020176  000000
                                        ;
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND A
020200  040252  125252  125252           .WORD   040200+52,125252,125252,125252   ;(NO - PRE.SHIFT)
020206  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<010101010.101010101010101010.101010101010101010.101010101010101010>      OPERAND B
020210  140252  125252  125252           .WORD   140200+52,125252,125252,125252   ;(RIGHT-1 - NORM.SHIFT)
020216  125252
                                        ;
                                        ;/59 ------------------> BIT.RANGE ----------------------> 03\
                                        ;<000000000.000000000000000000.000000000000000000.000000000000000000>      RESULT
020220  000000  000000  000000           .WORD   000000+0,0,0,0   ;(RIGHT-1 - NORM.SHIFT)
```

```
6451  020226  000000
6452                                  ;
6453                                  ;
6454                                  ;-------------------------------------------------------------------------------
6455                                  ;
6456                                  ;
6457                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6458                                  ;<010001111.0000111100001111.0000111100001111.0000111100001111>      OPERAND A
6459  020230  040217  007417  007417          .WORD   040200+17,7417,7417,7417   ;(NO - PRE.SHIFT)
6460  020236  007417
6461                                  ;
6462                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6463                                  ;<010001111.0000111100001111.0000111100001111.0000111100001111>      OPERAND B
6464  020240  040217  007417  007417          .WORD   040200+17,7417,7417,7417   ;(NO - PRE.SHIFT)
6465  020246  007417
6466                                  ;
6467                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6468                                  ;<010001111.0000111100001111.0000111100001111.0000111100001111>      RESULT
6469  020250  040417  007417  007417          .WORD   040400+17,7417,7417,7417   ;(RIGHT-1 - NORM.SHIFT)
6470  020256  007417
6471                                  ;
6472                                  ;
6473                                  ;-------------------------------------------------------------------------------
6474                                  ;
6475                                  ;
6476                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6477                                  ;<011110000.1111000011110000.1111000011110000.1111000011110000>      OPERAND A
6478  020260  040360  170360  170360          .WORD   040200+160,170360,170360,170360   ;(NO - PRE.SHIFT)
6479  020266  170360
6480                                  ;
6481                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6482                                  ;<011110000.1111000011110000.1111000011110000.1111000011110000>      OPERAND B
6483  020270  040360  170360  170360          .WORD   040200+160,170360,170360,170360   ;(NO - PRE.SHIFT)
6484  020276  170360
6485                                  ;
6486                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6487                                  ;<011110000.1111000011110000.1111000011110000.1111000011110000>      RESULT
6488  020300  040560  170360  170360          .WORD   040400+160,170360,170360,170360   ;(RIGHT-1 - NORM.SHIFT)
6489  020306  170360
6490                                  ;
6491                                  ;
6492                                  ;-------------------------------------------------------------------------------
6493                                  ;
6494                                  ;
6495                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6496                                  ;<010010110.1001011010010110.1001011010010110.1001011010010111>      OPERAND A
6497  020310  040226  113226  113226          .WORD   040200+26,113226,113226,113227   ;(NO - PRE.SHIFT)
6498  020316  113227
6499                                  ;
6500                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6501                                  ;<010000111.1000011110000111.1000011110000111.1000011110000111>      OPERAND B
6502  020320  040207  103607  103607          .WORD   040200+7,103607,103607,103607   ;(NO - PRE.SHIFT)
6503  020326  103607
6504                                  ;
6505                                  ;/59 ------------------> BIT.RANGE ----------------------> 03\
6506                                  ;<010001111.0000111100001111.0000111100001111.0000111100001111>      RESULT
```

```
 6507  020330  040417  007417  007417          .WORD   040400+17,7417,7417,7417   ;(RIGHT-1 - NORM.SHIFT)
 6508  020336  007417
 6509                                  ;
 6510                                  ;-------------------------------------------------------------------------------
 6511                                  ;
 6512                                  ;
 6513                                  ;
 6514                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6515                                  ;<011101001.011010010110100l.011010010110100l.011010010110l001>      OPERAND A
 6516  020340  040351  064551  064551          .WORD   040200+151,64551,64551,64551   ;(NO - PRE.SHIFT)
 6517  020346  064551
 6518                                  ;
 6519                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6520                                  ;<011111000.011110000l111000.011110000l111000.011110000l111000>      OPERAND B
 6521  020350  040370  074170  074170          .WORD   040200+170,74170,74170,74170   ;(NO - PRE.SHIFT)
 6522  020356  074170
 6523                                  ;
 6524                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6525                                  ;<011110000.111100001111000O.1111000011110000.1111000011110000>      RESULT
 6526  020360  040560  170360  170360          .WORD   040400+160,170360,170360,170360   ;(RIGHT-1 - NORM.SHIFT)
 6527  020366  170360
 6528                                  ;
 6529                                  ;-------------------------------------------------------------------------------
 6530                                  ;
 6531                                  ;
 6532                                  ;
 6533                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6534                                  ;<011001011.010010110100l011.010010110100l011.010010110100l011>      OPERAND H
 6535  020370  040313  045513  045513          .WORD   040200+113,45513.45513,45513   ;(NO - PRE.SHIFT)
 6536  020376  045513
 6537                                  ;
 6538                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6539                                  ;<011010010.110100101101001O.1101001011010010.1101001011010011>      OPERAND B
 6540  020400  040322  151322  151322          .WORD   040200+122,151322,151322,151323   ;(NO - PRE.SHIFT)
 6541  020406  151323
 6542                                  ;
 6543                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6544                                  ;<011001111.000011110000l111.000011110000l111.000011110000l111>      RESULT
 6545  020410  040517  007417  007417          .WORD   040400+117,7417,7417,7417   ;(RIGHT-1 - NORM.SHIFT)
 6546  020416  007417
 6547                                  ;
 6548                                  ;
 6549                                  ;-------------------------------------------------------------------------------
 6550                                  ;
 6551                                  ;
 6552                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6553                                  ;<010110100.101101001011010O.1011010010110100.1011010010110100>      OPERAND A
 6554  020420  040264  132264  132264          .WORD   040200+64,132264,132264,132264   ;(NO - PRE.SHIFT)
 6555  020426  132264
 6556                                  ;
 6557                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
 6558                                  ;<010101101.001011010010110l.001011010010110l.001011010010110l>      OPERAND B
 6559  020430  040255  026455  026455          .WORD   040200+55,26455,26455,26455   ;(NO - PRE.SHIFT)
 6560  020436  026455
 6561                                  ;
 6562                                  ;/59 ------------------> BIT.RANGE ---------------------> 03\
```

```
                                ;<010110000.111100001111000.11110000111100000.11110000111100000>      RESULT
020440  040460  170360  170360          .WORD   040400+60,170360,170360,170360   ;(RIGHT-1 - NORM.SHIFT)
020446  170360
                                ;
                                ;
                                ;-------------------------------------------------------------------------------
                                ;
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<010110100.10110100101101100.10110100101101100.10110100101101101>      OPERAND A
020450  040264  132264  132264          .WORD   040200+64,132264,132264,132265   ;(NO - PRE.SHIFT)
020456  132265
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<011101001.01101001011010001.01101001011010001.01101001011010001>      OPERAND B
020460  040351  064551  064551          .WORD   040200+151,64551,64551,64551   ;(NO - PRE.SHIFT)
020466  064551
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<011001111.000011110000111.000011110000111.000011110000111>      RESULT
020470  040517  007417  007417          .WORD   040400+117,7417,7417,7417   ;(RIGHT-1 - NORM.SHIFT)
020476  007417
                                ;
                                ;
                                ;-------------------------------------------------------------------------------
                                ;
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<011001011.010010110100101l.01001011010010l1.01001011010010l1>      OPERAND A
020500  040313  045513  045513          .WORD   040200+113,45513,45513,45513   ;(NO - PRE.SHIFT)
020506  045513
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<0!0010110.10010110100101l0.10010110100101l0.10010110100101l0>      OPERAND B
020510  040226  113226  113226          .WORD   040200+26,11322(.113226,113226   ;(NO - PRE.SHIFT)
020516  113226
                                ;
                                ;/59 -------------------> BIT.RANGE ----------------------> 03\
                                ;<010110000.111100001111000.11110000111100000.11110000111100000>      RESULT
020520  040460  170360  170360          .WORD   040400+60,170360,170360,170360   ;(RIGHT-1 - NORM.SHIFT)
020526  170360
                                ;                                ,
                                ;
                                ;-------------- ---------------------------------------------------------------
                                ;
020530  000000                          .WORD   0       ;<END THE TABLE>
                                ;
                                ;
020532                          FALU01:
020532  000400                          BR      TST57           ;;      ;EXIT TO NEXT TEST
                                ;
```

```
                        ;;*****************************************************************
                        ;*TEST 57       IFORK/(ADD+SUB)*M0 "ADDD"*-M0 EXECUTE
                        ;
                        ;       THIS TEST PROCEEDS THRU ALL THE NON-TRIVIAL PATHS OF THE
                        ;       FP11-E "ADDX/SUBX" FLOWS.  DATA HAS BEEN SELECTED (SEE BELOW)
                        ;       THAT GENERATES THE DESIRED RESPONSE FROM THE FP11-E SEQUENCING
                        ;       HARDWARE.
                        ;
                        ;       ALTHOUGH "MODE-0" IS NOT SPECIFICALLY EMPLOYED HERE, THE
                        ;       MICROCODE AND HARDWARE USE THE "FORCE-ADD" SIGNAL TO EVOKE
                        ;       THE SAME RESPONSE AT THE "IFORK" MICROBRANCH.
                        ;
                        ;       THE "FETOPND" SUBRROJTINE FETCHES THE SOURCE DATA, PLACES IT
                        ;       IN AC6, AND THEN GOES TO THE EXECUTION FLOWS.  THIS REQUIRES
                        ;       THAT THE FP11-E "BUT(SUBROUTINE)/JREG/BUT(RETURN)" LOGIC IS
                        ;       FUNCTIONING CORRECTLY.  NOTE ALSO THAT LOGIC ON FNUA/K8 SHOULD
                        ;       FORCE AC[SF]=AC6 FOR NOT(MODE.0) IN SF.  ACTUAL AC[SF]=(0)
                        ;       IN THE FIRB<2:0> FIELD.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       AC0     -AC[DF] OPERAND
                        ;       AC1     -TEMP FOR AC0
                        ;
                        ;       MFAC0+  -AC[SF] OPERAND
                        ;       MFAC1+  -AC[DF] OPERAND
                        ;       MFAC2+  -EXP'D SUM/DIFF OF AC[SF], AC[DF]
                        ;       MFAC3+  -RCV'D SUM/DIFF OF ABOVE
                        ;
                        ;       R0      -PTR TO MFAC0
                        ;       R1      -DATA SET NUMBER
                        ;       R2      -PTR
                        ;       R3      -CNTR
                        ;       R4      -TABLE CNTR
                        ;       R5      -TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(A.PLUS.B,A.MINUS.B)
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, SHFTR.CNTRL(PRESHIFT/NORMK)
                        ;
                        ;       FMUL/K10
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         FSPAD[TEMP'S], FALU(ADD/SUB,CARRY.LOOKAHEAD)
                        ;         SHIFTER(LEFT/RITE), NORMK.OVF
                        ;
                        ;*****************************************************************
020534  000004          TST57:  SCOPE
                                                                ;
```

```
6673  020536  170127  040240                  LDFPS   #040240                 ;INTR-DISAB/D-MODE/TRUNC
6674  020542  012700  002646                  MOV     #MFAC0,R0               ;SETUP PTR FOR ADDD SF
6675  020546  012701  000001                  MOV     #1,R1                   ;DATA SET NUMBER
6676  020552  012704  000032                  MOV     #32,R4                  ;32(8) TABLE ENTRIES
6677  020556  012705  020650                  MOV     #40$,R5                 ;DATA TABLE PTR
6678                                                                          ;
6679                                          ;*DATA LOOP ENTERS HERE*
6680  020562  005237  002640          50$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
6681  020566  012703  000014                  MOV     #12.,R3                 ;3*4 WORDS, OPR-A, OPR-B, EXP'D
6682  020572  012702  002646                  MOV     #MFAC0+0,R2             ;PTR TO DEST.
6683  020576  012522                  51$:    MOV     (R5)+,(R2)+             ;MOVE A WORD
6684  020600  077302                          SOB     R3,51$                  ;LOOP
6685                                                                          ;
6686  020602  172537  002656                  LDD     MFAC1,AC1               ;GET AC[DF]
6687                                                                          ;
6688  020606  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
6689                                          ;--------ERROR-LOOP-ENTERS-HERE--------
6690                                                                          ;
6691  020610  174100                  63$:    STD     AC1,AC0                 ;RESET AC[DF]
6692  020612  172010                          ADDD    (R0),AC0                ;ADDD EXEC, W/ (-MO), SF=0, FETOPND, FORCE-ADD
6693  020614  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
6694  020620  001373                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
6695                                                                          ;
6696  020622  174037  002676                  STD     AC0,MFAC3               ;GET RESULT TO MEMORY
6697                                                                          ;
6698  020626  104425  002666  002676          CMP64M  MFAC2,MFAC3             ;(EXP'D) = (RCV'D) ??
6699  020634  001401                          BEQ     59$                     ;BR IF AGREE
6700  020636  104073                          ERROR   73                      ;ELSE ADDD SUMPATH/DIFFPATH ERROR
6701                                          ;"IFORK/(ADD+SUB)*MO EXECUTE ERR"
6702                                          ;       DATA-SET        = DATA SET NUMBER, 01 -> 32
6703                                          ;       AC6[SF]         = AC[SF] OPERAND, FROM MFAC0
6704                                          ;       AC0[DF]         = AC[DF] OPERAND, FROM AC0/AC1
6705                                          ;       EXPD RESULT     = EXPECTED AC[SF]+AC[DF] SUM/DIFF
6706                                          ;       RCVD RESULT     = RECEIVED SUM/DIFF
6707                                                                          ;
6708  020640  005201                  59$:    INC     R1                      ;DATA SET NUMBER
6709  020642  077431                          SOB     R4,50$                  ;LOOP ON TABLE
6710  020644  000137  022030                  JMP     ADDD01                  ;EXIT THE HARD WAY
6711                                          ;
6712                                          ;////////////////////////////////////////////////////////////////////////
6713                                          ;
6714                                          ;       DATA TABLE FOR ABOVE TEST:
6715                                          ;
6716  020650                          40$:
6717
6718                                          ;--- DIFF.PATH, ER >= ZERO ---
6719
6720                                          ;DATA**
6721  020650  000177  177777  177777  A01$:   S0!000*X!177,177777,177777,177777       ;AC[SF]:  EXPNT=0
6722  020656  177777
6723  020660  177600  177777  000000          S1!377*X!000,177777,000000,177777       ;AC[DF]:  EXPNT#0
6724  020666  177777
6725  020670  177600  177777  000000          S1!377*X!000,177777,000000,177777       ;DIFFPACSZ: AC[DF]=0, ANS=AC[DF]
6726  020676  177777
6727
6728  020700  000177  177777  000000  A02$:   S0!000*X!177,177777,000000,177777       ;AC[SF]:  EXPNT=0
```

```
6729  020706  177777
6730  020710  100177  000000  177777        S1!000*X!177,000000,177777,000000      ;AC[DF]:  EXPNT=0
6731  020716  000000
6732  020720  000000  000000  000000        S0!000*X!000,000000,000000,000000      ;DIFFPACSZ: AC[DF]=0*AC[SF]=0, EXACT ZER
                                                                                                 0
6733  020726  000000
6734
6735  020730  052525  052525  052525  A03$: S0!252*X!125,052525,052525,052525      ;AC[SF]
6736  020736  052525
6737  020740  152525  052525  052525        S1!252*X!125,052525,052525,052525      ;AC[DF]
6738  020746  052525
6739  020750  000000  000000  000000        S0!000*X!000,000000,000000,000000      ;DIFFPRC0: EXPNT=, FRAC=, EXACT ZERO
6740  020756  000000
6741
6742  020760  052577  000000  000000  A04$: S0!252*X!177,000000,000000,000000      ;AC[SF]
6743  020766  000000
6744  020770  152577  000001  000000        S1!252*X!177,000001,000000,000000      ;AC[DF]
6745  020776  000000
6746  021000  144600  000000  000000        S1!223*X!000,000000,000000,000000      ;DIFFPRC0: EXPNT=, F[DF] > F[SF]
6747  021006  000000
6748
6749  021010  052577  000000  000000  A05$: S0!252*X!177,000000,000000,000000      ;AC[SF]
6750  021016  000000
6751  021020  152576  000000  000000        S1!252*X!176,000000,000000,000000      ;AC[DF]
6752  021026  000000
6753  021030  050600  000000  000000        S0!243*X!000,000000,000000,000000      ;DIFFPRC0: EXPNT=, F[DF] < F[SF]
6754  021036  000000
6755
6756  021040  140177  177777  177777  A06$: S1!200*X!177,177777,177777,177777      ;AC[SF]
6757  021046  177777
6758  021050  040377  177777  177777        S0!201*X!177,177777,177777,177777      ;AC[DF]
6759  021056  177777
6760  021060  040177  177777  177777        S0!200*X!177,177777,177777,177777      ;DIFFPRC1: ER=+1, ANS=AC[DF]-AC[SF]
6761  021066  177777
6762
6763  021070  040177  177777  177777  A07$: S0!200*X!177,177777,177777,177777      ;AC[SF]
6764  021076  177777
6765  021100  141377  177777  177777        S1!205*X!177,177777,177777,177777      ;AC[DF]
6766  021106  177777
6767  021110  141367  177777  177777        S1!205*X!167,177777,177777,177777      ;DIFFPRC2: ER=+2/+13, ANS=AC[DF]-AC[SF]
6768  021116  177777
6769
6770  021120  140177  177777  177777  A10$: S1!200*X!177,177777,177777,177777      ;AC[SF]
6771  021126  177777
6772  021130  050177  177777  177777        S0!240*X!177,177777,177777,177777      ;AC[DF]
6773  021136  177777
6774  021140  050177  177777  177377        S0!240*X!177,177777,177377,177777      ;DIFFPRC3: ER=+14/+70, ANS=AC[DF]-AC[SF]
6775  021146  177777
6776
6777  021150  020177  177777  177777  A11$: S0!100*X!177,177777,177777,177777      ;AC[SF]
6778  021156  177777
6779  021160  160125  000000  177777        S1!300*X!125,000000,177777,000000      ;AC[DF]
6780  021166  000000
6781  021170  160125  000000  177777        S1!300*X!125,000000,177777,000000      ;DIFFPRC4: ER=+71/+377, ANS=AC[DF]
6782  021176  000000
6783
```

```
6784
6785
6786                                        ;--- SUM.PATH, ER >= ZERO ---
6787
6788  021200  100177  177777  177777  A12$:  S1!000*X!177,177777,177777,177777      ;AC[SF]:  EXPNT=0
6789  021206  177777
6790  021210  125200  177777  177777         S1!125*X!000,177777,177777,177777      ;AC[DF]:  EXPNT#0
6791  021216  177777
6792  021220  125200  177777  177777         S1!125*X!000,177777,177777,177777      ;SUMPACSZ: AC[DF]=0, ANS=AC[DF]
6793  021226  177777
6794
6795  021230  000125  000000  177777  A13$:  S0!000*X!125,000000,177777,000000      ;AC[SF]:  EXPNT=0
6796  021236  000000
6797  021240  000052  177777  000000         S0!000*X!052,177777,000000,177777      ;AC[DF]:  EXPNT=0
6798  021246  177777
6799  021250  000000  000000  000000         S0!000*X!000,000000,000000,000000      ;SUMPACSZ: AC[DF]=0*AC[SF]=0, EXACT ZERO
6800  021256  000000
6801
6802  021260  152525  000000  052525  A14$:  S1!252*X!125,000000,052525,000000      ;AC[SF]
6803  021266  000000
6804  021270  152452  125252  000000         S1!252*X!052,125252,000000,125252      ;AC[DF]
6805  021276  125252
6806  021300  152677  152525  025252         S1!253*X!077,152525,025252,152525      ;SUMPRC0: ER=0, ANS=AC[DF]+AC[SF]
6807  021306  152525
6808
6809  021310  040052  125252  125252  A15$:  S0!200*X!052,125252,125252,125252      ;AC[SF]
6810  021316  125252
6811  021320  040252  125252  125252         S0!201*X!052,125252,125252,125252      ;AC[DF]
6812  021326  125252
6813  021330  040377  177777  177777         S0!201*X!177,177777,177777,177777      ;SUMPRC1: ER=+1, ANS=AC[DF]+AC[SF]
6814  021336  177777
6815
6816  021340  140052  125252  000000  A16$:  S1!200*X!052,125252,000000,125252      ;AC[SF]
6817  021346  125252
6818  021350  140525  052525  052525         S1!202*X!125,052525,052525,052525      ;AC[DF]
6819  021356  052525
6820  021360  140577  177777  152525         S1!202*X!177,177777,152525,077777      ;SUMPRC2: ER=+2/+13, ANS=AC[DF]+AC[SF]
6821  021366  077777
6822
6823  021370  140000  177777  000000  A17$:  S1!200*X!000,177777,000000,177777      ;AC[SF]
6824  021376  177777
6825  021400  143177  000000  177777         S1!214*X!177,000000,177777,000000      ;AC[DF]
6826  021406  000000
6827  021410  143177  004020  177757         S1!214*X!177,004020,177757,000017      ;SUMPRC3: ER=+14/+70, ANS=AC[DF]+AC[SF]
6828  021416  000017
6829
6830  021420  120177  177777  177777  A20$:  S1!100*X!177,177777,177777,177777      ;AC[SF]
6831  021426  177777
6832  021430  136325  000000  177777         S1!171*X!125,000000,177777,000000      ;AC[DF]
6833  021436  000000
6834  021440  136325  000000  177777         S1!171*X!125,000000,177777,000000      ;SUMPRC4: ER=+71/+377, ANS=AC[DF]
6835  021446  000000
6836
6837
6838                                        ;--- DIFF.PATH, ER < ZERO ---
6839
```

```
6840  021450  025200  177777  000000  A21$:  SO!125*X!000,177777,000000,177777      ;AC[SF]:  EXPNT#0
6841  021456  177777
6842  021460  100177  000000  177777        S1!000*X!177,000000,177777,052525      ;AC[DF]:  EXPNT=0
6843  021466  052525
6844  021470  025200  177777  000000        SO!125*X!000,177777,000000,177777      ;DIFFPAZERO: AC[DF]=0, ANS=AC[SF]
6845  021476  177777
6846
6847  021500  140377  000000  177777  A22$:  S1!201*X!177,000000,177777,000000      ;AC[SF]
6848  021506  000000
6849  021510  040000  000000  177777        SO!200*X!000,000000,177777,000000      ;AC[DF], SHIFTED
6850  021516  000000
6851  021520  140277  000000  077777        S1!201*X!077,000000,077777,100000      ;DIFFPRCM1: ER=-1, ANS=AC[DF]-AC[SF]
6852  021526  100000
6853
6854  021530  042600  177777  000000  A23$:  SO!213*X!000,177777,000000,177777      ;AC[SF]
6855  021536  177777
6856  021540  140177  000000  177777        S1!200*X!177,000000,177777,000000      ;AC[DF], SHIFTED
6857  021546  000000
6858  021550  042600  160036  177741        SO!213*X!000,160036,177741,000037      ;DIFFPRCM2: ER=-2/-13, ANS=AC[DF]-AC[SF]
6859  021556  000037
6860
6861  021560  054177  177777  177777  A24$:  SO!260*X!177,177777,177777,177777      ;AC[SF]
6862  021566  177777
6863  021570  140177  177777  177777        S1!200*X!177,177777,177777,177777      ;AC[DF], SHIFTED
6864  021576  177777
6865  021600  054177  177777  177777        SO!260*X!177,177777,177777,177377      ;DIFFPRCM3: ER=-14/-70, ANS=AC[DF]-AC[SF
                                                                                     ]
6866  021606  177377
6867
6868  021610  066652  177777  000000  A25$:  SO!333*X!052,177777,000000,177777      ;AC[SF]
6869  021616  177777
6870  021620  122200  177777  177777        S1!111*X!000,177777,177777,000000      ;AC[DF], SHIFTED
6871  021626  000000
6872  021630  066652  177777  000000        SO!333*X!052,177777,000000,177777      ;DIFFPRCM4: ER=-71/-377, ANS=AC[SF]
6873  021636  177777
6874
6875
6876                                        ;--- SUM.PATH, ER < ZERO ---
6877
6878  021640  077777  177777  177777  A26$:  SO!377*X!177,177777,177777,177777      ;AC[SF]:  EXPNT#0
6879  021646  177777
6880  021650  000125  000000  177777        SO!000*X:125,000000,177777,000000      ;AC[DF]:  EXPNT=0
6881  021656  000000
6882  021660  077777  177777  177777        SO!377*X!177,177777,177777,177777      ;SUMPAZERO: AC[DF]=0, ANS=AC[SF]
6883  021666  177777
6884
6885  021670  140252  000000  052525  A27$:  S1!201*X!052,000000,052525,000000      ;AC[SF]
6886  021676  000000
6887  021700  140000  000001  052524        S1!200*X!000,000001,052524,000000      ;AC[DF], SHIFTED
6888  021706  000000
6889  021710  140352  000000  177777        S1!201*X!152,000000,177777,000000      ;SUMPRCM1: ER=-1, ANS=AC[DF]+AC[SF]
6890  021716  000000
6891
6892  021720  042177  000000  000000  A30$:  SO!210*X!177,000000,000000,000000      ;AC[SF]
6893  021726  000000
6894  021730  040177  000000  177777        SO!200*X!177,000000,177777,000000      ;AC[DF], SHIFTED
```

```
6895  021736  000000
6896  021740  042177  177400  000377  SO!210*X!177,177400,000377,177400       ;SUMPRCM2: ER=-2/-13, ANS=AC[DF]+AC[SF]
6897  021746  177400
6898
6899  021750  052177  000000  177777  A31$:  SO!250*X!177,000000,177777,052525       ;AC[SF]
6900  021756  052525
6901  021760  040052  000000  000000         SO!200*X!052,000000,000000,000000       ;AC[DF], SHIFTED
6902  021766  000000
6903  021770  052177  000000  177777         SO!250*X!177,000000,177777,177525       ;SUMPRCM3: ER=-14/-70, ANS=AC[DF]+AC[SF]
6904  021776  177525
6905
6906  022000  022652  177777  052525  A32$:  SO!113*X!052,177777,052525,177777       ;AC[SF]
6907  022006  177777
6908  022010  004525  177777  177777         SO!022*X!125,177777,177777,177777       ;AC[DF], SHIFTED
6909  022016  177777
6910  022020  022652  177777  052525         SO!113*X!052,177777,052525,177777       ;SUMPRCM4: ER=-71/-377, ANS=AC[SF]
6911  022026  177777
6912
6913  022030                          ADDD01:
6914  022030  000400                          BR      TST60           ;;      ;NEXT TEST THE HARD WAY
6915
6916
6917
```

```
                        ;;*****************************************************************
                        ;*TEST 60       "DIVF" EXEC,  DIVIDE W/INBUF-AR.SHIFT, FSPAD.SELECT
                        ;
                        ;       THIS TEST PROCEEDS THRU ALL THE NON-TRIVIAL PATHS OF THE
                        ;       FP11-E "DIVF" FLOWS.  DATA HAS BEEN SELECTED (SEE BELOW)
                        ;       THAT GENERATES THE DESIRED RESPONSE FROM THE FP11-E SEQUENCING
                        ;       HARDWARE.
                        ;
                        ;       THE "FETOPND" SUBRROUTINE FETCHES THE SOURCE DATA, PLACES IT
                        ;       IN AC6, AND THEN GOES TO THE EXECUTION FLOWS.  THIS REQUIRES
                        ;       THAT THE FP11-E "BUT(SUBROUTINE)/JREG/BUT(RETURN)" LOGIC IS
                        ;       FUNCTIONING CORRECTLY.  NOTE ALSO THAT LOGIC ON FNUA/K8 SHOULD
                        ;       FORCE AC[SF]=AC6 FOR NOT(MODE.0) IN SF.  ACTUAL AC[SF]=(0)
                        ;       IN THE FIRB<2:0> FIELD.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       AC0     -AC[DF] OPERAND, DIVIDEND
                        ;       AC1     -TEMP FOR AC0
                        ;       AC6     -AC[SF], DIVISOR
                        ;
                        ;       MFAC0+  -AC[SF] OPERAND, DIVISOR
                        ;       MFAC1+  -AC[DF] OPERAND, DIVIDEND
                        ;       MFAC2+  -EXP'D QUOTIENT OF AC[SF], AC[DF]
                        ;       MFAC3+  -RCV'D QUOTIENT OF ABOVE
                        ;
                        ;       R0      -PTR TO MFAC0
                        ;       R1      -DATA SET NUMBER
                        ;       R4      -TABLE CNTR
                        ;       R5      -TABLE PTR
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES, JREG/BUA, FALU.CNTL(DIVIDE),
                        ;         INBUF.A/B.LEFT-SHIFT(DATA/CNTL)
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES, BUT<2:0>-LOGIC, EALU.DATA/CNTL(A.MINUS.B)
                        ;
                        ;       FMUL/K10
                        ;         [PREVICUSLY VERIFIED]
                        ;
                        ;       FALU/K11
                        ;         FSPAD[TEMP'S], FALU(<59>.SHIFT.OUT)
                        ;
                        ;*****************************************************************
022032  000004          ST60:   SCOPE
                                ;
022034  170127  040040          LDFPS   #040040                 ;INTR-DISAB/F-MODE/TRUNC
022040  012700  002646          MOV     #MFAC0,R0               ;SETUP PTR FOR DIVF SF
022044  012701  000001          MOV     #1,R1                   ;DATA SET NUMBER
022050  012704  000011          MOV     #11,R4                  ;11(8) TABLE ENTRIES
022054  012705  022162          MOV     #40$,R5                 ;DATA TABLE PTR
```

```
                                                ;
                        ;*DATA LOOP ENTERS HERE*
022060  005237  002640  50$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
022064  012537  002656          MOV     (R5)+,MFAC1+0   ;WORD-A, DIVIDEND [ACC]
022070  012537  002660          MOV     (R5)+,MFAC1+2   ;WORD-B
022074  012537  002646          MOV     (R5)+,MFAC0+0   ;WORD-A, DIVISOR [MEMORY]
022100  012537  002650          MOV     (R5)+,MFAC0+2   ;WORD-B
022104  012537  002666          MOV     (R5)+,MFAC2+0   ;WORD-A, QUOTIENT
022110  012537  002670          MOV     (R5)+,MFAC2+2   ;WORD-B
022114  172537  002656          LDF     MFAC1,AC1       ;GET AC[DF]
                                                ;
022120  104406                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                ;
022122  174100          63$:    STF     AC1,AC0         ;RESET AC[DF]
022124  174410                  DIVF    (R0),AC0        ;DIVF EXEC, INBUF/AR.SHIFT, FETOPND SUBR
022126  105737  002645          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
022132  001373                  BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
                                                ;
022134  174037  002676          STF     AC0,MFAC3       ;GET RESULT TO MEMORY
                                                ;
022140  104426  002666  002676  CMP32M  MFAC2,MFAC3     ;(EXP'D) = (RCV'D) ??
022146  001401                  BEQ     59$             ;BR IF AGREE
022150  104104                  ERROR   104             ;ELSE DIVF-EXEC, INBUF-SHIFT ERR
                                ;"DIVIDE INBUF/AR-SHIFT FSPAD-SELECT ERR"
                                ;       DATA-SET        = DATA SET NUMBER, 1 -> 11
                                ;       AC6[SF]         = AC[SF] DIVISOR, FROM MFAC0
                                ;       AC0[DF]         = AC[DF] DIVIDEND, FROM AC0/AC1
                                ;       EXPD RESULT     = EXPECTED AC[DF]/AC[SF] QUOTIENT
                                ;       RCVD RESULT     = RECEIVED QUOTIENT
                                                ;
022152  005201          59$:    INC     R1              ;DATA SET NUMBER
022154  077437                  SOB     R4,50$          ;LOOP ON TABLE
022156  000137  022336          JMP     DIVF01          ;EXIT THE HARD WAY
                                ;
                                ;////////////////////////////////////////////////////////////////////////
                                ;
                                ;       DATA TABLE FOR ABOVE TEST:
                                ;
022162                  40$:    ;
                                ;
022162  040000  000000  D01$:   S0!200*X!000,000000     ;AC[DF], DIVIDEND (AC0)
022166  040000  000000          S0!200*X!000,000000     ;AC[SF], DIVISOR (MFAC0)
022172  040200  000000          S0!201*X!000,000000     ;AC[DF], QUOTIENT
                                ;
022176  152525  052525  D02$:   S1!252*X!125,052525     ;AC[DF], DIVIDEND (AC0)
022202  140200  000000          S1!201*X!000,000000     ;AC[SF], DIVISOR (MFAC0)
022206  052525  052525          S0!252*X!125,052525     ;AC[DF], QUOTIENT
                                ;
022212  025252  125252  D03$:   S0!125*X!052,125252     ;AC[DF], DIVIDEND (AC0)
022216  120200  000000          S1!101*X!000,000000     ;AC[SF], DIVISOR (MFAC0)
022222  145252  125252          S1!225*X!052,125252     ;AC[DF], QUOTIENT
                                ;
022226  177777  177777  D04$:   S1!377*X!177,177777     ;AC[DF], DIVIDEND (AC0)
02223c  077777  177777          S0!377*X!177,177777     ;AC[SF], DIVISOR (MFAC0)
022236  140200  000000          S1!201*X!000,000000     ;AC[DF], QUOTIENT
```

```
 7030
 7031  022242  040200  000000    D05$:   S0!201*X!000,000000     ;AC[DF], DIVIDEND (AC0)
 7032  022246  040500  000000            S0!202*X!100,000000     ;AC[SF], DIVISOR (MFAC0)
 7033  022252  037652  125252            S0!177*X!052,125252     ;AC[DF], QUOTIENT
 7034
 7035  022256  044525  052525    D06$:   S0!222*X!125,052525     ;AC[DF], DIVIDEND (AC0)
 7036  022262  164777  177777            S1!323*X!177,177777     ;AC[SF], DIVISOR (MFAC0)
 7037  022266  117725  052525            S1!077*X!125,052525     ;AC[DF], QUOTIENT
 7038
 7039  022272  140252  125252    D07$:   S1!201*X!052,125252     ;AC[DF], DIVIDEND (AC0)
 7040  022276  040377  177777            S0!201*X!177,177777     ;AC[SF], DIVISOR (MFAC0)
 7041  022302  140052  125252            S1!200*X!052,125252     ;AC[DF], QUOTIENT
 7042
 7043  022306  125377  177777    D10$:   S1!125*X!177,177777     ;AC[DF], DIVIDEND (AC0)
 7044  022312  125325  052525            S1!125*X!125,052525     ;AC[SF], DIVISOR (MFAC0)
 7045  022316  040231  114631            S0!201*X!031,114631     ;AC[DF], QUOTIENT
 7046
 7047  022322  025377  177777    D11$:   S0!125*X!177,177777     ;AC[DF], DIVIDEND (AC0)
 7048  022326  052452  052525            S0!252*X!052,052525     ;AC[SF], DIVISOR (MFAC0)
 7049  022332  013100  060057            S0!054*X!100,060057     ;AC[DF], QUOTIENT
 7050
 7051  022336            DIVF01:
 7052  022336  000400            BR      TST61           ;;      ;EXIT
 7053
 7054
 7055
```

```
;;*****************************************************************
;*TEST 61       "LDC.I.F" EXEC, FPINMUX/DOUT, SHIFT/NORMALIZE
;
;       THIS TEST PROCEEDS THRU ALL THE NON-TRIVIAL PATHS OF THE
;       FP11-E "LDC.I.F" FLOWS.  DATA HAS BEEN SELECTED (SEE BELOW)
;       THAT GENERATES THE DESIRED RESPONSE FROM THE FP11-E SEQUENCING
;       HARDWARE.
;
;       THE MAIN INTENT OF THIS TEST IS TO EXERCISE THE "FPINMUX/DOUT" PORT
;       ON FNUA/K8.
;
;       ----------
;       REGISTER/LOCATION USE:
;
;       AC0     -AC[DF] OUTPUT F-MODE RESULT
;
;       MFAC0+  -EXPECTED AC0 OUTPUT
;       MFAC1+  -AC[DF] MEMORY OUTPUT OF AC0
;
;       R1      -INTEGER 16. BIT OPERAND, SOURCE
;       R4      -TABLE CNTR
;       R5      -TABLE PTR
;
;       ----------
;       MODULE/ERROR INFO:
;
;       FNUA/K8
;         CROM/LATCHES, JREG/BUA, FPINMUX(DOUT<15:00>), F.BUS.A-DRIVER/ENABLE,
;         FP.EMIT.E, INBUF.A/B.DATA
;
;       FEXP/K9
;         CROM/LATCHES, BUT<2:0>-LOGIC
;
;       FMUL/K10
;         [PREVIOUSLY VERIFIED]
;
;       FALU/K11
;         [PREVIOUSLY VERIFIED]
;
;;*****************************************************************
022340  000004          TST61:  SCOPE
                                                        ;
022342  170127  040040          LDFPS   #040040         ;INTR-DISAB/F-MODE/I-MODE/TRUNC
022346  012704  000011          MOV     #11,R4          ;11(8) TABLE ENTRIES
022352  012705  022432          MOV     #40$,R5         ;DATA TABLE PTR
                                                        ;
                                ;*DATA LOOP ENTERS HERE*
022356  005237  002640  10$:    INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
022362  012501                  MOV     (R5)+,R1        ;GET INTEGER OPERAND
022364  012537  002646          MOV     (R5)+,MFAC0+0   ;WORD-A, EXP'D RESULT
022370  012537  002650          MOV     (R5)+,MFAC0+2   ;WORD-B
                                                        ;
022374  104406                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
                                ;--------ERROR-LOOP-ENTERS-HERE--------
                                                        ;
022376  170400          63$:    CLRF    AC0             ;RESET AC[DF]
```

```
7112  022400  177001                        LDCIF   R1,AC0                  ;BM(R1) -> FPINMUX/DOUT -> INBUF -> AC0
7113  022402  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
7114  022406  001373                        BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
7115                                                                          ;
7116  022410  174037  002656                STF     AC0,MFAC1               ;GET RESULT TO MEMORY
7117                                                                          ;
7118  022414  104426  002646  002656        CMP32M  MFAC0,MFAC1             ;(EXP'D) = (RCV'D) ??
7119  022422  001401                        BEQ     19$                     ;BR IF AGREE
7120  022424  104105                        ERROR   105                     ;ELSE ERROR
7121                                        ;"LDCP(I->F) FPINMUX/DOUT CONVERT ERR"
7122                                        ;       INTEGER = 16. BIT INTEGER INPUT, FROM R1
7123                                        ;       EXPD AC0 = EXPECTED F-MODE/AC0 OUTPUT
7124                                        ;       RCVD AC0 = RECEIVED AC0 OUTPUT FROM HFP
7125                                                                          ;
7126  022426  077425                19$:    SOB     R4,10$                  ;LOOP ON TABLE
7127  022430  000433                        BR      TST62           ;;      ;NEXT TEST WHEN DONE
7128
7129                                        ;////////////////////////,,,,////////////////////////////////////, /
7130                                        ;
7131                                        ;       DATA TABLE FOR ABOVE TEST:
7132                                        ;
7133                                        ;               INTEGER   -----F-MODE-EXP'D-----
7134
7135  022432  000000  000000  000000  40$:   .WORD   000000,      000000+000, 000000
7136
7137  022440  077777  043777  177000        .WORD   077777,      043600+177, 177000
7138
7139  022446  052525  043652  125000        .WORD   052525,      043600+052, 125000
7140
7141  022454  025252  043452  124000        .WORD   025252,      043400+052, 124000
7142
7143  022462  065252  043725  052000        .WORD   065252,      043600+125, 052000
7144
7145  022470  177777  140200  000000        .WORD   177777,      140200+000, 000000
7146
7147  022476  100000  144000  000000        .WORD   100000,      144000+000, 000000
7148
7149  022504  125252  143652  126000        .WORD   125252,      143600+052, 126000
7150
7151  022512  152525  143452  126000        .WORD   152525,      143400+052, 126000
7152
7153
7154
```

```
                        ;;*****************************************************************
                        ;*TEST 62       MULNET, BASIC DATAPATH
                        ;
                        ;       THIS SERIES OF TESTS IS THE FIRST USE OF THE MULNET REGISTERS
                        ;       AND ROM MULTIPLIER OF THE FP11-E.
                        ;
                        ;       THIS TEST IN PARTICULAR CONSISTS OF TWO SECTIONS:
                        ;
                        ;         PART 1 - MULTIPLIES, USING 'MPP', MIER(0)*MAND(0)=PROD(0;
                        ;               CHECKS THAT ALL REGISTERS/DATAPATHS CAN BE LOADED/READ,
                        ;               AND THAT THERE ARE NO FLOATING DATA LINES.
                        ;
                        ;         PART 2 -  RIPPLES "1010"/"0101" DATA PATTERNS THRU 4-BIT SECTIONS
                        ;               TO CHECK FOR DATA LINE STUCK-LOW/SHORTS/FLOATING CONDITIONS,
                        ;               AND THE REGISTER LOAD FUNCTIONS (MIER,MAND,SUM,CARRY)
                        ;               ON THE MULNET BOARD.
                        ;
                        ;       ----------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -MPP INPUT DATA PATTERN
                        ;       MFAC1+  -EXP'D MULNET OUTPUT
                        ;       MFAC2+  -RCV'D MULNET(SUM) OUTPUT
                        ;       MFAC3+  -RCV'D MULNET(SUM+CARRY) OUTPUT
                        ;
                        ;       AC0     -MPP (0,0,0,0) INPUT
                        ;       AC1     -MPP (SUM) OUTPUT
                        ;       AC2     -MPP (SUM+CARRY) OUTPUT
                        ;
                        ;       R5      -DATA TABLE PTR
                        ;
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         MNETSUM-ENABLE-LOGIC, 'MPP'-EXEC, CROM/LATCHES
                        ;
                        ;       FEXP/K9
                        ;         MNETREG-CLK, MNET-ALU-CONTROL, MIER/MAND-FUNCTION-CONTROL,
                        ;         MIER/MAND-CLOCKS, 'MPP'-EXEC, CROM/LATCHES
                        ;
                        ;       FMUL/K10
                        ;         MIER-REG/MUX-(BYTE4), MAND-REG-(LOW28), MULXX-ROMS,
                        ;         CNTR-ROMS, SUM-REG, CARRY-REG, MNET-ALU
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;;*****************************************************************
022520  000004          TST62:  SCOPE
                        ;
                        ;------------PART #1: MIER(0)*MAND(0)=PRODUCT(0)------------
                        ;       ;LOOKING FOR STUCK-H/DATAPATH, LACK OF CONTROL OVER REGISTER LOADING
                        ;
022522  170127  040200          LDFPS   #040200                 ;INTR-DISABLE/D-MODE
```

```
7211
7212  022526  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
7213                                        :--------ERROR-LOOP-ENTERS-HERE--------
7214                                        ;
7215  022530  170400                        CLRD    AC0                     ;SET MIER, MAND TO ZEROS
7216  022532  172537  003004                LDD     FPALTP,AC1              ;PRESET (SUM) OUTPUT TO ALT 0/1
7217  022536  172637  003024                LDD     FPALTN,AC2              ;PRESET (S+C) OUTPUT TO ALT 0/1
7218                                        ;
7219  022542  170005                63$:    MPP                             ;RUN THE DATA THRU THE MULNET
7220  022544  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
7221  022550  001374                        BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
7222                                        ;
7223  022552  174137  002656                STD     AC1,MFAC1               ;GET MULNET(SUM)
7224  022556  174237  002666                STD     AC2,MFAC2               ;GET MULNET(SUM+CARRY)
7225                                        ;
7226  022562  104423  002656                FIXFRA  ,MFAC1+0                ;REMOVE SIGN/EXP, INSERT BIT(59:58)
7227  022566  104423  002666                FIXFRA  ,MFAC2+0                ;
7228                                        ;
7229  022572  104425  003060  002656        CMP64M  ,FPZERO,MFAC1           ;IS MULNET(SUM) = (0,u,0,0)?
7230  022600  001402                        BEQ     11$                     ;BR IF OK
7231  022602  104036                        ERROR   36                      ;NO - A "1" CAME THRU SOMEWHERE
7232                                        ;"MULNET - DATA STUCK/H OR REGISTER LOAD ERROR - 0*0=0"
7233                                        ;       RCVD AC1 = MULNET(SUM) FROM HFP AC1 (*ERR*)
7234                                        ;       RCVD AC2 = MULNET(SUM+CARRY) FROM HFP AC2
7235  022604  000405                        BR      20$                     ;NEXT
7236                                        ;
7237  022606  104425  003060  002666  11$:  CMP64M  ,FPZERO,MFAC2           ;IS MULNET(SUM+CARRY) = (0,0,0,0)?
7238  022614  001401                        BEQ     20$                     ;BR IF OK
7239  022616  104036                        ERROR   36                      ;NO - "1" CAME THRU SOMEWHERE
7240                                        ;"MULNET - DATA STUCK/H OR REGISTER LOAD ERROR - 0*0=0"
7241                                        ;       RCVD AC1 = MULNET(SUM) FROM HFP AC1
7242                                        ;       RCVD AC2 = MULNET(SUM+CARRY) FROM HFP AC2 (*ERR*)
7243                                        ;
7244
7245                                :-------PART #2: RIPPLE ALT 0/1 PATTERNS THRU MIER/MAND/PRODUCT-------
7246  022620  012705  022732        20$:    MOV     #40$,R5                 ;SETUP PTR TO DATA TABLE
7247                                        ;
7248                                        ;*DATA LOOP ENTERS HERE*
7249  022624  005237  002640        21$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
7250  022630  012700  000010                MOV     #8.,R0                  ;8 WORDS FOR INITIAL/EXPECTED
7251  022634  012704  002646                MOV     #MFAC0+0,R4             ;INTO MFAC0,1
7252  022640  012524                22$:    MOV     (R5)+,(R4)+             ;
7253  022642  077002                        SOB     R0,22$                  ;
7254                                        ;
7255  022644  104406                        ERRPNT                          ;DON: CHANGE DATA IN ERROR LOOP
7256                                        ;--------ERROR-LOOP-ENTERS-HERE--------
7257                                        ;
7258  022646  172437  002646                LDD     MFAC0,AC0               ;INITIAL MIER, MAND
7259  022652  170401                        CLRD    AC1                     ;INITIAL MULNET(SUM) = 0
7260  022654  170402                        CLRD    AC2                     ;INITIAL MULNET(S+C) = 0
7261                                        ;
7262  022656  170005                62$:    MPP                             ;RUN THE DATA THRU THE MULNET
7263  022660  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
7264  022664  001374                        BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
7265                                        ;
7266  022666  174137  002666                STD     AC1,MFAC2               ;GET MULNET(SUM)
```

```
7267  022672  174237  002676                    STD     AC2,MFAC3               ;GET MULNET(SUM+CARRY)
7268                                              ;
7269  022676  104425  002656  002666            CMP64M  ,MFAC1,MFAC2            ;COMPARE (EXP'D-PROD) = RCV'D-SUM)
7270  022704  001401                            BEQ     23$                     ;BR IF OK
7271  022706  104037                            ERROR   37                      ;ERROR IN DATA FROM MULNET(SUM)
7272                                            ;"MULNET - RIPPLE ALT 0/1 ERROR"
7273                                            ;       HFPP AC0 = INITIAL MIER/MAND IN HFP AC0, BEFORE 'MPP'
7274                                            ;       EXPD AC1 = EXPECTED PRODUCT IN HFP AC1/AC2, AFTER 'MPP'
7275                                            ;       RCVD AC1 = RECEIVED PRODUCT MULNET(S) FROM HFP AC1
7276                                            ;       RCVD AC2 = RECEIVED PRODUCT MULNET(S+C) FROM HFP AC2
7277                                              ;
7278  022710  104425  002666  002676  23$:      CMP64M  ,MFAC2,MFAC3            ;COMPARE (RCV'D-SUM) = (RCV'D-S+C)
7279  022716  001401                            BEQ     24$                     ;BR IF OK
7280  022720  104040                            ERROR   40                      ;ERROR IN DATA FROM MULNET(SUM+CARRY)
7281                                            ;"MULNET - RIPPLE ALT 0/1 ERROR - (S)*(S+C)"
7282                                            ;       (SEE ABOVE FOR DESCRIPTION OF TYPEOUTS)
7283                                              ;
7284                                            ;LOOP ON DATA IN TABLE
7285  022722  005715                    24$:    TST     (R5)                    ;(-1) FLAGS END OF TABLE
7286  022724  100337                            BPL     21$                     ;MORE
7287                                              ;
7288  022726  000137  023474                    JMP     EXT002                  ;ON TO NEXT TEST, THE HARD WAY
7289                                              ;
7290                                            ;---DATA TABLE FOR ABOVE TEST FOLLOWS ON NEXT PAGE---
```

```
7291                                                                    ;
7292            ;////////////////////////////////////////////////////////////////////////////////////////////////
7293            ;
7294            ;       *** THIS IS A DESCRIPTION OF THE DATA TABLE FOR PREVIOUS TEST ***
7295            ;
7296            ;////////////////////////////////////////////////////////////////////////////////////////////////
7297            ;
7298            ;       ALTERNATING 1S/0S THRU MIER[1-0], PRODUCT[1-0] 4-BIT SLICES
7299            ;
7300            ;       !-MIER--! !--------------MAND--------------! !----------------------PRODUCT--------------------!
7301            ;       /B1\ /B0\ /B6\ /B5\ /B4\ /B3\ /B2\ /B1\ /B0\ /EXP \ /A1\ /A0\ /B3\ /B2\ /B1\ /B0\ /C3\ /C2\ /C1\
7302            ;       0000,0101 0000,0000,0000,0000,0000,0000,0001 077000,0000,0000,0000,0000,0000,0000,0000,0000,0101
7303            ;       0000,1010 0000,0000,0000,0000,0000,0000,0001 077000,0000,0000,0000,0000,0000,0000,0000,0000,1010
7304            ;       0101,0000 0000,0000,0000,0000,0000,0000,0001 077000,0000,0000,0000,0000,0000,0000,0000,0101,0000
7305            ;       1010,0000 0000,0000,0000,0000,0000,0000,0001 077000,0000,0000,0000,0000,0000,0000,0000,1010,0000
7306            ;
7307            ;
7308            ;////////////////////////////////////////////////////////////////////////////////////////////////
7309            ;
7310            ;       ALTERNATING 1S/0S THRU MAND[6-0], PRODUCT[8-0] 4-BIT SLICES
7311            ;
7312            ;       !-MIER--! !--------------MAND--------------! !----------------------PRODUCT--------------------!
7313            ;       /B1\ /B0\ /B6\ /B5\ /B4\ /B3\ /B2\ /B1\ /B0\ /EXP \ /A1\ /A0\ /B3\ /B2\ /B1\ /B0\ /C3\ /C2\ /C1\
7314            ;       0000,0001 0000,0000,0000,0000,0000,0000,1010 077000,0000,0000,0000,0000,0000,0000,0000,0000,1010
7315            ;       0000,0001 0000,0000,0000,0000,0000,0000,0101 077000,0000,0000,0000,0000,0000,0000,0000,0000,0101
7316            ;       0000,0001 0000,0000,0000,0000,0000,1010,0000 077000,0000,0000,0000,0000,0000,0000,0000,1010,0000
7317            ;       0000,0001 0000,0000,0000,0000,0000,0101,0000 077000,0000,0000,0000,0000,0000,0000,0000,0101,0000
7318            ;       0000,0001 0000,0000,0000,0000,1010,0000,0000 077000,0000,0000,0000,0000,0000,0000,1010,0000,0000
7319            ;       0000,0001 0000,0000,0000,0000,0101,0000,0000 077000,0000,0000,0000,0000,0000,0000,0101,0000,0000
7320            ;       0000,0001 0000,0000,0000,1010,0000,0000,0000 077000,0000,0000,0000,0000,0000,1010,0000,0000,0000
7321            ;       0000,0001 0000,0000,0000,0101,0000,0000,0000 077000,0000,0000,0000,0000,0000,0101,0000,0000,0000
7322            ;       0000,0001 0000,0000,1010,0000,0000,0000,0000 077000,0000,0000,0000,0000,1010,0000,0000,0000,0000
7323            ;       0000,0001 0000,0000,0101,0000,0000,0000,0000 077000,0000,0000,0000,0000,0101,0000,0000,0000,0000
7324            ;       0000,0001 0000,1010,0000,0000,0000,0000,0000 077000,0000,0000,0000,1010,0000,0000,0000,0000,0000
7325            ;       0000,0001 0000,0101,0000,0000,0000,0000,0000 077000,0000,0000,0000,0101,0000,0000,0000,0000,0000
7326            ;       0000,0001 1010,0000,0000,0000,0000,0000,0000 077000,0000,0000,1010,0000,0000,0000,0000,0000,0000
7327            ;       0000,0001 0101,0000,0000,0000,0000,0000,0000 077000,0000,0000,0101,0000,0000,0000,0000,0000,0000
7328            ;       0001,0000 1010,0000,0000,0000,0000,0000,0000 077000,0000,1010,0000,0000,0000,0000,0000,0000,0000
7329            ;       0001,0000 0101,0000,0000,0000,0000,0000,0000 077000,0000,0101,0000,0000,0000,0000,0000,0000,0000
7330            ;       1000,0000 1010,0000,0000,0000,0000,0000,0000 077600,0101,0000,0000,0000,0000,0000,0000,0000,0000
7331            ;       1000,0000 0100,0000,0000,0000,0000,0000,0000 077400,0010,0000,0000,0000,0000,0000,0000,0000,0000
7332            ;
7333            ;
7334            ;////////////////////////////////////////////////////////////////////////////////////////////////
7335                                                                    ;
7336                                                ;////////////////////////////////////////////////////////////
7337                                                ;       ALTERNATING 1S/0S THRU MIER[1-0], PRODUCT[1-0] 4-BIT SLICES
7338  022732                                        40$:
7339  022732  040200  000005  000000                .WORD   040200,^B00000101,^B0000000000000,^B0000000000000001
7340  022740  000001
7341  022742  077000  000000  000120                .WORD   077000!^B00000000,^B0000000000000000,^B0000000001010000,0000
7342  022750  000000
7343  022752  040200  000012  000000                .WORD   040200,^B00001010,^B0000000000000,^B0000000000000001
7344  022760  000001
7345  022762  077000  000000  000240                .WORD   077000!^B00000000,^B0000000000000000,^B0000000010100000,0000
7346  022770  000000
```

```
7347  022772  040200  000120  000000          .WORD   040200,↑B01010000,↑B0000000000000,↑B0000000000000001
7348  023000  000001
7349  023002  077000  000000  002400          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000010100000000,0000
7350  023010  000000
7351  023012  040200  000240  000000          .WORD   040200,↑B10100000,↑B0000000000000,↑B0000000000000001
7352  023020  000001
7353  023022  077000  000000  005000          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000101000000000,0000
7354  023030  000000
7355                                          ;////////////////////////////////////////////////////////////////////////////
7356                                          ;       ALTERNATING 1S/0S THRU MAND[6-0], PROUUCT[8-0] 4-BIT SLICES
7357  023032  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0000000000001010
7358  023040  000012
7359  023042  077000  000000  000240          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000000010100000,0000
7360  023050  000000
7361  023052  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B000000000000010!
7362  023060  000005
7363  023062  077000  000000  000120          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000000001010000,0000
7364  023070  000000
7365  023072  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0000000010100000
7366  023100  000240
7367  023102  077000  000000  005000          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000101000000000,0000
7368  023110  000000
7369  023112  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0000000001010000
7370  023120  000120
7371  023122  077000  000000  002400          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0000010100000000,0000
7372  023130  000000
7373  023132  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0000010100000000
7374  023140  005000
7375  023142  077000  000000  120000          .WORD   077000!↑B00000000,↑B0000000000000000,↑B1010000000000000,0000
7376  023150  000000
7377  023152  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0000010100000000
7378  023160  002400
7379  023162  077000  00000C  050000          .WORD   077000!↑B00000000,↑B0000000000000000,↑B0101000000000000,0000
7380  023170  000000
7381  023172  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B1010000000000000
7382  023200  120000
7383  023202  077000  000012  000000          .WORD   077000!↑B00000000,↑B0000000000001010,↑B0000000000000000,0000
7384  023210  000000
7385  023212  040200  000001  000000          .WORD   040200,↑B00000001,↑B0000000000000,↑B0101000000000000
7386  023220  050000
7387  023222  077000  000005  000000          .WORD   077000!↑B00000000,↑B0000000000000101,↑B0000000000000000,0000
7388  023230  000000
7389  023232  040200  000001  000012          .WORD   040200,↑B00000001,↑B0000000001010,↑B0000000000000000
7390  023240  000000
7391  023242  077000  000240  000000          .WORD   077000!↑B00000000,↑B0000000010100000,↑B0000000000000000,0000
7392  023250  000000
7393  023252  040200  000001  000005          .WORD   040200,↑B00000001,↑B0000000000101,↑B0000000000000000
7394  023260  000000
7395  023262  077000  000120  000000          .WORD   077000!↑B00000000,↑B0000000001010000,↑B000000000000000C,0000
7396  023270  000000
7397  023272  040200  000001  000240          .WORD   040200,↑B00000001,↑B0000010100000,↑B0000000000000000
7398  023300  000000
7399  023302  077000  005000  000000          .WORD   077000!↑B00000000,↑B0000101000000000,↑B0000000000000000,0000
7400  023310  000000
7401  023312  040200  000001  000120          .WORD   040200,↑B00000001,↑B0000001010000,↑B0000000000000000
7402  023320  000000
```

```
7403  023322  077000  002400  000000          .WORD   077000!↑B000000000,↑B0000010100000000,↑B0000000000000000,0000
7404  023330  000000
7405  023332  040200  000001  005000          .WORD   040200,↑B000000001,↑B1010000000000,↑B0000000000000000
7406  023340  000000
7407  023342  077000  120000  000000          .WORD   077000!↑B000000000,↑B1010000000000000,↑B0000000000000000,0000
7408  023350  000000
7409  023352  040200  000001  002400          .WORD   040200,↑B000000001,↑B0101000000000,↑B0000000000000000
7410  023360  000000
7411  023362  077000  050000  000000          .WORD   077000!↑B000000000,↑B0101000000000000,↑B0000000000000000,0000
7412  023370  000000
7413  023372  040200  000020  005000          .WORD   040200,↑B000010000,↑B1010000000000,↑B0000000000000000
7414  023400  000000
7415  023402  077012  000000  000000          .WORD   077000!↑B000001010,↑B0000000000000000,↑B0000000000000000,0000
7416  023410  000000
7417  023412  040200  000020  002400          .WORD   040200,↑B000010000,↑B0101000000000,↑B0000000000000000
7418  023420  000000
7419  023422  077005  000000  000000          .WORD   077000!↑B000000101,↑B0000000000000000,↑B0000000000000000,0000
7420  023430  000000
7421  023432  040200  000200  005000          .WORD   040200,↑B100000000,↑B1010000000000,↑B0000000000000000
7422  023440  000000
7423  023442  077720  000000  000000          .WORD   077600!↑B001010000,↑B0000000000000000,↑B0000000000000000,0000
7424  023450  000000
7425  023452  040200  000200  002000          .WORD   040200,↑B100000000,↑B0101000000000,↑B0000000000000000
7426  023460  000000
7427  023462  077440  000000  000000          .WORD   077400!↑B000100000,↑B0000000000000000,↑B0000000000000000,0000
7428  023470  000000
7429                                  ;////////////////////////////////////////////////////////////////////////////////
7430  023472  177777                          .WORD   -1              ;END OF TEST
7431                                  ;////////////////////////////////////////////////////////////////////////////////
7432                                                  ;
7433  023474                          EXT002:         ;ON TO NEXT TEST
7434
7435
7436
7437                                  ;;*****************************************************************
7438                                  ;*TEST 63        MULNET, MULTIPLY ROM CONTENTS
7439                                  ;
7440                                  ;       THIS TEST VERIFIES THE CONTENTS OF EACH OF THE FOURTEEN
7441                                  ;       (IDENTICAL) MULNET MULTIPLICATION ROMS.  FOR EACH ROM:
7442                                  ;
7443                                  ;               - MIER 4-BIT ADDRESS VARIED OVER RANGE (17)-(00)
7444                                  ;               - MAND 4-BIT ADDRESS VARIED OVER RANGE (17)-(00)
7445                                  ;               - AT EACH OF THE 256. DATA LOCATIONS (8. ADDRESS BITS),
7446                                  ;                  THE DATA VALUE IS CHECKED TO BE:
7447                                  ;
7448                                  ;                    DATA[8-BITS] = MIER[4-BITS] * MAND[4-BITS]
7449                                  ;
7450                                  ;       AN ERROR IN THIS TEST IS PROBABLY MOST DIRECTLY DUE TO A
7451                                  ;       FAULT IN THE MULNET MULTIPLY ROM UNDER TEST.  HOWEVER, THE
7452                                  ;       MIER, MAND, COUNTER ROMS, SUM AND CARRY REGISTERS, AND MULNET
7453                                  ;       ALU ARE ALSO IN THE DATAPATH.  CONTROL SIGNALS ALSO ORIGINATE
7454                                  ;       ON THE FEXP AND FNUA MODULES.
7455                                  ;
7456                                  ;       ----------
7457                                  ;       REGISTER/LOCATION USE:
7458                                  ;
```

```
                        ;       $REG0   -HOLDS SYMBOLIC ROM # UNDER TEST [EG, 00, 13, 16]
                        ;       $REG1   -MIER SECTION UNDER TEST [0,1]
                        ;       $REG2   -MAND SECTION UNDER TEST [0,1,2,3,4,5,6]
                        ;       $REG3   -MIER LEFT ALIGN SHIFT [0,4]
                        ;       $REG4   -MAND LEFT ALIGN SHIFT [0,4,8,12,16,20,24]
                        ;       $REG5   -PRODUCT RITE ALIGN SHIFT [-4,-8,-12,-16,-20,-24,-28]
                        ;       $REG6   -COUNT OF # OF ERRORS/ROM DETECTED
                        ;       $REG7   -COUNT OF # TIMES (S+C)<>(S) DATA
                        ;       $REG10  -INTERNAL COUNTER [=(7000)]
                        ;       $REG11  -INCLUSIVE "OR" OF BAD ROM DATA, LOC (000)-(377)
                        ;       $REG12  -"AND" OF BAD ROM DATA, LOC (000)-(377)
                        ;       $REG13  -FLAG, 1=CHECK/0=PRINT MODES
                        ;
                        ;       MFAC0+  -ASSEMBLED MIER/MAND OPERAND (AC0)
                        ;       MFAC1+  -MULNET SUM OUTPUT (AC1)
                        ;       MFAC2+  -MULNET SUM+CARRY OUTPUT (AC2)
                        ;
                        ;       R0      -MIER DATA, (00)-(17)
                        ;       R1      -MAND DATA, (00)-(17)
                        ;       R2      -(TEMP)
                        ;       R3      -PRODUCT DATA, (000)-(341) [EXPECTED CONTENTS]
                        ;       R4      -FLAG, UB=SUM ERROR, LB=SUM+CARRY ERROR
                        ;       R5      -(TEMP)
                        ;
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         MNETSUM-ENABLE-LOGIC, 'MPP'-EXEC, CROM/LATCHES
                        ;
                        ;       FEXP/K9
                        ;         MNETREG-CLK, MNET-ALU-CONTROL, MIER/MAND-FUNCTION-CONTROL,
                        ;         MIER/MAND-CLOCKS, 'MPP'-EXEC, CROM/LATCHES
                        ;
                        ;       FMUL/K10
                        ;         MIER-REG/MUX-(BYTE4), MAND-REG-(LOW28), MULXX-ROMS,
                        ;         CNTR-ROMS, SUM-REG, CARRY-REG, MNET-ALU
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
023474  000004          TST63:  SCOPE
                        ;
023476  104405  024202          CNDSES  ,40$                    ;SETUP FOR ESCAPE ON ERROR, IF SW6=1
023502  012737  052525  002646  MOV     #ALTP,MFAC0+0           ;SETUP SIGN, EXPN FOR MPP INSTR
023510  170127  040200          LDFPS   #040200                 ;INTR DISABLE, D-MODE
023514  005037  001322          CLR     $REG10                  ;ZAP INTERNAL COUNTER
023520  012737  000007  001342  MOV     #7,$TIMES               ;DO 7 ITERATIONS OF THIS TEST
                        ;
                        ;       ;LOOP ON MIER GROUP [1:0] (IN $REG1)
023526  012737  000001  001304  MOV     #1,$REG1                ;HOLDS MIER GROUP
                        ;
                        ;       ;LOOP ON MAND GROUP [6:0] (IN $REG2)
023534  012737  000006  001306  1$:     MOV     #6,$REG2                ;HOLDS MAND GROUP
```

```
7515
7516  023542  013700  001304          MOV     $REG1,R0        ;ALIGN CNST FOR MIER BITS
7517  023546  006300                  ASL     R0              ;  = 4*MIER-GROUP
7518  023550  006300                  ASL     R0              ;
7519  023552  010037  001310          MOV     R0,$REG3        ;  SAVE HERE
7520                                                          ;
7521                                  ;MAND LOOP TO HERE      ;
7522  023556  013700  001306  2$:     MOV     $REG2,R0        ;ALIGN CNST FOR MAND BITS
7523  023562  006300                  ASL     R0              ;  = 4*MAND-GROUP
7524  023564  006300                  ASL     R0              ;
7525  023566  010037  001312          MOV     R0,$REG4        ;  SAVE HERE
7526                                                          ;
7527  023572  063700  001310          ADD     $REG3,R0        ;PROD ALIGN CNST =
7528  023576  062700  000004          ADD     #4,R0           ;  -[MIERGRP+MANDGRP+4]
7529  023602  005400                  NEG     R0              ;
7530  023604  010037  001314          MOV     R0,$REG5        ;
7531                                                          ;
7532  023610  013700  001304          MOV     $REG1,R0        ;FORM "MULXX" ROM NUMBER
7533  023614  072027  000003          ASH     #3,R0           ;  AS "MIER-GROUP#MAND-GROUP"
7534  023620  053700  001306          BIS     $REG2,R0        ;
7535  023624  010037  001302          MOV     R0,$REG0        ;  SAVE HERE
7536                                                          ;
7537  023630  012737  000001  001330  MOV     #1,$REG13       ;SET FLAG FOR CHECK MODE
7538                                                          ;
7539  023636  005037  001316  30$:    CLR     $REG6           ;CLEAR #ERRORS CTR
7540  023642  005037  001320          CLR     $REG7           ;CLEAR DIFFERENCE CTR
7541  023646  005037  001324          CLR     $REG11          ;CLEAR "CR"
7542  023652  012737  000377  001326  MOV     #377,$REG12     ;SET "AND"
7543  023660  012700  000017          MOV     #17,R0          ;MIER DATA FROM (00)-(17)
7544                                                          ;
7545                                  ;LOOP ON MIER DATA ENTERS HERE
7546  023664  010002          3$:     MOV     R0,R2           ;ALIGN MIER DATA, VIA MIER-GROUP
7547  023666  072237  001310          ASH     $REG3,R2        ;  [LEFT 4/0]
7548  023672  010237  002650          MOV     R2,MFAC0+2      ;  [INTO WORDB<7:0> OF AC0]
7549  023676  012701  000017          MOV     #17,R1          ;MAND DATA FROM (00)-(17)
7550                                                          ;
7551                                  ;*LOOP ON MAND DATA ENTERS HERE*
7552  023702  005237  002640  4$:     INC     DWLOOP          ;BUMP CLOCK IN.A.LOOP COUNT
7553  023706  010103                  MOV     R1,R3           ;ALIGN MAND DATA, VIA MAND-GROUP
7554  023710  005002                  CLR     R2              ;  [ZAP HI 16. BITS]
7555  023712  073237  001312          ASHC    $REG4,R2        ;  [LEFT 24./20./16./12./8./4./0.]
7556  023716  010237  002652          MOV     R2,MFAC0+4      ;  [INTO WORDC<11:00>]
7557  023722  010337  002654          MOV     R3,MFAC0+6      ;  [INTO WORDD<15:00>]
7558                                                          ;
7559  023726  104406                  ERRPN.                  ;DONT CHANGE DATA IN ERROR LOOP
7560                                  ;--------ERROR-LOOP-ENTERS-HERE--------
7561                                                          ;
7562  023730  012705  002646  38$:     MOV     #MFAC0,R5       ;PTR TO DATA AREA
7563  023734  172425                  LDD     (R5)+,AC0       ;LOAD MIER, MAND TO AC0
7564  023736  170005          63$:     MPP                     ;GET MULNET RESULTS
7565  023740  105737  002645          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
7566  023744  001374                  BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
7567                                                          ;
7568  023746  174125                  STD     AC1,(R5)+       ;STORE MULNET(SUM) IN MFAC1
7569  023750  174215                  STD     AC2,(R5)        ;STORE MULNET(SUM+CARRY) IN MFAC2
7570                                                          ;
```

```
7571  023752  104423  002656          FIXFRA  ,MFAC1+0                ;ZAP SIGN, EXP AND
7572  023756  104423  002666          FIXFRA  ,MFAC2+0                ;  FORM FRAC<59:51> IN WORD-A
7573                                                                  ;
7574  023762  104431  001314  002656  ASH64M  ,$REG5,MFAC1            ;ALIGN PRODUCT RESULTS INTO
7575  023770  104431  001314  002666  ASH64M  ,$REG5,MFAC2            ;  WORDC<7:0>
7576                                                                  ;
7577  023776  010003                  MOV     R0,R3                   ;GENERATE
7578  024000  070301                  MUL     R1,R3                   ;  R3=PRODUCT=MIER*MAND
7579                                                                  ;
7580                                  ;*COMPARE (EXPD PRODUCT/R3):(RCVD [SUM] PRODUCT/MFAC1+4)
7581  024002  005004                  CLR     R4                      ;UB=[SUM] BAD, LB=[SUM+CARRY] BAD
7582  024004  120337  002662          CMPB    R3,MFAC1+4              ;
7583  024010  001413                  BEQ     5$                      ;BR IF AGREE
7584  024012  052704  177400          BIS     #UB,R4                  ;SET ERROR FLAG
7585  024016  113705  002662          MOVB    MFAC1+4,R5              ;GET BAD DATA
7586  024022  150537  001324          BISB    R5,$REG11               ;"IOR" OF BAD
7587  024026  105105                  COMB    R5                      ;
7588  024030  140537  001326          BICB    R5,$REG12               ;"AND" OF BAD
7589  024034  005237  001316          INC     $REG6                   ;BUMP ERROR COUNTER
7590                                                                  ;
7591                                  ;*COMPARE (EXPD PRODUCT/R3):(RCVD [SUM+CARRY] PRODUCT/MFAC2+4)
7592  024040  120337  002672  5$:     CMPB    R3,MFAC2+4              ;
7593  024044  001402                  BEQ     6$                      ;BR IF AGREE
7594  024046  052704  000377          BIS     #LB,R4                  ;SET ERROR FLAG
7595                                                                  ;
7596  024052  123737  002662  002672  6$:     CMPB    MFAC1+4,MFAC2+4 ;(S)=(S+C) ?
7597  024060  001402                  BEQ     8$                      ;BR IF SAME
7598  024062  005237  001320          INC     $REG7                   ;BUMP DIFFERNECE COUNTER
7599                                                                  ;
7600  024066  005737  001330  8$:     TST     $REG13                  ;CHECK OF PRINT ?
7601  024072  003015                  BGT     20$                     ;BR IF CHECK MODE
7602  024074  005704                  TST     R4                      ;ERROR OCCURRED ?
7603  024076  001413                  BEQ     20$                     ;BR IF NOT
7604                                                                  ;
7605  024100  100004                  BPL     7$                      ;BR IF NOT SUM ERROR
7606  024102  005002                  CLR     R2                      ;
7607  024104  153702  002662          BISB    MFAC1+4,R2              ;GET RCVD SUM IN LOB R2
7608                                                                  ;
7609  024110  104016                  ERROR   16                      ;MULNET MULTIPLY ROM ERROR, SUM
7610                                  ;"MULNET MULTIPLY ROM CONTENTS ERROR"
7611                                  ;       -MIER- = HFP MIER 4-BIT DATA SLICE, IN R0<03:00>
7612                                  ;       -MAND- = HFP MAND 4-BIT DATA SLICE, IN R1<03:00>
7613                                  ;       E-DATA = EXPD MULTIPLY ROM OUTPUT, IN R3<07:00>
7614                                  ;       R-DATA = RCVD MULTIPLY ROM OUTPUT, IN R2<07:00>
7615                                                                  ;
7616  024112  105704  7$:     TSTB    R4                              ;
7617  024114  001404                  BEQ     20$                     ;BR IF NO SUM+CARRY ERROR
7618  024116  005002                  CLR     R2                      ;
7619  024120  153702  002672          BISB    MFAC2+4,R2              ;GET RCVD SUM+CARRY IN LOB R2
7620                                                                  ;
7621  024124  104016                  ERROR   16                      ;MULNET MULITPLY ROM ERROR, SUM+CARRY
7622                                  ;"MULNET MULTIPLY ROM CONTENTS ERROR"
7623                                  ;       -MIER- = HFP MIER 4-BIT DATA SLICE, IN R0<03:00>
7624                                  ;       -MAND- = HFP MAND 4-BIT DATA SLICE, IN R1<03:00>
7625                                  ;       E-DATA = EXPD MULTIPLY ROM OUTPUT, IN R3<07:00>
7626                                  ;       R-DATA = RCVD MULTIPLY ROM OUTPUT, IN R2<07:00>
```

```
7627
7628                                        ;NO ERRORS DETECTED
7629  024126  005237  001322      20$:     INC     $REG10                  ;BUMP INTERNAL COUNTER
7630  024132  005301                       DEC     R1                      ;LOOP ON MAND DATA (17)-(00)
7631  024134  002262                       BGE     4$                      ;NEXT LOOP
7632                                                                        ;
7633  024136  005300                       DEC     R0                      ;LOOP ON MIER DATA (17)-(00)
7634  024140  002251                       BGE     3$                      ;NEXT LOOP
7635                                                                        ;
7636                                        ;DONE ALL LOCATIONS OF THIS ROM ?
7637                                        ;ANY ERRORS ENCOUNTERED ?
7638  024142  013705  001316               MOV     $REG6,R5                ;ANY ERRORS ?
7639  024146  053705  001320               BIS     $REG7,R5                ;
7640  024152  001413                       BEQ     40$                     ;BR IF NONE
7641                                                                        ;
7642  024154  005337  001330               DEC     $REG13                  ;ON TO NEXT MODE
7643  024160  100410                       BMI     40$                     ;-1=DONE PRINT, EXIT TO NEXT
7644                                                                        ;0=DONE CHECK, PRINT ERRORS
7645  024162  012737  024172  001222       MOV     #39$,$LPERR             ;EXIT TO AFTER ERROR CALL
7646  024170  104020                       ERROR   20                      ;MULTIPLY ROM ERROR, GENERAL
7647                                        ;"MULNET MULTIPLY ROM ERROR"
7648                                        ;       ROM### = MULTIPLY ROM NUMBER, (00) -> (16)
7649                                        ;       TOTERR = TOTAL # OF ERRORS DETECTED IN THIS ROM
7650                                        ;       S<>S+C = COUNT OF NUMBER OF TIMES (SUM) NOT= (SUM+CARRY)
7651                                        ;       BD-IOR = "OR" OF BAD DATA FOR THIS ROM, A "0"=STUCK-L
7652                                        ;       BD-AND = "AND" OF BAD DATA FOR THIS ROM, A "1"=STUCK-H
7653  024172  012737  023730  001222  39$:  MOV     #38$,$LPERR             ;MAKE A TIGHT ERROR LOOP
7654  024200  000616                       BR      30$                     ;NOW GO TO SPECIFIC
7655                                                                        ;
7656                                        ;ENTER HERE TO GO TO NEXT ROM
7657  024202  005337  001306      40$:     DEC     $REG2                   ;LOOP ON MAND SECTION,  (6)-(0)
7658  024206  002402                       BLT     41$                     ;EXIT
7659  024210  000137  023556               JMP     2$                      ;NEXT LOOP
7660                                                                        ;
7661  024214  005337  001304      41$:     DEC     $REG1                   ;LOOP ON MIER SECTION (1)-(0)
7662  024220  002402                       BLT     TST64           ;;      ;ON TO NEXT TEST WHEN DONE ALL MAND/MIER SECTIONS
7663  024222  000137  023534               JMP     1$                      ;NEXT LOOP
7664                                                                        ;
7665                                        ;DONE WITH ALL LOCATIONS OF ALL 14./(16) MULNET ROMS
7666
7667
7668
7669                                ;;**********************************************************************
7670                                ;*TEST 64       MULNET, SUM/CARRY REGISTERS, COUNTER ROM CONTENTS
7671                                ;
7672                                ;       THIS TEST VERIFIES THE CONTENTS OF EACH OF THE FOURTEEN
7673                                ;       (IDENTICAL) MULNET COUNTER ROMS.  FOR EACH ROM:
7674                                ;
7675                                ;               - 4 GROUPS OF 2 [8-ADDRESS LINES] ARE VARIED
7676                                ;                 TO ALL THE POSSIBLE ADDRESS CONBINATIONS THAT
7677                                ;                 CAN BE GENERATED AT THE "MUL-XX" ROM OUTPUTS
7678                                ;               - AT EACH REFERENCABLE DATA/ROM LOCATION (256. MAX)
7679                                ;                 THE SUM AND CARRY ROM OUTPLTS ARE CHECKED TO BE
7680                                ;                 THE CORRECT VALUE.
7681                                ;
```

```
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
;       AN ERROR IN THIS TEST IS PROBABLY MOST DIRECTLY DUE TO A
;       FAULT IN THE MULNET COUNTER ROM UNDER TEST.  HOWEVER, THE
;       MULNET ALU AND ITS CARRY LOGIC ARE ALSO IN THE DATAPATH,
;       ALONG WITH THE SUM AND CARRY REGISTERS AND THEIR CONTROL
;       LOGIC (ON FEXP).  THE MIER, MAND, MUL-XX ROMS WERE
;       SUBSTANTIALLY VERIFIED IN THE PREVIOUS TEST.
;
;       ----------
;       REGISTER/LOCATION USE:
;
;       $REG6   -CNTR ROM LOCATION COUNTER, (377)-(000)
;       $REG7   -ROM ADDRESS MASK, (000)-8BIT / (210)-6BIT
;       $REG10  -INTERNAL COUNTER [=(5044)]
;       $REG11  -PRODUCT ALIGN-SHIFT CONSTANT [-8,-10,...,-34]
;       $REG12  -MAND ALIGN-SHIFT CONSTANT [-4,0,4,8,12,16,20]
;       $REG13  -CLASS CODE = ROM###<00>
;       $REG14  -[A] 4-BIT MIER DATA
;       $REG15  -[B] 4-BIT MAND DATA
;       $REG16  -[C] 4-BIT MAND DATA
;       $REG17  -[D] 4-BIT MAND DATA
;
;       MFAC0+  -ASSEMBLED MIER/MAND OPERAND (AC0)
;       MFAC1+  -MULNET(SUM), FROM(AC1)
;       MFAC2+  -MULNET(CARRY), FROM(AC2-AC1)
;
;       R0      -(TEMP)
;       R1      -ACTUAL CNTR ROM ADDRESS, (000)-(377)
;       R2      -EXPECTED [CCSS] DATA FROM CNTR ROM, (TEMP)
;       R3      -RECEIVED [CCSS] DATA FROM CNTR ROM, (TEMP)
;       R4      -CNTR ROM ###, (00)-(15)
;       R5      -(TEMP)
;       ----------
;
;       PROGRAM   ACTUAL-CNTR   CONNECTIONS TO
;       ROM-###   ROM BIT **'S  "MUL-XX" ROMS
;       -------   -----------   ------------------------------
;          00       <01:00>     MUL-10,01<1:0>; MUL-...00<5:4>
;          01       <03:02>     MUL-10,01<3:2>; MUL-...00<7:6>
;          02       <05:04>     MUL-11,02<1:0>; MUL-10,01<5:4>
;          03       <07:06>     MUL-11,02<3:2>; MUL-10,01<7:6>
;          04       <09:08>     MUL-12,03<1:0>; MUL-11,02<5:4>
;          05       <11:10>     MUL-12,03<3:2>; MUL-11,02<7:6>
;          06       <13:12>     MUL-13,04<1:0>; MUL-12,03<5:4>
;          07       <15:14>     MUL-13,04<3:2>; MUL-12,03<7:6>
;          10       <17:16>     MUL-14,05<1:0>; MUL-13,04<5:4>
;          11       <19:18>     MUL-14,05<3:2>; MUL-13,04<7:6>
;          12       <21:20>     MUL-15,06<1:0>; MUL-14,05<5:4>
;          13       <23:22>     MUL-15,06<3:2>; MUL-14,05<7:6>
;          14       <25:24>     MUL-16...<1:0>; MUL-15,06<5:4>
;          15       <27:26>     MUL-16...<3:2>; MUL-15,06<7:6>
;
;       ----------
;       MODULE/ERROR INFO:
;
;       FNUA/K8
```

```
                                ;         [PREVIOUSLY VERIFIED]
                                ;
                                ;         FEXP/K9
                                ;           MNET-ALU-CONTROL
                                ;
                                ;         FMUL/K10
                                ;           CNTR-ROMS, SUM-REG, CARRY-REG, MNET-ALU
                                ;
                                ;         FALU/K11
                                ;           [PREVIOUSLY VERIFIED]
                                ;
                                ;*****************************************************************
024226  000004                  TST64:  SCOPE
                                ;
024230  104405  024736                  CNDSES  ,20$                    ;SETUP FOR ESCAPE ON ERROR, IF SW6=1
024234  012737  052525  002646          MOV     #ALTP,MFAC0+0           ;SETUP SIGN, EXPN FOR MPP INSTR
024242  170127  040200                  LDFPS   #040200                 ;INTR DISABLE,  D-MODE
024246  012737  000007  001342          MOV     #7,$TIMES               ;DO 7 ITERATIONS OF THIS TEST
024254  005037  001322                  CLR     $REG10                  ;ZAP INTERNAL COUNTER
                                ;
                                ;LOOP ON THE 14./(16) COUNTER ROMS  (#00-15)
024260  012704  000015                  MOV     #15,R4                  ;
                                ;
                                ;*LOOP ON NEXT ROM UNDER TEST ENTERS HERE
024264  010400                  1$:     MOV     R4,R0                   ;ALIGN SHIFT CNST FOR MAND BITS
024266  006200                          ASR     R0                      ;  =[ROM#/2]*4-4
024270  006300                          ASL     R0                      ;
024272  006300                          ASL     R0                      ;
024274  162700  000004                  SUB     #4,R0                   ;
024300  010037  001326                  MOV     R0,$REG12               ;STORE HERE
                                ;
024304  010400                          MOV     R4,R0                   ;GET ROM ###
024306  006300                          ASL     R0                      ;
024310  062700  000010                  ADD     #8.,R0                  ;ALIGN SHIFT CNST FOR PRODUCT BITS
024314  005400                          NEG     R0                      ;  =-[2*ROM###+8.]
024316  010037  001324                  MOV     R0,$REG11               ;STORE HERE
                                ;
024322  010437  001330                  MOV     R4,$REG13               ;CLASS<0>=ROM#<0>
024326  042737  177776  001330          BIC     #^CBIT00,^ _G13         ;STORE HERE
                                ;
                                ;LOOP ON (400) LOCATIONS/ROM, ROMS #(02)-(13)
                                ;LOOP ON (100) LOCATIONS/ROM, ROMS #(00)-(01), (14)-(15)
024334  012737  000377  001316          MOV     #377,$REG6              ;MAX OF 8-BITS
024342  005037  001320                  CLR     $REG7                   ;SETUP FOR FULL RANGE
024346  020427  000002                  CMP     R4,#02                  ;
024352  002403                          BLT     2$                      ;
024354  020427  000013                  CMP     R4,#13                  ;
024360  003403                          BLE     3$                      ;
024362  012737  000210  001320  2$:     MOV     #210,$REG7              ;6.BIT ADDR FOR (00)-(01), (14)-(15)
                                ;
                                ;*LOOP ON NEXT ROM LOCATION UNDER TEST ENTERS HERE
024370  005237  002642          3$:     INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
024374  004737  024752                  JSR     PC,MAPADR               ;SUBR THAT TRANSFORMS COUNT -> ROMADR
024400  103553                          BCS     11$                     ;IF RETURN WITH C-BIT SET, INDICATES
                                ;                                       ;  THAT IT IS PHYSICALLY IMPOSSIBLE TO
                                ;                                       ;  GENERATE THIS ROM ADDRESS.
```

```
7794
7795                                        ;THIS NEXT SEQUENCE OF CODE LOOPS UP THE APPROPRIATE [A], [B],
7796                                        ;[C], AND [D] VALUES TO USE TO GENERATE THE DESIRED ROM
7797                                        ;ADDRESS AT THE COUNTER ROM INPUTS.
7798  024402  004737  025156                JSR     PC,OFFCL1           ;GET RO=OFFSET IN CODEA TABLE
7799  024406  116003  025256                MOVB    CODEA(RO),R3        ;GET [A] FROM TABLE
7800  024412  100546                        BMI     11$                 ;ILLEGAL IF A (-1)
7801  024414  010337  001332                MOV     R3,$REG14           ;STORE AWAY
7802                                        ;
7803  024420  116002  025316                MOVB    CODEB(RO),R2        ;GET [B] FROM TABLE
7804  024424  100541                        BMI     11$                 ;ILLEGAL IF A (-1)
7805  024426  010237  001334                MOV     R2,$REG15           ;STORE AWAY
7806                                        ;
7807  024432  006303                        ASL     R3                  ;
7808  024434  006303                        ASL     R3                  ;
7809  024436  042703  177767                BIC     #^CBIT03,R3         ;=CODEA<1>#000
7810                                        ;
7811  024442  004737  025164                JSR     PC,OFFCL2           ;RO=OFFSET INTO CODEC TABLE
7812  024446  116002  025356                MOVB    CODEC(RO),R2        ;GET [C] FROM TABLE
7813  024452  100526                        BMI     11$                 ;ILLEGAL IF A (-1)
7814  024454  010237  001336                MOV     R2,$REG16           ;STORE AWAY
7815                                        ;
7816  024460  004737  025164                JSR     PC,OFFCL2           ;RO=OFFSET INTO CODED TABLE
7817  024464  116002  025376                MOVB    CODED(RO),R2        ;GET [D] FROM TABLE
7818  024470  100517                        BMI     11$                 ;ILLEGAL IF A (-1)
7819  024472  010237  001340                MOV     R2,$REG17           ;STORE AWAY
7820                                        ;
7821                                        ;PUT MAND=[D]#[B]#[C], ALIGNED, INTO WORD-C,D
7822  024476  013703  001340                MOV     $REG17,R3           ;GET [D]
7823  024502  072327  000004                ASH     #4,R3               ;LEFT-4
7824  024506  053703  001334                BIS     $REG15,R3           ;FORM [D]#[B]
7825  024512  072327  000004                ASH     #4,R3               ;LEFT-4
7826  024516  053703  001336                BIS     $REG16,R3           ;FORM [D]#[B]#[C]
7827  024522  005002                        CLR     R2                  ;ZAP HI BITS
7828  024524  073237  001326                ASHC    $REG12,R2           ;ALIGN MAND IN (R2:R3)
7829  024530  042702  170000                BIC     #170000,R2          ;ZAP UNUSED
7830  024534  010237  002652                MOV     R2,MFAC0+4          ;INSERT MAND
7831  024540  010337  002654                MOV     R3,MFAC0+6          ;  INTO OPERAND
7832                                        ;PUT MIER=[A]#[A] INTO WORD-B
7833  024544  013700  001332                MOV     $REG14,RO           ;GET [A]
7834  024550  072027  000004                ASH     #4,RO               ;LEFT-4
7835  024554  053700  001332                BIS     $REG14,RO           ;FORM [A]#[A]
7836  024560  010037  002650                MOV     RO,MFAC0+2          ;INSERT MIER
7837                                        ;
7838                                        ;CALCULATE EXPECTED [CCSS] IN R2<3:0>
7839  024564  010102                        MOV     R1,R2               ;GET ROM ADDRESS
7840  024566  012705  000002                MOV     #2,R5               ;2 GROUPS OF BITS
7841  024572  005046                4$:     CLR     -(SP)               ;CLEAR SUBTOTAL
7842  024574  012700  000004                MOV     #4,RO               ;4 BITS/GROUP
7843  024600  006202                5$:     ASR     R2                  ;C-BIT=NEXT IN GROUP
7844  024602  005516                        ADC     (SP)                ;ADD TO SUBTOTAL
7845  024604  077003                        SOB     RO,5$               ;LOOP ON 4 BITS/GROUP
7846  024606  077507                        SOB     R5,4$               ;LOOP ON 2 GROUPS
7847  024610  012602                        MOV     (SP)+,R2            ;GET B<3:0> SUM
7848  024612  006302                        ASL     R2                  ;ALIGN
7849  024614  062602                        ADD     (SP)+,R2            ;ADD A<3:0> SUM
```

```
7850
7851  024616  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
7852                                        ;--------ERROR-LOOP-ENTERS-HERE--------
7853                                        ;
7854                                        ;NOW ACTUALLY RUN THIS DATA THRU THE MULNET
7855  024620  012700  002646                MOV     #MFAC0,R0               ;PTR TO DATA AREA
7856  024624  172420                        LDD     (R0)+,AC0               ;LOAD MIER, MAND TO AC0
7857  024626  170005              63$:      MPP                             ;GET MULNET RESULTS
7858  024630  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
7859  024634  001374                        BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
7860                                        ;
7861  024636  174120                        STD     AC1,(R0)+               ;STORE MULNET(SUM) IN MFAC1
7862  024640  174210                        STD     AC2,(R0)                ;STORE MULNET(SUM+CARRY) IN MFAC2
7863                                        ;
7864  024642  104423  002656                FIXFRA  ,MFAC1+0                ;ZAP SIGN, EXP AND FORM
7865  024646  104423  002666                FIXFRA  ,MFAC2+0                ;  FRAC<59:51> IN WORD-A
7866                                        ;
7867  024652  104432  002656  002666        SUB64M  ,MFAC1,MFAC2            ;FORM MULNET(CARRY)=
7868                                        ;                               ;  MULNET(SUM+CARRY)-MULNET(SUM), IN MFAC2
7869                                        ;
7870  024660  104431  001324  002656        ASH64M  ,$REG11,MFAC1           ;SUM IN WORD-C<1:0>
7871  024666  104431  001324  002666        ASH64M  ,$REG11,MFAC2           ;CARRY IN WORD-C<3:2>
7872                                        ;
7873                                        ;COMBINE SUM, CARRY INTO [CCSS]  IN  R3<3:0>
7874  024674  013703  002662                MOV     MFAC1+4,R3              ;GET SUM
7875  024700  042703  177774                BIC     #^C3,R3                 ;CLEAR REST
7876  024704  013700  002672                MOV     MFAC2+4,R0              ;GET CARRY
7877  024710  042700  177763                BIC     #^C14,R0                ;CLEAR REST
7878  024714  050003                        BIS     R0,R3                   ;FORM [CCSS] RECEIVED DATA
7879                                        ;
7880                                        ;*COMPARE (EXPECTED [CCSS]/R2):(RECEIVED [CCSS]/R3)
7881  024716  020203                        CMP     R2,R3                   ;
7882  024720  001401                        BEQ     10$                     ;BR IF AGREE
7883                                        ;
7884  024722  104017                        ERROR   17                      ;ERROR IN CNTR ROM CONTENTS
7885                                        ;"MULNET COUNTER ROM CONTENTS ERROR"
7886                                        ;       ROM### = ROM NUMBER OF ROM IN WHICH ERROR DETECTED
7887                                        ;       ROMADR = ROM ADDRESS OF BAD LOCATION
7888                                        ;       E-DATA = EXPD DATA, IN FORMAT [CCSS] IN R2<03:00>
7889                                        ;       R-DATA = RCVD DATA, IN FORMAT [CCSS] IN R3<03:00>
7890                                        ;
7891                                        ;ENTER HERE IF LOCATION OK
7892  024724  005237  001322        10$:    INC     $REG10                  ;BUMP INTERNAL COUNTER
7893  024730  005337  001316        11$:    DEC     $REG6                   ;LOOP ON COUNT
7894  024734  002215                        BGE     3$                      ;NEXT LOOP
7895                                        ;
7896  024736  005304                20$:    DEC     R4                      ;LOOP ON ROM NUMBER (15)-(00)
7897  024740  002402                        BLT     21$                     ;EXIT IF DONE
7898  024742  000137  024264                JMP     1$                      ;NEXT ROM
7899                                        ;
7900                                        ;DONE WITH ALL LOCATIONS OF ALL 14./(16) COUNTER ROMS
7901  024746  000137  025416        21$:    JMP     EXT001                  ;MUST JUMP TO EXIT
7902                                        ;
7903                                        ;////////////////////////////////////////////////////////////////////////
7904                                        ;
7905                                        ;       THE FOLLOWING SUBROUTINE IS USED TO REMAP THE COUNTER VALUE
```

```
                        ;       TO THE ROM ADDRESS BITS AS THEY ACTUALLY EXIST AT THE INPUTS
                        ;       TO THE COUNTER (COUNTER) ROMS.
                        ;
                        ;       ENTER WITH:     R4=ROM NUMBER, (00)-(13),
                        ;                       $REG6=COUNT VALUE, (000)-(377)
                        ;                       $REG7=ILLEGAL BIT MASK, (000) OR (210)
                        ;       EXIT WITH:      C-BIT= 1/ILLEGAL-ADDR, 0/VALID-ADDR
                        ;                       R1=GENERATED ROM ADDRESS
                        ;       TEMPS:          R0,R5
                        ;
024752  010401          MAPADR: MOV     R4,R1                   ;GET ROM NUMBER
024754  006201                  ASR     R1                      ;SET LSB=0
024756  006301                  ASL     R1                      ;FOR A WORD OFFSET
024760  000171  025040          JMP     @40$(R1)                ;GO TO ROUTINE SPECIFIED BY TABLE
                                                                ;
                        ;TYPE#1, ROM-ADDR=COUNTER-VALUE, ROMS # 00,01,04,05,10,11
024764  013701  001316  10$:    MOV     $REG6,R1                ;EXACT COPY
024770  000415                  BR      30$                     ;AND CONTINUE
                                                                ;
                        ;TYPE#2, REMAP FUNCTION/A/, ROMS # 02,03,06,07,12,13
024772  012705  025056  11$:    MOV     #41$,R5                 ;PTR TO FUNCTION TABLE
024776  000402                  BR      20$                     ;
                                                                ;
                        ;TYPE#3, REMAP FUNCTION/B/, ROMS # 14,15
025000  012705  025116  12$:    MOV     #42$,R5                 ;PTR TO FUNCTION TABLE
                                                                ;
                        ;DO THE REMAP
025004  005001          20$:    CLR     R1                      ;BASE OF ZERO
025006  012500          21$:    MOV     (R5)+,R0                ;GET DEST BIT
025010  001405                  BEQ     30$                     ;IF ZERO, DONE
025012  032537  001316          BIT     (R5)+,$REG6             ;TEST SOURCE BIT
025016  001773                  BEQ     21$                     ;IF ZERO, SKIP IT
025020  050001                  BIS     R0,R1                   ;ELSE MOVE SRC -> DST
025022  000771                  BR      21$                     ;AND CONT
                                                                ;
                        ;NOW CHECK FOR VALID ADDRESSES
025024  000241          30$:    CLC                             ;ASSUME VALID
025026  033701  001320          BIT     $REG7,R1                ;ANY INVALID BITS SET ?
025032  001401                  BEQ     39$                     ;BR IF NOT
025034  000261                  SEC                             ;YES, SIGNAL INVALID
025036  000207          39$:    RTS     PC                      ;AND RETURN
                        ;
                        ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ;  TABLE # 1, SPECIFY REMAP FUNCTION TO USE
                        ;
                                        ADDR    ROM##   FUNCTION/##/
025040  024764          40$:    .WORD   10$     ;00-01   1-DIRECT
025042  024772                  .WORD   11$     ;02-03   2-REMAP/A/
025044  024764                  .WORD   10$     ;04-05   1-DIRECT
025046  024772                  .WORD   11$     ;06-07   2-REMAP/A/
025050  024764                  .WORD   10$     ;10-11   1-DIRECT
025052  024772                  .WORD   11$     ;12-13   2-REMAP/A/
025054  025000                  .WORD   12$     ;14-15   3-REMAP/B/
                        ;
                        ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                        ;       SPECIFY REMAP FUNCTION /A/, FOR ROMS:
```

```
                                        ;
                                        ;               02,03, 06,07, 12,13
                                        ;
                                        ;               DST  SRC
7966  025056  000200  000100            41$:    .WORD   BIT7,BIT6       ;DST7 <- SRC6
7967  025062  000100  000200                    .WORD   BIT6,BIT7       ;DST6 <- SRC7
7968  025066  000040  000020                    .WORD   BIT5,BIT4       ;DST5 <- SRC4
7969  025072  000020  000040                    .WORD   BIT4,BIT5       ;DST4 <- SRC5
7970  025076  000010  000004                    .WORD   BIT3,BIT2       ;DST3 <- SRC2
7971  025102  000004  000010                    .WORD   BIT2,BIT3       ;DST2 <- SRC3
7972  025106  000002  000001                    .WORD   BIT1,BIT0       ;DST1 <- SRC0
7973  025112  000001  000002                    .WORD   BIT0,BIT1       ;DST0 <- SRC1
7974                                    ;
7975                                    ;- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
7976                                    ;       SPECIFY REMAP FUNCTION /B/, FOR ROMS:
7977                                    ;
7978                                    ;               14,15
7979                                    ;
7980                                    ;               DST  SRC
7981  025116  000200  000100            42$:    .WORD   BIT7,BIT6       ;DST7 <- SRC6
7982  025122  000100  000200                    .WORD   BIT6,BIT7       ;DST6 <- SRC7
7983  025126  000040  000040                    .WORD   BIT5,BIT5       ;DST5 <- SRC5
7984  025132  000020  000020                    .WORD   BIT4,BIT4       ;DST4 <- SRC4
7985  025136  000010  000004                    .WORD   BIT3,BIT2       ;DST3 <- SRC2
7986  025142  000004  000010                    .WORD   BIT2,BIT3       ;DST2 <- SRC3
7987  025146  000002  000002                    .WORD   BIT1,BIT1       ;DST1 <- SRC1
7988  025152  000001  000001                    .WORD   BIT0,BIT0       ;DST0 <- SRC0
7989                                    ;
7990                                    ;////////////////////////////////////////////////////////////////////////
7991                                    ;
7992                                    ;       THE FOLLOWING SUBROUTINE IS USED ABOVE TO TRANSFORM THE
7993                                    ;       ROM ADDRESS, WHICH RANGES FROM (000) TO (377)/(077)
7994                                    ;       (DEPENDING UPON THE CNTR ROM UNDER TEST), INTO THE ACTUAL
7995                                    ;       DISPLACEMENT NEEDED TO OFFSET INTO THE CODEX TABLES BELOW
7996                                    ;
7997                                    ;       ENTER WITH:     R1=ROM ADDR
7998                                    ;                       $REG13=CLASS<0>
7999                                    ;       EXIT WITH:      R0=OFFSET CALC
8000                                    ;       TEMPS:          R2,R3,R5
8001                                    ;
8002                                    ;       "OFFSET FOR CODE-A/B" ENTRY POINT
8003  025156  012705  025210            OFFCL1: MOV     #OFFTBA,R5              ;INITIALIZE TABLE PTR
8004  025162  005003                            CLR     R3                      ;ZAP BASE MASK
8005                                            ;"OFFSET FOR CODE-C/D" ENTRY POINT
8006  025164  010300                    OFFCL2: MOV     R3,R0                   ;GET BASE MASK
8007  025166  012502                    9$:     MOV     (R5)+,R2                ;R2=DEST BIT POSITION
8008  025170  001404                            BEQ     10$                     ;IF ZERO, DONE
8009  025172  032501                            BIT     (R5)+,R1                ;TEST SOURCE BIT
8010  025174  001774                            BEQ     9$                      ;IF ZERO, SKIP IT
8011  025176  050200                            BIS     R2,R0                   ;IF -ZERO, SET DEST BIT
8012  025200  000772                            BR      9$                      ;
8013  025202  053700  001330            10$:    BIS     $REG13,R0               ;BIT<0> IS CLASS<0>
8014  025206  000207                            RTS     PC                      ;AND DONE
8015                                    ;
8016                                    ;       CODE-A/B OFFSET GENERATION TABLE
8017  025210  000020  000020            OFFTBA: .WORD   BIT4,BIT4               ;DST4 <- SRC4
```

```
8018  025214  000010  000001          .WORD   BIT3,BIT0                ;DST3 <- SRC0
8019  025220  000004  000100          .WORD   BIT2,BIT6                ;DST2 <- SRC6
8020  025224  000002  000004          .WORD   BIT1,BIT2                ;DST1 <- SRC2
8021  025230  000000                  .WORD   0                        ;DONE
8022                                  ;---------------------------------------
8023                                  ;CODE-C OFFSET GENERATION TABLE
8024  025232  000004  000200          .WORD   BIT2,BIT7                ;DST2 <- SRC7
8025  025236  000002  000010          .WORD   BIT1,BIT3                ;DST1 <- SRC3
8026  025242  000000                  .WORD   0                        ;DONE
8027                                  ;---------------------------------------
8028                                  ;CODE-D OFFSET GENERATION TABLE
8029  025244  000004  000040          .WORD   BIT2,BIT5                ;DST2 <- SRC5
8030  025250  000002  000002          .WORD   BIT1,BIT1                ;DST1 <- SRC1
8031  025254  000000                  .WORD   0                        ;DONE
8032
8033                                  ;
8034                                  ;////////////////////////////////////////////////////////////////////////
8035                                  ;
8036                                  ;       THE FOLLOWING FOUR TABLES [CODEA, CODEB, CODEC, CODED] CONTAIN
8037                                  ;       THE 4-BIT OPERANDS NECESSARY TO INPUT TO THE "MUL-XX" ROMS,
8038                                  ;       SO THAT THE FULL(EST) CYCLE OF ADDRESSES MAY BE INPUT TO
8039                                  ;       THE MULNET COUNTER ROMS, IN A PREDETERMINED ORDER.
8040                                  ;
8041                                  ;       NOTING:
8042                                  ;               [MIER/8]=[A/4]*[A/4]
8043                                  ;               [MAND/28]=[D/4]*[B/4]*[C/4], SHIFTED AND ZEROFILLED
8044                                  ;
8045                                  ;               WHERE A,B,C,D ARE THE 4-BIT VALUES OBTAINED
8046                                  ;               FROM THE TABLES BELOW
8047                                  ;
8048                                  ;       EXAMPLE#1:
8049                                  ;               COUNTER ROM ADDR INPUTS FROM MULTIPLY "MUL-XX"
8050                                  ;               ROMS BITS<5:4>, <1:0> (CLASS '5410')
8051                                  ;               [COUNTER ROM W/ OUTPUT MNETSUM<9:8>]
8052                                  ;
8053                                  ;               CNTR-ADDR<7,3>=MUL11<5:4>,[A=MIER<7:4>]*[C=MAND<07:04>]
8054                                  ;               CNTR-ADDR<6,2>=MUL12<1:0>,[A=MIER<7:4>]*[B=MAND<11:08>]
8055                                  ;               CNTR-ADDR<5,1>=MUL03<1:0>,[A=MIER<3:0>]*[D=MAND<15:12>]
8056                                  ;               CNTR-ADDR<4,0>=MUL02<5:4>,[A=MIER<3:0>]*[B=MAND<11:08>]
8057                                  ;
8058                                  ;       EXAMPLE#2:
8059                                  ;               COUNTER ROM ADDR INPUTS FROM MULTIPLY "MUL-XX"
8060                                  ;               ROMS' BITS<7,3>, <3:2>  (CLASS '7632')
8061                                  ;               [COUNTER ROM W/ OUTPUT MNETSUM<11:10>]
8062                                  ;
8063                                  ;               CNTR-ADDR<7,3>=MUL11<7:6>, [A=MIER<7:4>]*[C=MAND<07:04>]
8064                                  ;               CNTR-ADDR<6,2>=MUL12<3:2>, [A=MIER<7:4>]*[B=MAND<11:08>]
8065                                  ;               CNTR-ADDR<5,1>=MUL03<3:2>, [A=MIER<3:0>]*[D=MAND<15:12>]
8066                                  ;               CNTR-ADDR<4,0>=MUL02<7:6>, [A=MIER<3:0>]*[B=MAND<11:08>]
8067
8068
8069
8070                                  ;////////////////////////////////////////////////////////////////////////
8071
8072                                  ;       CODE-A TABLE
8073                                  ;
```

```
                        ;       OFFSET=CNTR-ADDR<4,0,6,2>*CLASS<0>
                        ;
                        ;       NOTE:   CLASS<0>=0 FOR '5410', CLASS<0>=1 FOR '7632'
                        ;
                        ;                       CODE    CODE      CNTR-ADDR
                        ;       CLASS:          (5410)  (7632)  <4,0>    <6,2>
                        ;                       ------  ------  -----    -----
025256  007     016     CODEA:  .BYTE   07,     16      ;00     00
025260  007     014             .BYTE   07,     14      ;00     01
025262  007     016             .BYTE   07,     16      ;00     10
025264  007     016             .BYTE   07,     16      ;00     11
025266  007     016             .BYTE   07,     16      ;01     00
025270  007     016             .BYTE   07,     16      ;01     01
025272  011     014             .BYTE   11,     14      ;01     10
025274  007     016             .BYTE   07,     16      ;01     11
025276  011     014             .BYTE   11,     14      ;10     00
025300  007     016             .BYTE   07,     16      ;10     01
025302  007     016             .BYTE   07,     16      ;10     10
025304  007     016             .BYTE   07,     16      ;10     11
025306  007     016             .BYTE   07,     16      ;11     00
025310  007     016             .BYTE   07,     16      ;11     01
025312  011     377             .BYTE   11,     -1      ;11     10  (-1=NOT POSSIBLE)
025314  007     377             .BYTE   07,     -1      ;11     11  (-1=NOT POSSIBLE)
                        ;
                        ;////////////////////////////////////////////////////////////////
                        ;
                        ;       CODE-B TABLE
                        ;
                        ;       OFFSET=CNTR-ADDR<4,0,6,2>*CLASS<0>
                        ;
                        ;       NOTE:   CLASS<0>=0 FOR '5410', CLASS<0>=1 FOR '7632'
                        ;
                        ;                       CODE    CODE      CNTR-ADDR
                        ;       CLASS:          (5410)  (7632)  <4,0>    <6,2>
                        ;                       ------  ------  -----    -----
025316  000     000     CODEB:  .BYTE   00,     00      ;00     00
025320  013     003             .BYTE   13,     03      ;00     01
025322  002     004             .BYTE   02,     04      ;00     10
025324  001     002             .BYTE   01,     02      ;00     11
025326  004     010             .BYTE   04,     10      ;01     00
025330  003     006             .BYTE   03,     06      ;01     01
025332  002     006             .BYTE   02,     06      ;01     10
025334  015     011             .BYTE   15,     11      ;01     11
025336  004     014             .BYTE   04,     14      ;10     00
025340  017     015             .BYTE   17,     15      ;10     01
025342  006     014             .BYTE   06,     14      ;10     10
025344  005     012             .BYTE   05,     12      ;10     11
025346  010     017             .BYTE   10,     17      ;11     00
025350  007     016             .BYTE   07,     16      ;11     01
025352  006     377             .BYTE   06,     -1      ;11     10  (-1=NOT POSSIBLE)
025354  011     377             .BYTE   11,     -1      ;11     11  (-1=NOT POSSIBLE)
                        ;
                        ;////////////////////////////////////////////////////////////////
                        ;
```

```
                        ;       CODE-C TABLE
                        ;
                        ;       OFFSET=CODEA<1>*CNTR-ADDR<7,3>*CLASS<0>
                        ;
                        ;       NOTE:   CLASS<0>=0 FOR '5410', CLASS<0>=1 FOR '7632'
                        ;               CODEA IS SELECTED FROM ABOVE TABLE
                        ;
                        ;                       CODE    CODE    CODEA/CNTR-ADDR
                        ;               CLASS:  (5410)  (7632)  <1> /<7,3>
                        ;                       ------  ------  -----
025356  000     000     CODEC:  .BYTE   00,     00      ;0/00
025360  002     010             .BYTE   02,     10      ;0/01
025362  004     015             .BYTE   04,     15      ;0/10
025364  006     377             .BYTE   06,     -1      ;0/11
025366  000     000             .BYTE   00,     00      ;1/00
025370  004     010             .BYTE   04,     10      ;1/01
025372  017     015             .BYTE   17,     15      ;1/10
025374  010     016             .BYTE   10,     16      ;1/11
                        ;
                        ;////////////////////////////////////////////////////////////////////////////////
                        ;       CODE-D TABLE
                        ;
                        ;       OFFSET=CODEA<1>*CNTR-ADDR<5,1>*CLASS<0>
                        ;
                        ;       NOTE:   CLASS<0>=0 FOR '5410', CLASS<0>=1 FOR '7632'
                        ;               CODEA IS SELECTED FROM ABOVE TABLE
                        ;
                        ;                       CODE    CODE    CODEA/CNTR-ADDR
                        ;               CLASS:  (5410)  (7632)  <1> /<5,1>
                        ;                       ------  ------  -----
025376  000     000     CODED:  .BYTE   00,     00      ;0/00
025400  001     003             .BYTE   01,     03      ;0/01
025402  002     002             .BYTE   02,     02      ;0/10
025404  003     001             .BYTE   03,     01      ;0/11
025406  000     000             .BYTE   00,     00      ;1/00
025410  003     006             .BYTE   03,     06      ;1/01
025412  002     004             .BYTE   02,     04      ;1/10
025414  001     002             .BYTE   01,     02      ;1/11
                        ;////////////////////////////////////////////////////////////////////////////////
025416                  EXTOO1:                         ;EXIT THE HARD WAY


                        ;;*****************************************************************
                        ;*TEST 65        MULNET, MIER REGISTER DATA/SHIFTING
                        ;
                        ;       THIS TEST CHECKS THE FULL RANGE OF BITS IN THE MIER REGISTER,
                        ;       THE MIER REGISTER SHIFT LOGIC, AND THE MIERMUX SELECT LOGIC.
                        ;
                        ;       THE METHOD EMPLOYED INVOLVES RIPPLING A "1" THRU THE MIER
                        ;       REGISTER BITS<58:03>, MULTIPLYING THIS VALUE BY A CONSTANT
                        ;       (IN THE MAND REGISTER), AND THEN COMPARING THE RECEIVED
                        ;       RESULT (AFTER THE MULD) WITH THE EXPECTED.
                        ;
```

```
                        ;       ---------------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -INITIAL MAND, BEFORE ALIGNMENT
                        ;       MFAC1+  -INITIAL MIER, BEFORE ALIGNMENT
                        ;       MFAC2+  -EXPECTED PRODUCT
                        ;       MFAC3+  -MAND, AFTER ALIGNMENT
                        ;       MFAC4+  -MIER, AFTER ALIGNMENT
                        ;       MFAC5+  -RECEIVED PRODUCT
                        ;
                        ;       AC0     -(TEMP)
                        ;       AC1     -(TEMP), MIER/AFTER ALIGN.
                        ;       AC2     -RECEIVED PRODUCT
                        ;       AC3     -MAND/AFTER ALIGN.
                        ;
                        ;       R0      -(TEMP)/(SRCPTR)
                        ;       R1      -(TEMP)/(DSTPTR)
                        ;       R2      -(TEMP)/(CNTR)
                        ;       R3      -(NU)
                        ;       R4      -(NU)
                        ;       R5      -(NU)
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES-'MULD'-EXECUTION
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES-'MULD'-EXECUTION, MIER/MAND-FUNCTION-CONTROL,
                        ;         MIER/MAND-REGISTER-CLOCKS
                        ;
                        ;       FMUL/K10
                        ;         MIER/MAND-REGISTERS[FULL.WIDTH]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
025416  000004          ;ST65:  SCOPE
025420  170127  040240  ;       LDFPS   #040240                 ;INTR-DISABL/D-MODE/TRUNCATE
                        ;                                       ;
                        ;       ;-------PART 1: MIER<59:11>-------
025424  012700  025724  ;       MOV     #40$,R0                 ;INIT MAND/MIER/PROD IN
025430  012701  002646  ;       MOV     #MFAC0,R1               ;  IN MFAC0/1/2
025434  012702  000014  ;       MOV     #12.,R2                 ;  3-4 WORD CNST
025440  012021          ;8$:    MOV     (R0)+,(R1)+             ;
025442  077202          ;       SOB     R2,8$                   ;LOOP
                        ;                                       ;
                        ;       ;CGLC INIT MAND = (0040000000)#(0040000000)  (EACH 28. BITS  56. TOTAL)
025444  172437  002646  ;       LDD     MFAC0,AC0               ;USE MNS FOR:
025450  170401          ;       CLRD    AC1                     ;  F[AC1] <- F[AC0]-LEFT-6,
025452  170004          ;       MNS                             ;  E[AC1] <- E[AC0]-MINUS-4
025454  174103          ;       STD     AC1,AC3                 ;SAVE IN AC3
025456  174137  002676  ;       STD     AC1,MFAC3               ;  AND MEMORY (MAND)
                        ;                                       ;
```

```
8242                                        ;*LOOP ON MIER DATA ENTERS HERE
8243  025462  005237  002640          1$:   INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
8244  025466  172437  002656                LDD     MFAC1,AC0               ;USE MNS FOR:
8245  025472  170401                        CLRD    AC1                     ;  F[AC1] <- F[AC0]-LEFT-6,
8246  025474  170004                        MNS                             ;  E[AC1] <- E[AC0]-MINUS-4
8247  025476  174137  002706                STD     AC1,MFAC4               ;SAVE IN MEMORY (MIER)
8248                                                                        ;
8249  025502  104406                        ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
8250                                        ;--------ERROR-LOOP-ENTERS-HERE--------
8251                                                                        ;
8252  025504  174102                  63$:  STD     AC1,AC2                 ;COPY AC2 = MIER
8253  025506  171203                        MULD    AC3,AC2                 ;D-MODE, AC2 = (AC2)*(AC3)
8254                                                                        ;NORM. STEP ALWAYS "LEFT-4"; EADJ=(-4)
8255  025510  105737  002645                TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8256  025514  001373                        BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
8257                                                                        ;
8258  025516  174237  002716                STD     AC2,MFAC5               ;GET PRODUCT IN MEMORY
8259                                                                        ;
8260                                        ;COMPARE (EXP'D-PROD)=(RCV'D-PROD)
8261  025522  104425  002666  002716        CMP64M  ,MFAC2,MFAC5            ;64. BIT EQ/NE
8262  025530  001401                        BEQ     2$                      ;BR IF AGREE
8263  025532  104041                        ERROR   41                      ;NOPE - BAD RESULT
8264                                        ;"MULNET MULD RIPPLE A '1' THRU MIER ERROR"
8265                                        ;       HFPP AC1 = HFP AC1 /MIER/ BEFORE MULD
8266                                        ;       HFPP AC3 = HFP AC3 /MAND/ BEFORE MULD
8267                                        ;       RCVD AC2 = HFP AC2 /PRODUCT/ AFTER MULD, RECEIVED
8268                                        ;       EXPD AC2 = EXPECTED /PRODUCT/ AFTER MULD
8269                                                                        ;
8270                                        ;PRODUCT OK - GET NEXT DATA SET
8271  025534  106237  002666          2$:   ASRB    MFAC2+0                 ;NEW EXP'D PRODUCT
8272  025540  006037  002670                ROR     MFAC2+2                 ;  [64. BITS, RITE-1]
8273  025544  006037  002672                ROR     MFAC2+4                 ;
8274  025550  006037  002674                ROR     MFAC2+6                 ;
8275  025554  042737  000001  002674        BIC     #1,MFAC2+6              ;ALWAYS A ZERO
8276                                                                        ;
8277  025562  106237  002656                ASRB    MFAC1+0                 ;NEW MIER
8278  025566  006037  002660                ROR     MFAC1+2                 ;  [64. BITS, RITE-1]
8279  025572  006037  002662                ROR     MFAC1+4                 ;
8280  025576  006037  002664                ROR     MFAC1+6                 ;
8281                                                                        ;
8282                                        ;IF THE RIPPLED "1" FELL OUT THE BOTTOM, DONE WITH <59:11>
8283  025602  032737  000002  002664        BIT     #2,MFAC1+6              ;TEST FOR MIER<10>
8284  025610  001724                        BEQ     1$                      ;  MORE TO DO
8285                                                                        ;
8286                                        ;-------PART 2: MIER<10:03>-------
8287                                                                        ;
8288  025612  012700  025754                MOV     #41$,R0                 ;INIT MAND/MIER/PROD IN
8289  025616  012701  002646                MOV     #MFAC0,R1               ;  MFAC0/1/2
8290  025622  012702  000014                MOV     #12.,R2                 ;  3-4 WORD CNST
8291  025626  012021                  9$:   MOV     (R0)+,(R1)+             ;
8292  025630  077202                        SOB     R2,9$                   ;LOOP
8293                                                                        ;
8294  025632  172737  002646                LDD     MFAC0,AC3               ;INITIAL MAND
8295  025636  174337  002676                STD     AC3,MFAC3               ;SAVE IN MEMORY (MAND)
8296                                                                        ;
8297                                        ;*LOOP ON MIER DATA ENTERS HERE
```

K14


```
8298  025642  005237  002640          11$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
8299  025646  172537  002656                  LDD     MFAC1,AC1               ;INITIAL MIER
8300  025652  174137  002706                  STD     AC1,MFAC4               ;SAVE IN MEMORY (MIER)
8301                                                                          ;
8302  025656  104406                         ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
8303                                          ;--------ERROR-LOOP-ENTERS-HERE--------
8304                                                                          ;
8305  025660  174102                 62$:    STD     AC1,AC2                 ;COPY AC2=MIER
8306  025662  171203                         MULD    AC3,AC2                 ;D-MODE, AC2 = (AC2)*(AC3)
8307  025664  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8308  025670  001373                         BNE     62$                     ;  WITH/ LINE-CLOCK OFF, & FPS(F.D=1/CMM=0)
8309                                                                          ;
8310  025672  174237  002716                  STD     AC2,MFAC5               ;GET PRODUCT IN MEMORY
8311                                                                          ;
8312                                          ;COMPARE (EXP'D-PROD)=(RCV'D-PROD)
8313  025676  104425  002666  002716          CMP64M  ,MFAC2,MFAC5            ;64. BIT EQ/NE
8314  025704  001401                         BEQ     12$                     ;BR IF AGREE
8315  025706  104041                         ERROR   41                      ;NOPE - BAD RESULT
8316                                          ;"MULNET MULD RIPPLE A '1' THRU MIER ERROR"
8317                                          ;       HFPP AC1 = HFP AC1 /MIER/ BEFORE MULD
8318                                          ;       HFPP AC3 = HFP AC3 /MAND/ BEFORE MULD
8319                                          ;       RCVD AC2 = HFP AC2 /PRODUCT/ AFTER MULD, RECEIVED
8320                                          ;       EXPD AC2 = EXPECTED /PRODUCT/ AFTER MULD
8321                                                                          ;
8322                                          ;PRODUCT OK - GET NEXT DATA
8323  025710  006237  002674          12$:    ASR     MFAC2+6                 ;NEW EXP'D PRODUCT
8324                                                                          ;  [LOW 16. BITS, RITE-1]
8325                                                                          ;
8326  025714  006237  002664                  ASR     MFAC1+6                 ;NEW MIER
8327                                                                          ;  [LOW 16. BITS, RITE-1]
8328                                                                          ;
8329                                          ;IF THE RIPPLED '1' FELL OUT THE BOTTOM, DONE WITH <10:03>
8330  025720  103350                         BCC     11$                     ;BR IF NYD
8331                                                                          ;
8332  025722  000430                         BR      TST66           ;;      ;ALL DONE
8333                                                                          ;
8334                                          ;////////////////////////////////////////////////////////////////////////////////
8335                                          ;       DATA FOR MIER<59:11> TEST
8336  025724  042000  020000  000002  40$:    .WORD   042000,020000,000002,000000             ;MAND
8337  025732  000000
8338  025734  041002  000000  000000          .WORD   041002,000000,000000,000000             ;MIER
8339  025742  000000
8340  025744  040100  000000  002000          .WORD   040100,000000,002000,000000             ;PRODUCT
8341  025752  000000
8342                                          ;
8343                                          ;////////////////////////////////////////////////////////////////////////////////
8344                                          ;       DATA FOR MIER<10:03> TEST
8345  025754  040200  000000  004000  41$:    .WORD   040200,000000,004000,000000             ;MAND
8346  025762  000000
8347  025764  040000  000000  000000          .WORD   040000,000000,000000,000200             ;MIER
8348  025772  000200
8349  025774  040000  000000  004000          .WORD   040000,000000,004000,000200             ;PRODUCT
8350  026002  000200
8351
8352
8353
```

```
                        ;;*****************************************************************
                        ;*TEST 66       MULNET, MAND REGISTER DATA/SHIFTING
                        ;
                        ;       THIS TEST CHECKS THE FULL RANGE OF BITS IN THE MAND REGISTER,
                        ;       AND THE MAND REGISTER SHIFT LOGIC.
                        ;
                        ;       THE METHOD EMPLOYED INVOLVES RIPPLING A "1" THRU THE MAND
                        ;       REGISTER BITS<58:03>, MULTIPLYING THIS VALUE BY A CONSTANT
                        ;       (IN THE MIER REGISTER), AND THEN COMPARING THE RECEIVED
                        ;       RESULT (AFTER THE MULD) WITH THE EXPECTED.
                        ;
                        ;       LOWEST ORDER BITS IN MAND CHECKED IN PREVIOUS TESTS.
                        ;
                        ;       ---------------
                        ;       REGISTER/LOCATION USE:
                        ;
                        ;       MFAC0+  -INITIAL MIER, BEFORE ALIGNMENT
                        ;       MFAC1+  -INITIAL MAND, BEFORE ALIGNMENT
                        ;       MFAC2+  -EXPECTED PRODUCT
                        ;       MFAC3+  -MIER, AFTER ALIGNMENT
                        ;       MFAC4+  -MAND, AFTER ALIGNMENT
                        ;       MFAC5+  -RECEIVED PRODUCT
                        ;
                        ;       AC0     -(TEMP)
                        ;       AC1     -(TEMP), MAND/AFTER ALIGN.
                        ;       AC2     -RECEIVED PRODUCT
                        ;       AC3     -MIER/AFTER ALIGN.
                        ;
                        ;       R0      -(TEMP)/(SRCPTR)
                        ;       R1      -(TEMP)/(DSTPTR)
                        ;       R2      -(TEMP)/(CNTR)
                        ;       R3      -(NU)
                        ;       R4      -(NU)
                        ;       R5      -(NU)
                        ;
                        ;       ----------
                        ;       MODULE/ERROR INFO:
                        ;
                        ;       FNUA/K8
                        ;         CROM/LATCHES-'MULD'-EXECUTION
                        ;
                        ;       FEXP/K9
                        ;         CROM/LATCHES-'MULD'-EXECUTION, MIER/MAND-FUNCTION-CONTROL,
                        ;         MIER/MAND-REGISTER-CLOCKS
                        ;
                        ;       FMUL/K10
                        ;         MIER/MAND-REGISTERS[FULL.WIDTH]
                        ;
                        ;       FALU/K11
                        ;         [PREVIOUSLY VERIFIED]
                        ;
                        ;*****************************************************************
026004  000004          TST66:  SCOPE
026006  170127  040240          LDFPS   #040240                 ;INTR-DISABL/D-MODE/TRUNCATE
                                ;
026012  012700  026162          MOV     #40$,R0                 ;INIT MIER/MAND/PROD IN
```

```
8410  026016  012701  002646          MOV     #MFAC0,R1               ;  IN MFAC0/1/2
8411  026022  012702  000014          MOV     #12.,R2                 ;  3-4 WORD CNST
8412  026026  012021          8$:     MOV     (R0)+,(R1)+             ;
8413  026030  077202                  SOB     R2,8$                   ;LOOP
8414                                                                  ;
8415                                  ;CALC INIT MIER = (000)*(200)*(000)*(000)*(000)*(000)*(000)
8416  026032  172437  002646          LDD     MFAC0,AC0               ;USE MNS FOR:
8417  026036  170401                  CLRD    AC1                     ;  F[AC1] <- F[AC0]-LEFT-6,
8418  026040  170004                  MNS                             ;  E[AC1] <- E[AC0]-MINUS-4
8419  026042  174103                  STD     AC1,AC3                 ;SAVE IN AC3
8420  026044  174137  002676          STD     AC1,MFAC3               ;  AND MEMORY (MIER)
8421                                                                  ;
8422                                  ;*LOOP ON MAND DATA ENTERS HERE
8423  026050  172437  002656    1$:   LDD     MFAC1,AC0               ;USE MNS FOR:
8424  026054  170401                  CLRD    AC1                     ;  F[AC1] <- F[AC0]-LEFT-6,
8425  026056  170004                  MNS                             ;  E[AC1] <- E[AC0]-MINUS-4
8426  026060  174137  002706          STD     AC1,MFAC4               ;SAVE IN MEMORY (MAND)
8427                                                                  ;
8428  026064  104406                  ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
8429                                  ;--------ERROR-LOOP-ENTERS-HERE--------
8430                                                                  ;
8431  026066  174102          63$:    STD     AC1,AC2                 ;COPY AC2 = MAND
8432  026070  171203                  MULD    AC3,AC2                 ;D-MODE, AC2 = (AC2)*(AC3)
8433                                                                  ;NORM. STEP ALWAYS "LEFT-4"; EADJ=(-4)
8434  026072  105737  002645          TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8435  026076  001373                  BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
8436                                                                  ;
8437  026100  174237  002716          STD     AC2,MFAC5               ;GET PRODUCT IN MEMORY
8438                                                                  ;
8439                                  ;COMPARE (EXP'D-PROD)=(RCV'D-PROD)
8440  026104  104425  002666  002716  CMP64M   MFAC2,MFAC5            ;64. BIT EQ/NE
8441  026112  001401                  BEQ     2$                      ;BR IF AGREE
8442  026114  104042                  ERROR   42                      ;NOPE - BAD RESULT
8443                                  ;"MULNET MULD RIPPLE A '1' THRU MAND ERROR"
8444                                  ;       HFPP AC1 = HFP AC1 /MAND/ BEFORE MULD
8445                                  ;       HFPP AC3 = HFP AC3 /MIER/ BEFORE MULD
8446                                  ;       RCVD AC2 = HFP AC2 /PRODUCT/ AFTER MULD, RECEIVED
8447                                  ;       EXPD AC2 = EXPECTED /PRODUCT/ AFTER MULD
8448                                                                  ;
8449                                  ;PRODUCT OK - GET NEXT DATA SET
8450  026116  106237  002666    2$:   ASRB    MFAC2+0                 ;NEW EXP'D PRODUCT
8451  026122  006037  002670          ROR     MFAC2+2                 ;  [64. BITS, RITE-1]
8452  026126  006037  002672          ROR     MFAC2+4                 ;
8453  026132  006037  002674          ROR     MFAC2+6                 ;
8454                                                                  ;
8455  026136  106237  002656          ASRB    MFAC1+0                 ;NEW MAND
8456  026142  006037  002660          ROR     MFAC1+2                 ;  [64. BITS, RITE-1]
8457  026146  006037  002662          ROR     MFAC1+4                 ;
8458  026152  006037  002664          ROR     MFAC1+6                 ;
8459                                                                  ;
8460                                  ;IF THE RIPPLED "1" FELL OUT THE BOTTOM, DONE WITH <59:03>
8461  026156  103334                  BCC     1$                      ;BR IF NYD
8462                                                                  ;
8463  026160  000414                  BR      TST67           ;;      ;ALL DONE
8464                                                                  ;
8465                                  ;////////////////////////////////////////////////////////////
```

N14


```
 8466                                   ;       DATA FOR MAND<59:03> TEST
 8467  026162  042000  020000  000000  40$:    .WORD   042000,020000,000000,000000          ;MIER
 8468  026170  000000
 8469  026172  041002  000000  000000          .WORD   041002,000000,000000,000000          ;MAND
 8470  026200  000000
 8471  026202  040100  000000  000000          .WORD   040100,000000,000000,000000          ;PRODUCT
 8472  026210  000000
 8473                                   ;
 8474                                   ;////////////////////////////////////////////////////////////////
```

```
8475
8476
8477
8478
8479
8480
8481
8482
8483
8484
8485
8486
8487
8488
8489
8490
8491
8492
8493
8494
8495
8496
8497
8498
8499
8500
8501
8502
8503
8504
8505
8506
8507
8508
8509
8510
8511
8512
8513
8514
8515
8516
8517
8518
8519
8520
8521
8522
8523
8524
8525
8526
8527
8528
8529
8530
```

```
;;*****************************************************************
;*TEST 67       EXPNT, ECR & FCCR EXCEPTION/HFP(CC) CONDITIONS
;
;       THIS TEST CHECKS THE EXPNT "EXPONENT.CONDITION.REGISTER" (ECR)
;       AND THE "FLOATING.CONDITION.CODE.REGISTER" (FCCR) LOGIC.
;
;       USING THE "MULF" INSTRUCTION, OVERFLOW, UNDERFLOW, AND NO.EXCEPTION
;       CONDITIONS ARE GENERATED TO CHECK THAT THE EXCEPTION.CONDITION
;       LOGIC IS FUNCTIONING, AND THAT THE FLOATING CONDITION CODES
;       ARE SET/CLEARED AS EXPECTED.
;
;       NOTES:
;               FCC(N) = SD(1).H
;               FCC(Z) = ERZERO.H
;               FCC(V) = EOFLO.H
;               FCC(C) = 0, ALWAYS
;
;               EXPNT.OVERFLOW  = EOFLO.H = ER9(0)H*ER8(1)H          = 01XXXXXXXX
;               EXPNT.UNDERFLOW = EUFLO.H = ER9(0)h*ER8(0)H*ERZEROH = 0000000000
;                                                  + ER9(1)H          = 1XXXXXXXXX
;
;       ---------------
;       REGISTER/LOCATION USE:
;
;       AC0     -EXPNT OPERAND.A FOR MULF, E[SF]
;       AC1     -EXPNT OPERAND.B FOR MULF, E[DF]
;       AC2     -E[DF] RESULT ACC
;
;       MFAC0+  -COPY OF AC0, E[SF]
;       MFAC1+  -COPY OF AC1, E[DF]
;       MFAC2+  -EXPECTED PRODUCT FROM AC2
;       MFAC3+  -RECEIVED PRODUCT FROM AC2
;
;       R0      -INITIAL FPS BEFORE EXEC.
;       R1      -EXP'D FPS/CC AFTER EXEC.
;       R2      -RCV'D FPS/CC AFTER EXEC.
;       R3      -EXP'D FEC.CODE AFTER EXEC (10=OVF/12=UNF/377=NONE)
;       R4      -RCV'D FEC.CODE AFTER EXEC.
;       R5      -DATA TABLE PTR
;
;       ----------
;       MODULE/ERROR INFO:
;
;       FNUA/K8
;         CROM/LATCHES-'MULF'-EXECUTION
;
;       FEXP/K9
;         CROM/LATCHES-'MULF'-EXECUTION, SIGN.LOGIC
;         FCCR, ECR, EXCEPTION.GENERATE.LOGIC, BUT(EXCEPTION,EUFLO)
;
;       FMUL/K10
;         [PREVIOUSLY VERIFIED]
;
;       FALU/K11
;         [PREVIOUSLY VERIFIED]
```

```
8531                                    ;;*****************************************************************
8532  026212  000004                    TST67:  SCOPE
8533                                    ;
8534  026214  012705  026350                    MOV     #40$,R5                 ;DATA TABLE PTR
8535  026220  005037  002650                    CLR     MFAC0+2                 ;AC[SF], WORD.B IS ZERO
8536  026224  005037  002660                    CLR     MFAC1+2                 ;AC[DF], WORD.B IS ZERO
8537  026230  005037  002670                    CLR     MFAC2+2                 ;EXP'D PRODUCT, WORD.B IS ZERO
8538                                    ;
8539                                            ;*DATA TABLE LOOP ENTERS HERE*
8540  026234  005237  002640            10$:    INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
8541  026240  104416                            ZAPHFP                          ;INIT HFP, SET FEC=(377)
8542  026242  012500                            MOV     (R5)+,R0                ;GET INITIAL FPS
8543  026244  100514                            BMI     TST70           ;;      ;IF = -1, WE'RE DONE
8544  026246  012501                            MOV     (R5)+,R1                ;GET EXP'D FPS AFTER
8545  026250  012503                            MOV     (R5)+,R3                ;GET EXP'D FEC AFTER
8546  026252  012537  002646                    MOV     (R5)+,MFAC0+0           ;OPERAND.A, WORD.A
8547  026256  012537  002656                    MOV     (R5)+,MFAC1+0           ;OPERAND.B, WORD.A
8548  026262  012537  002666                    MOV     (R5)+,MFAC2+0           ;EXP'D RESULT, WORD.A
8549                                    ;
8550  026266  170100                            LDFPS   R0                      ;INITIAL FPS
8551  026270  172437  002646                    LDF     MFAC0,AC0               ;AC[SF]
8552  026274  172537  002656                    LDF     MFAC1,AC1               ;AC[DF]
8553                                    ;
8554  026300  104406                            ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
8555                                            ;--------ERROR-LOOP-ENTERS-HERE--------
8556                                    ;
8557  026302  174102                    63$:    STF     AC1,AC2                 ;SETUP AC[DF]
8558  026304  171200                            MULF    AC0,AC2                 ;DO THE MULTIPLY
8559  026306  105737  002645                    TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8560  026312  001373                            BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
8561                                    ;
8562  026314  170202                            STFPS   R2                      ;GET FPS/CC AFTER
8563  026316  170304                            STST    R4                      ;GET FEC AFTER
8564  026320  174237  002676                    STF     AC2,MFAC3               ;GET RESULT TO MEMORY
8565                                    ;
8566  026324  104426  002666  002676            CMP32M  ,MFAC2,MFAC3            ;(EXP'D.PRODUCT) = (RCV'D.PRODUCT) ??
8567  026332  001004                            BNE     30$                     ;IF DISAGREE, ERROR
8568                                    ;
8569  026334  020102                            CMP     R1,R2                   ;(EXP'D.FPS/CC.AFTER) = (RCV'D.FPS/CC.AFTER) ??
8570  026336  001002                            BNE     30$                     ;IF DISAGREE, ERROR
8571                                    ;
8572  026340  020304                            CMP     R3,R4                   ;(EXP'D.FEC.AFTER) = (RCV'D.FEC.AFTER) ??
8573  026342  001734                            BEQ     10$                     ;BR IF ALL 3 WERE OK
8574                                    ;
8575  026344  104111                    30$:    ERROR   111                     ;SIGNAL FPS/FEC/RESULT ERROR
8576                                            ;"ECR/FCCR EXCEPTION+FCC ERR"
8577                                            ;       PRV.FPS = PREVIOUS FPS/CC, PRIOR TO EXEC.
8578                                            ;       E-FPS   = EXP'D FPS/CC AFTER EXEC.
8579                                            ;       R-FPS   = RCV'D FPS/CC AFTER EXEC.
8580                                            ;       E-FEC   = EXP'D HFP FEC.CODE AFTER EXEC (10=OVF/12=UNF/377=NONE)
8581                                            ;       R-FEC   = RCV'D HFP FEC.CODE AFTER EXEC
8582                                            ;       HFPP AC0 = OPERAND.A FOR MULF, FROM AC0[SF]
8583                                            ;       HFPP AC1 = OPERAND.B FOR MULF, FROM AC1[DF]
8584                                            ;       EXPD AC2 = EXPECTED PRODUCT, FROM AC2[DF]
8585                                            ;       RCVD AC2 = RECEIVED PRODUCT, FROM AC2[DF]
8586                                    ;
```

```
8587  026346  000732                          BR      10$                     ;NEXT
8588                                  ;
8589                                  ;
8590                                  ;//////////////////////////////////////////////////////////////////////////
8591                                  ;
8592                                  ;       DATA FOR ABOVE TEST:
8593                                  ;
8594          000200                          X=200           ;EXPNT SHIFT.LEFT VALUE
8595          000000                          S0=000000/X     ;SIGN(0)
8596          177400                          S1=100000/X     ;SIGN(1)
8597                                  ;
8598          000010                          EOFLO=10        ;FEC.CODE (10) FOR OVERFLOW
8599          000012                          EUFLO=12        ;FEC.CODE (12) FOR UNDERFLOW
8600          000377                          NONE =377       ;DEFAULT CODE FOR NONE WRITTEN
8601                                  ;
8602                                  ;       --FPS.BEFORE-- --FPS.AFTER---  -FEC- ---AC0--- ---AC1--- -AC2.EXPO-
8603
8604  026350  043047  043050  000377  40$:    043040+↑B0111, 043040+↑B1000,  NONE, S0!201*X, S1!201*X, S1!201*X
8605  026356  040200  140200  140200
8606
8607
8608  026364  043041  143056  000010          043040+↑B0001, 143040+↑B1110, EOFLO, S1!301*X, S0!300*X, S1!000*X
8609  026372  160200  060000  100000
8610
8611  026400  043055  143042  000010          043040+↑B1101, 143040+↑B0010, EOFLO, S0!377*X, S0!377*X, S0!175*X
8612  026406  077600  077600  037200
8613
8614  026414  042051  042046  000377          042040+↑B1001, 042040+↑B0110,  NONE, S0!377*X, S1!377*X, S0!000*X
8615  026422  077600  177600  000000
8616
8617
8618  026430  043043  143054  000012          043040+↑B0011, 143040+↑B1100, EUFLO, S0!101*X, S1!100*X, S1!000*X
8619  026436  020200  120000  100000
8620
8621  026444  043057  143040  000012          043040+↑B1111, 143040+↑B0000, EUFLO, S1!001*X, S1!001*X, S0!201*X
8622  026452  100200  100200  040200
8623
8624  026460  041053  041044  000377          041040+↑B1011, 041040+↑B0100,  NONE, S1!001*X, S0!001*X, S0!000*X
8625  026466  100200  000200  000000
8626
8627
8628  026474  177777                          -1      ;<DONE>
8629                                  ;
8630                                  ;//////////////////////////////////////////////////////////////////////////
```

```
                        ;*****************************************************************
                        ;*TEST 70       'MPP' MAINT. INSTR - FUNCTIONAL TEST
                        ;
                        ;       THIS TEST PERFORMS A FUNCTIONAL CHECK OF THE FP11-E SPECIFIC
                        ;       MAINTENANCE INSTRUCTION "MPP", OR "MAINTENANCE PARTIAL PRODUCT".
                        ;
                        ;       THF FUNCTION OF THIS INSTRUCTION IS AS FOLLOWS:
                        ;
                        ;         INPUT ACC'S:  AC0             OUTPUT ACC'S:   AC1, AC2
                        ;
                        ;         1] MIER<07:00> = FSPAD[AC0]<42:35>
                        ;            MAND<27:00> = FSPAD[AC0]<30:03>
                        ;
                        ;         2] AR<59:00> = MULNET.SUM(MIER,MAND)
                        ;            SSPAD[AC1] = SSPAD[AC1]
                        ;            ESPAD[AC1] = EADJ(AR<59:54>)
                        ;            FSPAD[AC1] = MNETSUM<57:23>*ZEROES<22:03>
                        ;
                        ;         3] AR<53:00> = MULNET.SUM+CARRY(MIER,MAND)
                        ;            SSPAD[AC2] = SSPAD[AC2]
                        ;            ESPAD[AC2] = EADJ(AR<59:54>)
                        ;            FSPAD[AC2] = MNETSUM+CARRY<57:23>*ZEROES<22:03>
                        ;
                        ;         4] ALL RESULTS ARE INDEPENDENT OF FPS F/D-MODE, AND R/T-MODE.
                        ;            FPS CONDITION CODES ARE NOT CHANGED.
                        ;
                        ;*****************************************************************
026476  000004          TST70:  SCOPE
                                ;
026500  012737  000012  001342  MOV     #10.,$TIMES             ;DO 10. ITERATIONS OF THIS TEST
                                ;
026506  104420  002646          DBLDAT  ,MFAC0                  ;4 WORDS OF RANDOM DATA IN MFAC0
026512  104420  002656          DBLDAT  ,MFAC1                  ;  AND MFAC1
026516  104422                  RANFPS                          ;GENERATE A RANDOM FPS IN $FPS
                                                                ; [WITH FER=0, FID=1, FMM=0]
026520  105037  002650          CLRB    MFAC0+2                 ;SET MIER=(000) AT START
                                ;
                                ;*DATA LOOP ENTERS HERE*
                                ;GENERATE MFAC6 = EXPECTED RESULT IN HFP AC2 = MNET(S+C)
026524  005237  002640  1$:     INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
026530  012700  002726          MOV     #MFAC6,R0               ;
026534  012701  000004          MOV     #4,R1                   ;CLEAR MFAC6 TO START
026540  005020          11$:    CLR     (R0)+                   ;
026542  077102                  SOB     R1,11$                  ;
                                ;
026544  013700  002650          MOV     MFAC0+2,R0              ;MIER=MFAC0<42:35>
026550  042700  177400          BIC     #UB,R0                  ;
026554  005001                  CLR     R1                      ;MAND-H=(000000)
026556  013702  002652          MOV     MFAC0+4,R2              ;MAND-M=0000*MFAC0<30:19>
026562  042702  170000          BIC     #170000,R2              ;
026566  013703  002654          MOV     MFAC0+6,R3              ;MAND-L=MFAC0<18:03>
                                ;
                                ;MULT LOOP
026572  006200          12$:    ASR     R0                      ;MIER-RITE-1, C-BIT=LSB
026574  103014                  BCC     13$                     ;BR IF LSB=0, NO ADD
                                ;
```

```
8687  026576  060337  002732            ADD     R3,MFAC6+4          ;LSB=1, ADD MAND TO PARTIAL PRODUCT
8688  026602  005537  002730            ADC     MFAC6+2             ;
8689  026606  005537  002726            ADC     MFAC6+0             ;
8690  026612  060237  002730            ADD     R2,MFAC6+2          ; [48. BIT ADD]
8691  026616  005537  002726            ADC     MFAC6+0             ;
8692  026622  060137  002726            ADD     R1,MFAC6+0          ;
8693                                                                ;
8694  026626  005700            13$:    TST     R0                  ;DONE WHEN MIER = ZERO
8695  026630  001404                    BEQ     14$                 ;BR IF DONE
8696                                                                ;
8697  026632  006303                    ASL     R3                  ;
8698  026634  006102                    ROL     R2                  ;48. BIT ASL OF MAND
8699  026636  006101                    ROL     R1                  ;
8700  026640  000754                    BR      12$                 ;CONTINUE
8701                                                                ;
8702                                    ;DONE LOOPING               ;
8703  026642  012700  000004    14$:    MOV     #4,R0               ;PUT PRODUCT IN MFAC6<58:23>
8704  026646  006337  002732    15$:    ASL     MFAC6+4             ;
8705  026652  006137  002730            ROL     MFAC6+2             ; [48. BIT ASL 4 OF PRODUCT
8706  026656  006137  002726            ROL     MFAC6+0             ;   TO ALIGN W/ HFP]
8707  026662  077007                    SOB     R0,15$              ;
8708                                                                ;
8709  026664  013700  002726            MOV     MFAC6,R0            ;FRAC<59:54> IN R0<08:03>
8710  026670  104424                    EADJ                        ;CALC HFP EADJ: R0=EADJ[R0<8:3>]
8711  026672  072027  000007            ASH     #7,R0               ;ALIGN EXPONENT (LEFT-7)
8712  026676  042737  177600  002726    BIC     #^C177,MFAC6        ;CLEAR SIGN, EXP POSITIONS
8713  026704  050037  002726            BIS     R0,MFAC6            ;INSERT EXPONENT
8714                                                                ;
8715  026710  005737  002656            TST     MFAC1+0             ;TEST SIGN OF INITIAL AC2
8716  026714  100404                    BMI     16$                 ;BR IF A (1)
8717  026716  042737  100000  002726    BIC     #BIT15,MFAC6+0      ;INSERT A (0)=(+)
8718  026724  000403                    BR      17$                 ;
8719  026726  052737  100000  002726  16$:    BIS     #BIT15,MFAC6+0      ;INSERT A (1)=(-)
8720  026734                    17$:
8721                                                                ;
8722  026734  104406                    ERRPNT                      ;DONT CHANGE DATA IN ERROR LOOP
8723                                    ;--------ERROR-LOOP-ENTERS-HERE--------
8724                                                                ;
8725  026736  170127  040200            LDFPS   #040200             ;INTR DISABLE, D-MODE
8726  026742  172437  002646            LDD     MFAC0+0,AC0         ;INPUT DATA
8727  026746  172537  002652            LDD     MFAC0+4,AC1         ;PRELOAD OUTPUT AC'S
8728  026752  172637  002656            LDD     MFAC1+0,AC2         ; "
8729                                                                ;
8730  026756  170137  002610            LDFPS   $FPS                ;LOAD THE FPS
8731  026762  170005            63$:    MPP                         ;EXECUTE MAINT INSTR
8732  026764  105737  002645            TSTB    LPTITE              ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8733  026770  001374                    BNE     63$                 ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
8734                                                                ;
8735  026772  170237  002612            STFPS   FPS                 ;SAVE FPS/FEC/FEA AFTER
8736  026776  170337  002614            STST    FEC                 ;
8737                                                                ;
8738  027002  170127  040200            LDFPS   #040200             ;INTR DISABLE, D-MODE
8739  027006  012700  002666            MOV     #MFAC2,R0           ;
8740  027012  174020                    STD     AC0,(R0)+           ;MFAC2=HFP AC0 (INP)
8741  027014  174120                    STD     AC1,(R0)+           ;MFAC3=HFP AC1 (OUT) (S)
8742  027016  174210                    STD     AC2,(R0)            ;MFAC4=HFP AC2 (OUT) (S+C)
```

```
8743
8744
8745  027020  104425  002646  002666          ;*CHECK (ORIGINAL AC0) = (CURRENT AC0)
8746  027026  001401                          CMP64M  ,MFAC0,MFAC2
8747  027030  104010                          BEQ     .+4             ;BR IF OK
8748                                          ERROR   10              ;'MPP' ALTERED AC0 CONTENTS
8749
8750  027032  023737  002612  002610          ;*CHECK (FPS AFTER) = (FPS BEFORE)
8751  027040  001401                          CMP     FPS,$FPS
8752  027042  104011                          BEQ     .+4             ;BR IF OK
8753                                          ERROR   11              ;FPS WAS ALTERED
8754
8755  027044  104425  002706  002726          ;*CHECK (RECEIVED S+C/MFAC4) = (EXPECTED S+C/MFAC6)
8756  027052  001401                          CMP64M  ,MFAC4,MFAC6
8757  027054  104012                          BEQ     .+4             ;BR IF EQUAL
8758                                          ERROR   12              ;WRONG MNET(S+C) FROM HFP
8759
8760  027056  105237  002650                  ;*LOOP ON (000) TO (377) VALUES IN MIER
8761  027062  001220                          INCB    MFAC0+2         ;BUMP MIER
8762                                          BNE     1$              ;LOOP FOR (000)-(377)
8763
8764                                  ;;**********************************************************************
8765                                  ;*TEST 71        'MNS' MAINT. INSTR - FUNCTIONAL TEST
8766                                  ;
8767                                  ;       THIS TEST PERFORMS A FUNCTIONAL CHECK OF THE FP11-E SPECIFIC
8768                                  ;       MAINTENANCE INSTRUCTION "MNS", OR "MAINTENANCE NORMALIZATION SHIFT".
8769                                  ;
8770                                  ;       THE FUNCTION OF THIS INSTRUCTION IS AS FOLLOWS:
8771                                  ;
8772                                  ;         INPUT ACC'S:  AC0             OUTPUT ACC'S:   AC1
8773                                  ;
8774                                  ;         1] AR<59:35/03> = FSPAD[AC0]<59:35/03> UNDER F/D, R/T MODES
8775                                  ;
8776                                  ;         2] AR<59:00> = AR<59:00>-LEFT-2, VIA SHIFTER RIF 2
8777                                  ;
8778                                  ;         3] SSPAD[AC1] = SSPAD[AC1]
8779                                  ;            ESPAD[AC1] = EADJ(AR<59:54>) PLUS ESPAD[AC0]
8780                                  ;
8781                                  ;         4] AR<59:00> = NORMK-SHIFT( AR<59:00> )
8782                                  ;            FSPAD[AC1]<59:35/03> = AR<59:35/03> UNDER F/D MODES
8783                                  ;
8784                                  ;         5] FPS CONDITION CODES ARE NOT CHANGED.
8785                                  ;
8786                                  ;;**********************************************************************
8787  027064  000004                  $ST71:  SCOPE
8788  027066  012737  000012  001342          MOV     #10.,$TIMES             ;DO 10. ITERATIONS OF THIS TEST
8789                                          ;
8790  027074  104420  002646                  DBLDAT  ,MFAC0                  ;4 WORDS OF RANDOM DATA IN MFAC0
8791  027100  104420  002656                  DBLDAT  ,MFAC1                  ;  AND MFAC1
8792  027104  104422                          RANFPS                          ;GENERATE A RANDOM FPS IN $FPS
8793                                          ; [WITH FER=0, FID=1, FMM=0]
8794  027106  105037  002646                  CLRB    MFAC0+0                 ;SET FRAC<57:51>=(000) AT START
8795                                          ;
8796                                          ;*DATA LOOP ENTERS HERE*
8797                                          ;*GENERATE MFAC5 = EXPECTED RESULT IN HFP AC1 = NORMK(AR)
8798  027112  005237  002640          1$:     INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
```

```
 8799  027116  012704  002646              MOV     #MFAC0,R4       ;
 8800  027122  012400                      MOV     (R4)+,R0        ;WORD-A [F,D]
 8801  027124  012401                      MOV     (R4)+,R1        ;WORD-B [F,D]
 8802  027126  105737  002610              TSTB    $FPS            ;F(=0) OR D(=1) MODE ?
 8803  027132  100403                      BMI     11$             ;BR IF D-MODE
 8804  027134  005002                      CLR     R2              ;F-MODE, 32. LOB ARE ZEROES
 8805  027136  005003                      CLR     R3              ; [WORD-C,D]
 8806  027140  000402                      BR      12$             ;
 8807  027142  012402              11$:    MOV     (R4)+,R2        ;WORD-C [D]
 8808  027144  011403                      MOV     (R4),R3         ;WORD-D [D]
 8809                                                              ;
 8810  027146  012704  000002      12$:    MOV     #2,R4           ;LOOP TWICE
 8811  027152  000241                      CLC                     ;SETUP FOR TRUNCATE (FPS05=1)
 8812  027154  032737  000040  002610      BIT     #BIT05,$FPS     ;TEST R/T BIT
 8813  027162  001001                      BNE     13$             ;BR IF TRUNCATE
 8814  027164  000261                      SEC                     ;SETUP FOR ROUND (FPS05=0)
 8815                                                              ;USE "BIT" TO PRESERVE C-BIT
 8816  027166  032737  000200  002610  13$:     BIT     #BIT07,$FPS     ;F(=0) OR D(=1) MODE ?
 8817  027174  001402                      BEQ     15$             ;BR IF F-MODE
 8818  027176  006103              14$:    ROL     R3              ;D-MODE ROUND BIT INSERT
 8819  027200  006102                      ROL     R2              ;
 8820  027202  006101              15$:    ROL     R1              ;F-MODE ROUND BIT INSERT
 8821  027204  006100                      ROL     R0              ;
 8822  027206  000241                      CLC                     ;SHIFT ZEROES IN
 8823  027210  077406                      SOB     R4,14$          ;TWICE FOR 64. BITS/LEFT-2
 8824                                                              ;
 8825  027212  013737  002646  002716      MOV     MFAC0,MFAC5     ;COPY ESPAD[AC0] TO OUTPUT
 8826  027220  010046                      MOV     R0,-(SP)        ;SAVE FRAC
 8827  027222  104424                      EADJ                    ;CALC HFP EADJ: R0=EADJ[R0<8:3>]
 8828  027224  072027  000007              ASH     #7,R0           ;ALIGN TO EXP POSITION (LEFT-7)
 8829  027230  060037  002716              ADD     R0,MFAC5        ;ADJUST OUTPUT EXP
 8830  027234  042737  100177  002716      BIC     #100177,MFAC5   ;ZAP SIGN, FRAC
 8831                                                              ;
 8832  027242  012600                      MOV     (SP)+,R0        ;RESTORE FRAC
 8833  027244  042700  177000              BIC     #177000,R0      ;CLEAR SIGN, EXP
 8834                                                              ;
 8835  027250  032700  000400              BIT     #BIT08,R0       ;AR<59>=1 ?
 8836  027254  001405                      BEQ     16$             ;BR IF =0
 8837  027256  006200                      ASR     R0              ;NORMALIZE/RITE-1
 8838  027260  006001                      ROR     R1              ;
 8839  027262  006002                      ROR     R2              ;
 8840  027264  006003                      ROR     R3              ;
 8841  027266  000411                      BR      18$             ;AND DONE
 8842                                                              ;
 8843  027270  012704  000004      16$:    MOV     #4,R4           ;MAX OF LEFT-4
 8844  027274  105700              17$:    TSTB    R0              ;STOP WHEN AR<59:58>="01"
 8845  027276  100405                      BMI     18$             ;BR IF DONE
 8846  027300  006303                      ASL     R3              ;
 8847  027302  006102                      ROL     R2              ; [64. BIT/LEFT-1]
 8848  027304  006101                      ROL     R1              ;
 8849  027306  006100                      ROL     R0              ;
 8850  027310  077407                      SOB     R4,17$          ;MAX OF 4 TIMES
 8851                                                              ;
 8852  027312  012704  002716      18$:    MOV     #MFAC5,R4       ;
 8853  027316  042700  177600              BIC     #^C177,R0       ;ZAP SIGN, EXP
 8854  027322  050024                      BIS     R0,(R4)+        ;INSERT FRAC, WORD-A
```

```
8855  027324  010124                          MOV     R1,(R4)+                ;WORD-B
8856  027326  105737  002610                  TSTB    $FPS                    ;F(=0) OR D(=1) MODE ?
8857  027332  100404                          BMI     19$                     ;BR IF D-MODE
8858  027334  013702  002662                  MOV     MFAC1+4,R2              ;F-MODE, 32. LOB SAME AS PREV
8859  027340  013703  002664                  MOV     MFAC1+6,R3              ;
8860  027344  010224                  19$:    MOV     R2,(R4)+                ;WORD-C
8861  027346  010314                          MOV     R3,(R4)                 ;WORD-D
8862                                                                          ;[SIGN=0 TO START]
8863  027350  005737  002656                  TST     MFAC1+0                 ;TEST SIGN OF ORIG AC1
8864  027354  100003                          BPL     20$                     ;BR IF A (0)
8865  027356  052737  100000  002716          BIS     #BIT15,MFAC5+0          ;INSERT A (1)=(-)
8866  027364                          20$:                                    ;
8867                                                                          ;
8868  027364  104406                          ERRPNT                          ;DONT CHANGE DATA IN ERROR LOOP
8869                                           ;--------ERROR-LOOP-ENTERS-HERE--------
8870                                                                          ;
8871  027366  170127  040200                  LDFPS   #040200                 ;INTR DISABLE, D-MODE
8872  027372  172437  002646                  LDD     MFAC0+0,AC0             ;INPUT DATA
8873  027376  172537  002656                  LDD     MFAC1+0,AC1             ;PRELOAD OUTPUT AC'S
8874                                                                          ;
8875  027402  170137  002610                  LDFPS   $FPS                    ;LOAD THE FPS
8876  027406  170004                  63$:    MNS                             ;EXECUTE MAINT INSTR
8877  027410  105737  002645                  TSTB    LPTITE                  ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
8878  027414  001374                          BNE     63$                     ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
8879                                                                          ;
8880  027416  170237  002612                  STFPS   FPS                     ;SAVE FPS/FEC/FEA AFTER
8881  027422  170337  002614                  STST    FEC                     ;
8882                                                                          ;
8883  027426  170127  040200                  LDFPS   #040200                 ;INTR DISABLE, D-MODE
8884  027432  012700  002666                  MOV     #MFAC2,R0               ;
8885  027436  174020                          STD     AC0,(R0)+               ;MFAC2=HFP AC0 (INP)
8886  027440  174110                          STD     AC1,(R0)                ;MFAC3=HFP AC1 (OUT) (NORM)
8887                                                                          ;
8888                                           ;*CHECK (ORIGINAL AC0) = (CURRENT AC0)
8889  027442  104425  002646  002666          CMP64M  ,MFAC0,MFAC2            ;
8890  027450  001401                          BEQ     .+4                     ;BR IF OK
8891  027452  104010                          ERROR   10                      ;'MNS' ALTERED AC0 CONTENTS
8892                                                                          ;
8893                                           ;*CHECK (FPS AFTER) = (FPS BEFORE)
8894  027454  023737  002612  002610          CMP     FPS,$FPS                ;
8895  027462  001401                          BEQ     .+4                     ;BR IF OK
8896  027464  104011                          ERROR   11                      ;FPS WAS ALTERED
8897                                                                          ;
8898                                           ;*CHECK (RECEIVED NORMK[AR]/MFAC3) = (EXPECTED NORMK[AR]/MFAC5)
8899  027466  104425  002716  002676          CMP64M  ,MFAC5,MFAC3            ;
8900  027474  001401                          BEQ     .+4                     ;BR IF EQUAL
8901  027476  104013                          ERROR   13                      ;WRONG NORMK[AR] FROM HFP
8902                                                                          ;
8903                                           ;*LOOP FOR FRAC<57:51> = (000) TO (377)
8904  027500  105237  002646                  INCB    MFAC0+0                 ;BUMP FRACTION BITS
8905  027504  001202                          BNE     1$                      ;LOOP FOR (000) TO (377)
8906                                                                          ;
8907
8908                                   ;;**************************************************************
8909                                   ;*TEST 72        'MAS' MAINT. INSTR - FUNCTIONAL TEST
8910                                   ;
```

```
                                    ;         THIS TEST PERFORMS A FUNCTIONAL CHECK OF THE FP11-E SPECIFIC
                                    ;         MAINTENANCE INSTRUCTION "MAS", OR "MAINTENANCE ALIGNMENT SHIFT".
                                    ;
                                    ;         THE FUNCTION OF THIS INSTRUCTION IS AS FOLLOWS:
                                    ;
                                    ;           INPUT ACC'S:  AC0              OUTPUT ACC'S:   AC1, AC2
                                    ;
                                    ;           1] AR<59:00> = FSPAD[AC0]<59:00>
                                    ;              ER<8:0> = 0000*ESPAD[AC0]<5:0>
                                    ;              SHIFT-CODE = PRE-SHIFT-RESIDUE-ROM( ER<5:0> )
                                    ;              COUNTER<3:0> = PRE-SHIFT-QUOTIENT-ROM( ER<5:0> )
                                    ;              AR<59:00> = PRE-SHIFTER( AR<59:00> )
                                    ;
                                    ;           2] SSPAD[AC1] = SSPAD[AC1]
                                    ;              ESPAD[AC1] = EADJ( AR<59:54> )
                                    ;              FSPAD[AC1]<59:35/03> = AR<59:35/03> UNDER F/D-MODES
                                    ;
                                    ;           3] SSPAD[AC2] = SSPAD[AC2]
                                    ;              ESPAD[AC2] = INCREMENT-COUNTER-UNTIL-OVERFLOW(1-6)
                                    ;              FSPAD[AC2] = FSPAD[AC2]
                                    ;
                                    ;           4] FPS CONDITION CODES ARE NOT CHANGED.
                                    ;
                                    ;;*****************************************************************
027506  000004                      TST72:  SCOPE
                                                                            ;
027510  012737  000012  001342              MOV     #10.,$TIMES             ;DO 10. ITERATIONS OF THIS TEST
                                                                            ;
027516  104420  002646                      DBLDAT  ,MFAC0                  ;4 WORDS OF RANDOM DATA IN MFAC0
027522  104420  002656                      DBLDAT  ,MFAC1                  ;  AND MFAC1
027526  104422                              RANFPS                          ;GENERATE A RANDOM FPS IN $FPS
                                                                            ; [WITH FER=0, FID=1, FMM=0]
027530  042737  017600  002646              BIC     #017600,MFAC0+0         ;SET 6 LSB OF EXP TO (00) TO START
                                                                            ;
                                            ;*DATA LOOP ENTERS HERE*
                                            ;GENERATE MFAC5 = EXPECTED RESULT IN HFP AC1 = SHIFTED AC0
                                            ;GENERATE MFAC6 = EXPECTED RESULT IN HFP AC2 = CNTR
027536  005237  002640              1$:     INC     DWLOOP                  ;BUMP CLOCK IN.A.LOOP COUNT
027542  013737  002656  002726              MOV     MFAC1+0,MFAC6+0         ;
027550  013737  002660  002730              MOV     MFAC1+2,MFAC6+2         ;FRAC PART OF EXPEC AC2 SAME,
027556  013737  002662  002732              MOV     MFAC1+4,MFAC6+4         ;  ONLY EXP DIFF
027564  013737  002664  002734              MOV     MFAC1+6,MFAC6+6         ;
027572  042737  077600  002726              BIC     #077600,MFAC6+0         ;ZAP EXP
                                                                            ;
027600  013701  002646                      MOV     MFAC0+0,R1              ;GET AC0 WORD-A
027604  072127  177771                      ASH     #-7,R1                  ;SHIFT EXP RITE-7
027610  042701  177700                      BIC     #^C77,R1                ;USE 6 LSB ONLY
027614  005000                              CLR     R0                      ;ZAP HOB
027616  071027  000013                      DIV     #11.,R0                 ;FORM INT[EXP/11.]
                                            ;R0=QUOT, 0-5.          (TO CNTR)
                                            ;R1=REM, 0-10.          (TO SHIFTER)
027622  005200                              INC     R0                      ;CNTR IN RANGE 1.-6.
027624  072027  000007                      ASH     #7,R0                   ;ALIGN TC EXP (LEFT-7)
027630  050037  002726                      BIS     R0,MFAC6+0              ;INSERT INTO EXPTED AC2
                                                                            ;
027634  005201                              INC     R1                      ;SHIFTER, RANGE 1.-11.
```

```
 8967  027636  012700  002646          MOV     #MFAC0,R0       ;
 8968  027642  012002                  MOV     (R0)+,R2        ;GET 64. BIT INPUT #
 8969  027644  012003                  MOV     (R0)+,R3        ;
 8970  027646  012004                  MOV     (R0)+,R4        ;
 8971  027650  011005                  MOV     (R0),R5         ;
 8972  027652  042702  177600          BIC     #^C177,R2       ;ZAP SIGN, EXP
 8973  027656  052702  000200          BIS     #BIT07,R2       ;INSERT HIDDEN BIT
 8974  027662  006202          11$:    ASR     R2              ;
 8975  027664  006003                  ROR     R3              ;64. BIT SHIFT RIGHT,
 8976  027666  006004                  ROR     R4              ;  DEPENDING UPON COUNT
 8977  027670  006005                  ROR     R5              ;
 8978  027672  077105                  SOB     R1,11$          ;THE COUNT
 8979                                  ;                       ;
 8980                                  ;<R2:R5> IS THE ADJUSTED/SHIFTED AC0
 8981  027674  010200                  MOV     R2,R0           ;COPY AR<59:54>
 8982  027676  104424                  EADJ                    ;CALC HFP EADJ: R0=EADJ[R0<8:3>]
 8983  027700  072027  000007          ASH     #7,R0           ;SHIFT TO EXP (LEFT-7)
 8984  027704  050002                  BIS     R0,R2           ;AND INSERT INTO WORD-A
 8985                                  ;                       ;
 8986  027706  005737  002652          TST     MFAC0+4         ;GET SIGN OF ORIGINAL AC1
 8987  027712  100403                  BMI     12$             ;BR IF A (1)
 8988  027714  042702  100000          BIC     #BIT15,R2       ;INSERT A (0)=(+)
 8989  027720  000402                  BR      13$             ;
 8990  027722  052702  100000  12$:    BIS     #BIT15,R2       ;INSERT A (1)=(-)
 8991                                  ;                       ;
 8992  027726  105737  002610  13$:    TSTB    $FPS            ;F(=0) OR D(=1) MODE ?
 8993  027732  100404                  BMI     14$             ;BR IF D-MODE
 8994  027734  013704  002656          MOV     MFAC0+10,R4     ;F-MODE, KEEP ORIGINAL
 8995  027740  013705  002660          MOV     MFAC0+12,R5     ;  32. LOB
 8996                                  ;                       ;
 8997  027744  012700  002716  14$:    MOV     #MFAC5,R0       ;
 8998  027750  010220                  MOV     R2,(R0)+        ;EXPEC AC1, WORD-A
 8999  027752  010320                  MOV     R3,(R0)+        ;WORD-B
 9000  027754  010420                  MOV     R4,(R0)+        ;WORD-C
 9001  027756  010510                  MOV     R5,(R0)         ;WORD-D
 9002                                  ;                       ;
 9003  027760  104406                  ERRPNT                  ;DONT CHANGE DATA IN ERROR LOOP
 9004                                  ;--------ERROR-LOOP-ENTERS-HERE--------
 9005                                  ;                       ;
 9006  027762  170127  040200          LDFPS   #040200         ;INTR DISABLE, D-MODE
 9007  027766  172437  002646          LDD     MFAC0+0,AC0     ;INPUT DATA
 9008  027772  172537  002652          LDD     MFAC0+4,AC1     ;PRELOAD OUTPUT AC'S
 9009  027776  172637  002656          LDD     MFAC1+0,AC2     ;  "
 9010                                  ;                       ;
 9011  030002  170137  002610          LDFPS   $FPS            ;LOAD THE FPS
 9012  030006  170007          63$:    MAS                     ;EXECUTE MAINT INSTR
 9013  030010  105737  002645          TSTB    LPTITE          ;IF TIGHT LOOP-ON-ERROR SET, THEN HANG IN LOOP
 9014  030014  001374                  BNE     63$             ;  WITH/ LINE-CLOCK OFF, & FPS(FID=1/FMM=0)
 9015                                  ;                       ;
 9016  030016  170237  002612          STFPS   FPS             ;SAVE FPS/FEC/FEA AFTER
 9017  030022  170337  002614          STST    FEC             ;
 9018                                  ;                       ;
 9019  030026  170127  040200          LDFPS   #040200         ;INTR DISABLE, D-MODE
 9020  030032  012700  002666          MOV     #MFAC2,R0       ;
 9021  030036  174020                  STD     AC0,(R0)+       ;MFAC2=HFP AC0 (INP)
 9022  030040  174120                  STD     AC1,(R0)+       ;MFAC3=HFP AC1 (OUT) .SHIFT)
```

```
9023  030042  174210                    STD     AC2,(R0)                ;MFAC4=HFP AC2 (OUT) (EXP=CNTR)
9024                                                                      ;
9025                                    ;*CHECK (ORIGINAL AC0) = (CURRENT AC0)
9026  030044  104425  002646  002666    CMP64M  ,MFAC0,MFAC2            ;
9027  030052  001401                    BEQ     .+4                     ;BR IF OK
9028  030054  104010                    ERROR   10                      ;'MAS' ALTERED AC0 CONTENTS
9029                                                                      ;
9030                                    ;*CHECK (FPS AFTER) = (FPS BEFORE)
9031  030056  023737  002612  002610    CMP     FPS,$FPS                ;
9032  030064  001401                    BEQ     .+4                     ;BR IF OK
9033  030066  104011                    ERROR   11                      ;FPS WAS ALTERED
9034                                                                      ;
9035                                    ;*CHECK (RECEIVED SHIFT/MFAC3) = (EXPECTED SHIFT/MFAC5)
9036  030070  104425  002716  002676    CMP64M  ,MFAC5,MFAC3            ;
9037  030076  001401                    BEQ     .+4                     ;BR IF EQUAL
9038  030100  104014                    ERROR   14                      ;WRONG ALIGN-SHIFT FROM HFP
9039                                                                      ;
9040                                    ;*CHECK (RECEIVED CNTR/MFAC4) = (EXPECTED CNTR/MFAC6)
9041  030102  104425  002726  002706    CMP64M  ,MFAC6,MFAC4            ;
9042  030110  001401                    BEQ     .+4                     ;BR IF EQUAL
9043  030112  104015                    ERROR   15                      ;WRONG CNTR-INCR FROM HFP
9044                                                                      ;
9045                                    ;*LOOP ON 6 LSB OF EXPONENT = (00) TO (77)
9046  030114  062737  000200  002646    ADD     #000200,MFAC0+0         ;BUMP EXPONENT BY 1
9047  030122  032737  017600  002646    BIT     #017600,MFAC0+0         ;TEST 6 LSB
9048  030130  001202                    BNE     1$                      ;LOOP FOR (00) TO (77)
9049                                                                      ;
9050
9051                                                                      ;
9052                                    ;;**************************************************************
9053                                    ;;**************************************************************
9054                                    ;;**************************************************************
9055                                    ;;**************************************************************
9056                                    ;*TEST 73        ...EXIT TO EOP...
9057                                    ;;**************************************************************
9058  030132  000004                    TST73:  SCOPE
9059  030134  005037  001342                    CLR     $TIMES          ;NO ITERATIONS OF THIS "TEST"
9060  030140  005037  001214                    CLR     $ERFLG          ;NO ERRORS EITHER
9061                                                                      ;
9062  030144  005037  177546                    CLR     DW11LC          ;KILL CLOCK
9063                                                                      ;
9064  030150  012700  014507                    MOV     #014507,R0      ;INIT: /JAM/TRACK/GR/PS/FLAG/WHAMI/HFP/
9065  030154  076600  000352                    MED     .WINIT          ;DO IT
9066                                                                      ;
9067  030160  000137  030400                    JMP     $EOP            ;DONE
9068                                                                      ;
9069                                    ;;**************************************************************
9070                                    ;;**************************************************************
9071                                    ;;**************************************************************
```

```
                                .SBTTL  END OF PASS ROUTINE

                                ;;*****************************************************************
                                ;*INCREMENT THE PASS NUMBER ($PASS)
                                ;*IF THERES A MONITOR GO TO IT
                                ;*IF THERE ISN'T JUMP TO NEWPAS

030400                          $EOP:
030400  000004                          SCOPE                   ;FAKE OUT LAST TEST

030402  032777  002000  150644          BIT     #SW10,@SWR      ;IS BELL ON ERROR SET ?
030410  001002                          BNE     64$             ;YES, NO BELL ON PASS END
030412  104401  001346                  TYPE    ,$BELL          ;ELSE DING THE BELL
030416                          64$:                            ;
030416  005037  001212                  CLR     $TSTNM          ;;ZERO THE TEST NUMBER
030422  005037  001342                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
030426  005237  001364                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
030432  042737  100000  001364          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
030440  005327                          DEC     (PC)+           ;;LOOP?
030442  000001                  $EOPCT: .WORD   1
030444  003013                          BGT     $DOAGN          ;;YES
030446  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
030450  000001                  $ENDCT: .WORD   1
030452  030442                          $EOPCT
030454  013700  000042          $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
030460  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
030462  000005                          RESET                   ;;CLEAR THE WORLD
030464  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
030466  000240                          NOP                     ;;SAVE ROOM
030470  000240                          NOP                     ;;FOR
030472  000240                          NOP                     ;;ACT11
030474                          $DOAGN:
030474  000137                          JMP     @(PC)+          ;;RETURN
030476  003722                  $RTNAD: .WORD   ^NEWPAS
```

```
                                .SBTTL  INTERRUPT SERVICE ROUTINES

                                ;;******************************************************************

                                .SBTTL  ...DW11-L LINE CLOCK INTERRUPT SERVICE ROUTINE

030500  021637  002632    DW11_I: CMP     (SP),DWLOPC             ;SAME RETURN PC AS LAST TIME ?
030504  001033                    BNE     2$                      ;BR IF NOT

                                ;SAME RETURN PC AT LEAST 2 TIMES IN A ROW ...
030506  005337  002636            DEC     DWCNTR                  ;COUNT ANOTHER TIME
030512  002040                    BGE     4$                      ;BR IF COUNTED DOWN REQ'D # OF MATCHES

                                ;SAME RETURN PC # TIMES IN A ROW ...
030514  023737  002640  002642    CMP     DWLOOP,DWOLOP           ;CHECK IF HUNG COUNT CHANGED FROM BEFORE
030522  101026                    BHI     3$                      ;YES, SO WE'RE NOT HUNG, IN A LOOP INSTEAD

                                ;ELSE ASSUME PROC IS HUNG TRYING TO TALK TO FP11-E ...
030524  010637  002772            MOV     SP,OLDSP                ;GET OLD: SP/PC/PS
030530  011637  002766            MOV     (SP),OLDPC              ;
030534  016637  000002  002770    MOV     2(SP),OLDPS             ;
030542  017637  000000  002774    MOV     @0(SP),FPINST           ;GET THE OFFENDING INSTRUCTION

                                ;SAME RETURN PC # TIMES IN A ROW ... AND HUNG COUNT UNCHANGED
030550  105737  002644            TSTB    DWFLAG                  ;TEST ESCAPE ENABLE FLAG
030554  001001                    BNE     1$                      ;BR IF ESCAPE ENABLED

030556  104024                    ERROR   24                      ;"PROCESSOR HUNG" ERROR OTHERWISE
                                ;"PROC HUNG:  LINE CLOCK TIMEOUT"
                                ;       INSTR. = THE OFFENDING INSTRUCTION
                                ;       OLD-SF = SP AFTER TRAP TO LINE CLOCK ROUTINE
                                ;       OLD-PC = PC BEFORE TRAP, POINTS AT OFFENDING INSTRUCTION
                                ;       OLD-PS = PS BEFORE TRAP

030560  005737  002634    1$:     TST     DWESCP                  ;ESCAPE ADDRESS PRESENT ?
030564  001403                    BEQ     2$                      ;IF ZERO, NO
030566  013716  002634            MOV     DWESCP,(SP)             ;ELSE USE AS NEW RETURN PC
030572  000402                    BR      3$                      ;

030574  011637  002632    2$:     MOV     (SP),DWLOPC             ;GET NEW LAST OLD PC
030600  013737  003000  002636    MOV     DWICNT,DWCNTR           ;RESET COUNTER
030606  013737  002640  002642    MOV     DWLOOP,DWOLOP           ;RESET LOOP COUNT
030614  000002            4$:     RTI                             ;AND RETURN

                                ;;******************************************************************

                                .SBTTL  ...FPP INTERRUPT SERVICE ROUTINE

030616  010637  002772    FPPILT: MOV     SP,OLDSP                ;GET OLD: SP/PC/PS
030622  011637  002766            MOV     (SP),OLDPC              ;
030626  016637  000002  002770    MOV     2(SP),OLDPS             ;

030634  105737  002623            TSTB    NOFPIE                  ;1S=OK TO EXEC FP INSTR / 0S=DONT DO IT
030640  001411                    BEQ     FPPNFP                  ;BR IF TO NOT EXECUTE FP INSTR
```

```
9162  030642  170237  002612          STFPS   FPS                     ;OK, GET FPS STATUS WORD
9163  030646  170337  002614          STST    FEC                     ;  AND FEC, AND FEA, TOO
9164
9165  030652  105737  002622          TSTB    FPTPOK                  ;1S=TRAP IS OK / 0S=TRAP IS AN ERROR
9166  030656  001040                  BNE     FPPRTI                  ;BR IF TO IGNORE TRAP
9167
9168  030660  104001                  ERROR   01                      ;SIGNAL ILLEGAL/UNEXPECTED FPP TRAP
9169                                  ;"UNEXPECTED FPP TRAP TO (244)"
9170                                  ;       -FPS-- = FPS AFTER TRAP
9171                                  ;       -FEC-- = FEC AFTER TRAP
9172                                  ;       -FEA-- = FEA AFTER TRAP
9173                                  ;       OLD-SP = BM SP AFTER TRAP
9174                                  ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
9175                                  ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
9176  030662  000436                  BR      FPPRTI                  ;AND CONTINUE
9177
9178  030664  010046          FPPNFP: MOV     R0,-(SP)                ;SAVE R0, R1
9179  030666  010146                  MOV     R1,-(SP)                ;
9180
9181                                  ;*SIMULATE "STFPS  FPS" USING 'MED' INSTRUCTION
9182  030670  076600  000144          MED     ,RFLAG                  ;R0 = FLAGS*FPS<07:00>
9183  030674  010001                  MOV     R0,R1                   ;SAVE
9184  030676  076600  000036          MED     ,RFEC                   ;R0 = FPS<15:08>*FEC
9185  030702  042700  000377          BIC     #LB,R0                  ;FPS<15:08> LEFT
9186  030706  042701  177400          BIC     #UB,R1                  ;FPS<07:00> LEFT
9187  030712  050001                  BIS     R0,R1                   ;FPS<15:00> LEFT
9188  030714  010137  002612          MOV     R1,FPS                  ;STORE IT
9189
9190                                  ;*SIMULATE "STST  FEC" USING 'MED' INSTRUCTION
9191  030720  076600  000076          MED     ,RFEA                   ;R0 = FEA
9192  030724  010037  002616          MOV     R0,FEA                  ;STORE IT
9193  030730  076600  000036          MED     ,RFEC                   ;R0 = FPS<15:08>*FEC
9194  030734  042700  177400          BIC     #UB,R0                  ;FEC, GET WHOLE BYTE
9195  030740  010037  002614          MOV     R0,FEC                  ;STORE IT
9196
9197  030744  012601                  MOV     (SP)+,R1                ;RESTORE R0, R1
9198  030746  012600                  MOV     (SP)+,R0                ;
9199
9200  030750  105737  002622          TSTB    FPTPOK                  ;1S=TRAP IS OK / 0S=TRAP IS AN ERROR
9201  030754  001001                  BNE     FPPRTI                  ;BR IF TO IGNORE TRAP
9202
9203  030756  104002                  ERROR   02                      ;SIGNAL ILLEGAL/UNEXPECTED FPP TRAP, NO FP EXEC
9204                                  ;"UNEXPECTED FPP TRAP TO (244)"
9205                                  ;       -FPS-- = FPS AFTER TRAP
9206                                  ;       -FEC-- = FEC AFTER TRAP
9207                                  ;       -FEA-- = FEA AFTER TRAP
9208                                  ;       OLD-SP = BM SP AFTER TRAP
9209                                  ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
9210                                  ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
9211
9212  030760  105037  002622  FPPRTI: CLRB    FPTPOK                  ;CLEAR TRAP FLAG, IN CASE OF SUBSEQUENT TRAPS
9213  030764  005737  002620          TST     FPESCP                  ;ESCAPE ADDRESS EXIST ?
9214  030770  001402                  BEQ     FPPEND                  ;IF ZERO, NO
9215  030772  013716  002620          MOV     FPESCP,(SP)             ;GET ESCAPE ADDR FOR FP TRAP
9216  030776  000002          FPPEND: RTI                             ;CONTINUE,  RECOVER AT LAST TRAP ONLY
9217
```

```
9218
9219
9220                                    ;;*****************************************************************
9221
9222                                    .SBTTL  ...TRAP-TO-4 INTERRUPT SERVICE ROUTINE
9223
9224  031000  010637  002772            TRP004: MOV     SP,OLDSP                ;GET OLD: SP/PC/PS
9225  031004  011637  002766                    MOV     (SP),OLDPC              ;
9226  031010  016637  000002  002770            MOV     2(SP),OLDPS             ;
9227
9228  031016  032737  000200  177766            BIT     #BIT07,CPUERR           ;TRAP DUE TO A UBREAK?
9229  031024  001002                            BNE     UBRK04                  ;BR IF YES
9230
9231  031026  104003                            ERROR   03                      ;SOME OTHER ERROR
9232                                            ;"UNEXPECTED TRAP-TO-(4)"
9233                                            ;       CPUERR = CPU ERROR REGISTER, AFTER TRAP
9234                                            ;       MEMERR = MEMORY ERROR REGISTER, AFTER TRAP
9235                                            ;       OLD-SP = BM SP AFTER TRAP
9236                                            ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
9237                                            ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
9238  031030  000002                            RTI                             ;AND CONTINUE WHERE LEFT OFF ...
9239
9240  031032  005237  002630            UBRK04: INC     UBCNTR                  ;BUMP UBRK COUNTER
9241  031036  105737  002625                    TSTB    UBTPOK                  ;1S=UBRK IS OK / 0S=UBRK AN ERROR
9242  031042  001002                            BNE     UBRKOK                  ;BR IF OK
9243
9244  031044  104003                            ERROR   03                      ;SIGNAL ERROR
9245                                            ;"UNEXPECTED TRAP-TO-(4)"
9246                                            ;       CPUERR = CPU ERROR REGISTER, AFTER TRAP
9247                                            ;       MEMERR = MEMORY ERROR REGISTER, AFTER TRAP
9248                                            ;       OLD-SP = BM SP AFTER TRAP
9249                                            ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
9250                                            ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
9251  031046  000002                            RTI                             ;AND CONTINUE WHERE LEFT OFF ...
9252
9253  031050  010046                    UBRKOK: MOV     R0,-(SP)                ;RESET FLAG<8>, UBRK ENABLE
9254  031052  076600  000144                    MFD     RFLAG                   ;
9255  031056  052700  100000                    BIS     #BIT15,R0               ;
9256  031062  076600  000344                    MFD     WFLAG                   ;
9257  031066  012600                            MOV     (SP)+,R0                ;
9258
9259  031070  005737  002626                    TST     UBESCP                  ;ESCAPE ADDRESS EXIST ?
9260  031074  001402                            BEQ     10$                     ;BR IF NOT
9261  031076  013716  002626                    MOV     UBESCP,(SP)             ;ELSE RETURN TO IT
9262  031102  000002                    10$:    RTI                             ;AND CONTINUE WHERE LEFT OFF ...
9263
9264
9265                                    ;;*****************************************************************
9266
9267                                    .SBTTL  ...TRAP-TO-10 INTERRUPT SERVICE ROUTINE
9268
9269  031104  010637  002772            TRP010: MOV     SP,OLDSP                ;GET OLD: SP/PC/PS
9270  031110  011637  002766                    MOV     (SP),OLDPC              ;
9271  031114  016637  000002  002770            MOV     2(SP),OLDPS             ;
9272
9273  031122  104005                            ERROR   05                      ;FORCE AN ERROR
```

```
                                        ;"UNEXPECTED TRAP-TO-(10)"
                                        ;       CPUERR = CPU ERROR REGISTER, AFTER TRAP
                                        ;       MEMERR = MEMORY ERROR REGISTER, AFTER TRAP
                                        ;       OLD-SP = BM SP AFTER TRAP
                                        ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
                                        ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
031124  000002                          RTI                             ;AND CONTINUE WHERE LEFT OFF ...


                                ;;**************************************************************

                                .SBTTL  ...MEMORY/CACHE PARITY ERROR INTERRUPT SERVICE ROUTINE

031126  010637  002772          TRP114: MOV     SP,OLDSP                ;GET OLD: SP/PC/PS
031132  011637  002766                  MOV     (SP),OLDPC              ;
031136  016637  000002  002770          MOV     2(SP),OLDPS             ;

031144  104004                          ERROR   04                      ;SIGNAL ERROR
                                        ;"UNEXPECTED TRAP-TO-(114)"
                                        ;       CPUERR = CPU ERROR REGISTER, AFTER TRAP
                                        ;       MEMERR = MEMORY ERROR REGISTER, AFTER TRAP
                                        ;       OLD-SP = BM SP AFTER TRAP
                                        ;       OLD-PC = BM PC AFTER TRAP (RETURN PC)
                                        ;       OLD-PS = BM PS BEFORE/AT-TIME-OF TRAP
031146  000002                          RTI                             ;AND CONTINUE WHERE LEFT OFF ...


                                ;;**************************************************************
```

```
                        .SBTTL  MISC. SUPPORT SUBROUTINES

                        ;;******************************************************************

                        .SBTTL  ...RANDOM "FPS" SUBROUTINE (RANFPS)

                        ;*      GENERATES A PSEUDO-RANDOM FPS VALUE IN "$FPS"
                        ;*
                        ;*      CALLED BY:      RANFPS
                        ;*

                        ;       BITS OF FPS GENERATED AS FOLLOWS:
                        ;
                        ;       BIT  FCN  VALUE                 BIT  FCN  VALUE
                        ;       ---  ---  ----------            ---  ---  ----------
                        ;       15   FER    0                   07   FD   $PASS<2>
                        ;       14   FID    1                   06   FL   $PASS<1>
                        ;       13    0     0                   05   FT   $PASS<0>
                        ;       12    0     0                   04   FMM    0
                        ;       11  FIUV  RANDOM<11>            03   FN   RANDOM<03>
                        ;       10   FIU  RANDOM<10>            02   FZ   RANDOM<02>
                        ;       09   FIV  RANDOM<09>            01   FV   RANDOM<01>
                        ;       08   FIC  RANDOM<08>            00   FC   RANDOM<00>
                        ;

031150  010046          $RNFPS: MOV     R0,-(SP)                ;SAVE R0
031152  004737  031276          JSR     PC,$RAND                ;RANDOM BITS
031156  113700  001364          MOVB    $PASS,R0                ;SET <FD,FL,FT>=$PASS<2:0>
031162  072027  000005          ASH     #5,R0                   ;
031166  042700  177437          BIC     #^C340,R0               ;
031172  013746  031376          MOV     $LONUM,-(SP)            ;RANDOM BITS
031176  042716  170360          BIC     #170360,(SP)            ;CLEAR UNUSED
031202  052600                  BIS     (SP)+,R0                ;MERGE
031204  052700  040000          BIS     #BIT14,R0               ;FID=1
031210  010037  002610          MOV     R0,$FPS                 ;STORE IT
031214  012600                  MOV     (SP)+,R0                ;RESTORE R0
031216  000002                  RTI                             ;AND RETURN


                        ;;******************************************************************

                        .SBTTL  ...RANDOM FLOATING POINT DATA SUBROUTINES (DBLDAT, SGLDAT)

                        ;*      GENERATES RANDOM NUMBER OPERANDS FOR DATA
                        ;*
                        ;*      CALLED BY:      DBLDAT          ;4 WORDS
                        ;*                      ADDR(DESTINATION)
                        ;*              OR
                        ;*                      SGLDAT          ;2 WORDS
                        ;*                      ADDR(DESTINATION)


031220  010446          $SNGL:  MOV     R4,-(SP)                ;SAVE R4
031222  017604  000002          MOV     @2(SP),R4               ;R4 = ADDR(DEST)
031226  000411                  BR      RAND2                   ;GET 2 WORDS
```

```
                                ;SAVE R4
                                ;R4 = ADDR(DEST)
                                ;GET 4 WORDS

                                ;GET 2 WORDS
                                ;D-MODE WORD "A"
                                ;D-MODE WORD "B"

                                ;GET 2 WORDS
                                ;D-MODE WORD "C" / F-MODE WORD "A"
                                ;D-MODE WORD "D" / F-MODE WORD "B"

                                ;RESTORE R4
                                ;BUMP RETURN
                                ;AND RETURN

                        .SBTTL  RANDOM NUMBER GENERATOR ROUTINE

                        ;;**************************************************************
                        ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
                        ;*WITH A RANGE OF 0 TO 2(+33)-1.
                        ;*CALL:
                        ;*      JSR     PC,$RAND        ;;CALL THE ROUTINE
                        ;*      RETURN                  ;;RETURN HERE THE RANDOM
                        ;*                              ;;NUMBER WILL BE IN
                        ;*                              ;;$HINUM,$LONUM

031276                  $RAND:
031276  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
031300  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
031302  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
031304  013700  031376          MOV     $LONUM,R0       ;;SET R0 WITH LOW
031310  013701  031374          MOV     $HINUM,R1       ;;SET R1 WITH HIGH
031314  012702  177771          MOV     #-7,R2          ;;SET SHIFT COUNT
031320  006300          1$:     ASL     R0              ;;SHIFT R0 LEFT AND
031322  006101                  ROL     R1              ;;ROTATE CARRY INTO R1 AND
031324  005202                  INC     R2              ;;CHECK FOR DONE
031326  001374                  BNE     1$              ;;CONTINUE SHIFT LOOP
031330  063700  031376          ADD     $LONUM,R0       ;;ADD NUMBER TO MAKE X 129
031334  005501                  ADC     R1              ;;PROPOGATE CARRY
031336  063701  031374          ADD     $HINUM,R1       ;;ADD NUMBER TO MAKE X 129
031342  062700  001057          ADD     #1057,R0        ;;ADD LOW CONSTANT
031346  005501                  ADC     R1              ;;PROPOGATE CARRY
031350  062701  047401          ADD     #47401,R1       ;;ADD HIGH CONSTANT
031354  010037  031376          MOV     R0,$LONUM       ;;SAVE R0
031360  010137  031374          MOV     R1,$HINUM       ;;SAVE R1
031364  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
031366  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
031370  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
031372  000207                  RTS     PC              ;;RETURN
031374  176543          $HINUM: .WORD   176543
031376  123456          $LONUM: .WORD   123456

                        ;;**************************************************************
```

```
                                .SBTTL  ...INITIALIZE HFP/WFP, FPS/FEC/FEA (ZAPHFP, ZAPWFP)

                                ;*      THIS ROUTINE GIVES THE FP11-E A "FPP(INIT)" [VIA UCON]
                                ;*      AND ALSO SETS:  FEC=(377)
                                ;*                      FEA=(177777)
                                ;*                      FPS=(040000) FER=0,FID=1,FMM=0
                                ;*
                                ;*      AND EXITS WITH HFP ENABLED  [FLAGS=1] IF CALLED VIA "ZAPHFP".
                                ;*       OR EXITS WITH HFP DISABLED [FLAGS=0] IF CALLED VIA "ZAPWFP",
                                ;*      CSP/FPP CONSTANTS ARE ALSO RESTORED
                                ;*
                                ;*      CALLED BY:      ZAPHFP          ;DOIT, AND ENABLE HFP ON EXIT
                                ;*
                                ;*      OR
                                ;*
                                ;*                      ZAPWFP          ;DOIT, AND DISABLE HFP ON EXIT
                                ;*

031400  010046                  $ZPWFP: MOV     R0,-(SP)                ;SAVE R0
031402  005046                          CLR     -(SP)                   ;FLAG<5>=0, FOR DISABLE HFP
031404  000403                          BR      $ZPXFP                  ;
031406  010046                  $ZPHFP: MOV     R0,-(SP)                ;SAVE R0
031410  012746  010000                  MOV     #010000,-(SP)           ;FLAG<5>=1, FOR ENABLE HFP
                                ;
031414  012700  004000          $ZPXFP: MOV     #004000,R0              ;DISABLE HFP, CSP INVALID, DISABLE UBRK(BM)
031420  076600  000344                  MED     ,WFLAG                  ;  INTO FLAGS
                                ;
031424  012700  177777                  MOV     #177777,R0              ;ALL ONES
031430  076600  000236                  MED     ,WFEC                   ;(377) -> FEC
031434  170127  040000                  LDFPS   #040000                 ;WFP EXEC, FER=0, FID=1, FMM=0
031440  076600  000276                  MED     ,WFEA                   ;(177777) -> FEA
031444  076600  000266                  MED     ,WFPA                   ;(177777) -> FPA
                                ;
031450  012700  000001                  MOV     #000001,R0              ;SELECT FPP(INIT)
031454  076600  000352                  MED     ,WINIT                  ;EXEC BM INIT ROUTINE
                                ;
031460  076600  000144                  MED     ,RFLAG                  ;FLAGS -> R0
031464  052600                          BIS     (SP)+,R0                ;ENABLE/DISABLE HFP, FLAG<5>
031466  076600  000344                  MED     ,WFLAG                  ;WRITE BACK
                                ;
031472  012600                          MOV     (SP)+,R0                ;RESTORE R0
031474  000002                          RTI                             ;AND RETURN


                                ;;*****************************************************************

                                .SBTTL  ...NORMALIZATION COUNT GENERATION SUBROUTINE (EADJ)

                                ;*      GENERATES "EADJ" VALUE OF HFPP USING R0<08:03> = AR<59:54>
                                ;*      RESULT, IN RANGE +1 / -4, RETURNED IN R0
                                ;*
                                ;*      CALLED BY:      EADJ            ;EXEC SUBR   R0 IN / R0 OUT
                                ;*

031476  010446                  $EADJ:  MOV     R4,-(SP)                ;SAVE R3, R4
031500  010346                          MOV     R3,-(SP)                ;
```

```
                                                                ;
031502  012704  031526          MOV     #EADJT-2,R4           ;ADDR(EADJ TABLE)
031506  005724          8$:     TST     (R4)+                 ;BUMP PAST PREV EADJ VALUE
031510  012403                  MOV     (R4)+,R3              ;GET BIT WORKING ON, FROM TABLE
031512  001402                  BEQ     10$                   ;IF ALL ZERO, DONE
031514  030300                  BIT     R3,R0                 ;TEST THIS BIT OF PATTERN
031516  001773                  BEQ     8$                    ;BR IF ZERO, TO TRY NEXT LSB
031520  011400          10$:    MOV     (R4),R0               ;NONZERO, GET CURRENT EADJ VALUE
                                                              ;
031522  012603                  MOV     (SP)+,R3              ;RESTORE R3, R4
031524  012604                  MOV     (SP)+,R4              ;
031526  000002                  RTI                           ;AND RETURN

                        ;               BIT OF  EADJ    BIT OF
                        ;                 R0    VALUE     AR
                        ;               ------  -----   ------
031530  000400  000001  EADJT:  .WORD   BIT08,   +1      ;AR<59>="1"
031534  000200  000000          .WORD   BIT07,    0      ;AR<58>="1"
031540  000100  177777          .WORD   BIT06,   -1      ;AR<57>="1"
031544  000040  177776          .WORD   BIT05,   -2      ;AR<56>="1"
031550  000020  177775          .WORD   BIT04,   -3      ;AR<55>="1"
031554  000010  177774          .WORD   BIT03,   -4      ;AR<54>="1"
031560  000000  177774          .WORD      0,   -4      ;AR<59:54>="000000", NORMA. OVERFLOW

                        ;;*****************************************************************

                        .SBTTL  ...FRACTION ADJUSTMENT ROUTINE (FIXFRA)

                        ;*      INSERTS, USING "EADJ" VALUE IN CURRENT EXPONENT, BITS
                        ;*      <59:58> OF FRACTION.  OTHER BITS OF DATA WORD ARE ZEROED.
                        ;*      IN THE CASE WHEN FRAC<59>=1, FRAC<58> IS UNDETERMINED.
                        ;*      (SINCE EADJ SEES ONLY HIGHEST BIT).  IN THIS CASE, FRAC<58>=0.
                        ;*
                        ;*      CALLED BY:      FIXFRA                  ;EXEC SUBR
                        ;*                      ADDR(DATA)              ;POINT TO HI WORD FP DATA
                        ;*
                        ;*              EADJ    SIGN    FRAC<59:58>
                        ;*              ----    ----    ---------------------------
                        ;*               +1     (0)     "10"    [FORCE FRAC<58>="0"]
                        ;*                0     (0)     "01"
                        ;*              ELSE    (0)     "00"

                        $FIXFR: MOV     R0,-(SP)              ;SAVE R0-R1
                                MOV     R1,-(SP)              ;
031564  000004          MOV     @4(SP),R1             ;R1=ADDR(WORD-A)
031570  011100          MOV     (R1),R0               ;R0=WORD-A
031574  032700  077400  BIT     #077400,R0            ;EADJ=(0) OR (1) ?
031576  001005          BNE     1$                    ;BR IF NEITHER
031602  042700  177400  BIC     #^C377,R0             ;ZAP SIGN, EXP
031604  062700  000200  ADD     #BIT07,R0             ;FORM FRAC<59:58>
031610  000402          BR      2$                    ;DONE
031614  042700  177600  1$:     BIC     #^C177,R0     ;FRAC<59:58>="00"
031616  010011          2$:     MOV     R0,(R1)       ;STORE NEW FRAC HI WORD
031622  012601          MOV     (SP)+,R1              ;RESTORE R0-R1
031624  012600          MOV     (SP)+,R0              ;
031626
```

```
9525  031630  062716  000002          ADD     #2,(SP)                 ;FIX RETURN ADDRESS
9526  031634  000002                  RTI                             ;AND RETURN
9527
9528
9529                                  ;;*****************************************************************
```

```
                                .SBTTL  64. BIT ARITHMETIC/LOGICAL FUNCTION SUBROUTINES

                                ;;******************************************************************

                                .SBTTL  ...64. BIT "ASHC" ROUTINE (ASH64I, ASH64M)

                                ;*      THIS ROUTINE OPERATES ON 64. BITS OF MEMORY DATA TO SIMULATE
                                ;*      THE "ASHC" INSTRUCTION.  WORKS THE SAME WAY, EXCEPT THE
                                ;*      PSW CONDITION CODES ARE NOT SET.
                                ;*
                                ;*      CALLED BY:      ASH64M          ;EXEC ASHC
                                ;*                      ADDR(COUNT)     ;COUNT, +LEFT / -RITE
                                ;*                      ADDR(DATA)      ;64. BITS OF DATA
                                ;*      OR
                                ;*                      ASH64I          ;EXEC ASHC
                                ;*                      COUNT           ;COUNT IS IN NEXT WORD
                                ;*                      ADDR(DATA)      ;DATA POINTER
                                ;*

031636  010046          $AS64M: MOV     R0,-(SP)                ;SAVE R0-R4
031640  010146                  MOV     R1,-(SP)                ;
031642  010246                  MOV     R2,-(SP)                ;
031644  010346                  MOV     R3,-(SP)                ;
031646  010446                  MOV     R4,-(SP)                ;
031650  016600  000012          MOV     12(SP),R0               ;POINT AT ADDR(COUNT)
031654  013004                  MOV     @(R0)+,R4               ;GET R4=COUNT
031656  000410                  BR      $AS64                   ;CONT

031660  010046          $AS64I: MOV     R0,-(SP)                ;SAVE R0-R4
031662  010146                  MOV     R1,-(SP)                ;
031664  010246                  MOV     R2,-(SP)                ;
031666  010346                  MOV     R3,-(SP)                ;
031670  010446                  MOV     R4,-(SP)                ;
031672  016600  000012          MOV     12(SP),R0               ;POINT AT THE COUNT
031676  012004                  MOV     (R0)+,R4                ;GET R4=COUNT

031700  011003          $AS64:  MOV     (R0),R3                 ;POINT R3 AT DATA
031702  010346                  MOV     R3,-(SP)                ;SAVE RESULT PTR FOR LATER
031704  012300                  MOV     (R3)+,R0                ;GET FIRST 16. BITS
031706  012301                  MOV     (R3)+,R1                ;NEXT 16.
031710  012302                  MOV     (R3)+,R2                ;NEXT 16.
031712  011303                  MOV     (R3),R3                 ;LAST 16.
031714  005704                  TST     R4                      ;TEST COUNT
031716  100427                  BMI     10$                     ;<0 - RITE
031720  001457                  BEQ     30$                     ;=0 - DONE
                                                                ;>0 - LEFT

                                ;+ = LEFT
031722  020427  000100  5$:     CMP     R4,#64.                 ;SHIFT LEFT >= 64. ?
031726  002402                  BLT     6$                      ;BR IF LEFT 1.-63. BITS
031730  005000                  CLR     R0                      ;>=64. BITS,
031732  000447                  BR      21$                     ;  RESULT IS ALL ZERO
031734  162704  000020  6$:     SUB     #16.,R4                 ;DO 16. BIT ASL'S FIRST
031740  002405                  BLT     7$                      ;
031742  010100                  MOV     R1,R0                   ;16. BIT ASL
031744  010201                  MOV     R2,R1                   ;  WITH BRUTE FORCE DATA MOVE
```

```
9586  031746  010302                  MOV     R3,R2                   ;
9587  031750  005003                  CLR     R3                      ;  SHIFT IN ZEROS
9588  031752  000770                  BR      6$                      ;AGAIN
9589  031754  062704  000020  7$:     ADD     #16.,R4                 ;FIX COUNT
9590  031760  001437          8$:     BEQ     30$                     ;DONE WHEN COUNT=0
9591  031762  006303                  ASL     R3                      ;
9592  031764  006102                  ROL     R2                      ;1. BIT ASL
9593  031766  006101                  ROL     R1                      ;  ON 64. DATA BITS
9594  031770  006100                  ROL     R0                      ;
9595  031772  005304                  DEC     R4                      ;ADJUST COUNT
9596  031774  000771                  BR      8$                      ;AGAIN
9597
9598                                  ;- = RITE
9599  031776  020427  177700  10$:     CMP     R4,#-64.                ;SHIFT RITE >= 64. BITS ?
9600  032002  003421                  BLE     20$                     ;BR IF >63. BIT SHIFT
9601  032004  062704  000020  11$:     ADD     #16.,R4                 ;DO 16. BIT ASR'S FIRST
9602  032010  003005                  BGT     12$                     ;
9603  032012  010203                  MOV     R2,R3                   ;16. BIT ASR WITH
9604  032014  010102                  MOV     R1,R2                   ;  BRUTE FORCE DATA MOVE
9605  032016  010001                  MOV     R0,R1                   ;
9606  032020  006700                  SXT     R0                      ;SHIFT IN SIGN
9607  032022  000770                  BR      11$                     ;AGAIN
9608  032024  162704  000020  12$:     SUB     #16.,R4                 ;FIX COUNT
9609  032030  001413          13$:     BEQ     30$                     ;DONE WHEN COUNT=0
9610  032032  006200                  ASR     R0                      ;
9611  032034  006001                  ROR     R1                      ;1. BIT ASR OF
9612  032036  006002                  ROR     R2                      ;  64. DATA BITS
9613  032040  006003                  ROR     R3                      ;
9614  032042  005204                  INC     R4                      ;ADJUST COUNT
9615  032044  000771                  BR      13$                     ;AGAIN
9616
9617  032046  005700          20$:    TST     R0                      ;TEST BIT<15>
9618  032050  006700                  SXT     R0                      ;AND PROPOGATE THROUGHOUT
9619  032052  006701          21$:    SXT     R1                      ;THE WHOLE THING
9620  032054  006702                  SXT     R2                      ;
9621  032056  006703                  SXT     R3                      ;
9622
9623                                  ;DONE, STORE ANSWER
9624  032060  012604          30$:    MOV     (SP)+,R4                ;RETRIEVE RESULT PTR
9625  032062  010024                  MOV     R0,(R4)+                ;STORE
9626  032064  010124                  MOV     R1,(R4)+                ;  64.
9627  032066  010224                  MOV     R2,(R4)+                ;    BITS
9628  032070  010314                  MOV     R3,(R4)                 ;
9629  032072  012604                  MOV     (SP)+,R4                ;RESTORE REGISTERS
9630  032074  012603                  MOV     (SP)+,R3                ;
9631  032076  012602                  MOV     (SP)+,R2                ;
9632  032100  012601                  MOV     (SP)+,R1                ;
9633  032102  012600                  MOV     (SP)+,R0                ;
9634  032104  062716  000004          ADD     #4,(SP)                 ;FIX RETURN ADDRESS
9635  032110  000002                  RTI                             ;AND RETURN
9636
9637
9638                          ;;*****************************************************************
9639
9640                          .SBTTL  ...64. BIT "SUB" ROUTINE (SUB64M)
9641
```

```
                                ;*      THIS ROUTINE OPERATES ON 64. BITS OF MEMORY DATA TO SIMULATE
                                ;*      THE "SUB" INSTRUCTION.  WORKS THE SAME WAY, EXCEPT THE
                                ;*      PSW CONDITION CODES ARE NOT SET.
                                ;*
                                ;*      CALLED BY:      SUB64M          ;EXEC SUB
                                ;*                      ADDR(SRC)       ;SOURCE DATA ADDR
                                ;*                      ADDR(DST)       ;DESTINATION DATA ADDR
                                ;*

032112  010046          $SB64M: MOV     R0,-(SP)                ;SAVE R0-R1
032114  010146                  MOV     R1,-(SP)                ;
032116  016601  000004          MOV     4(SP),R1                ;POINT AT ADDR(SRC)
032122  012100                  MOV     (R1)+,R0                ;POINT R0 AT SRC DATA
032124  011101                  MOV     (R1),R1                 ;POINT R1 AT DST DATA

032126  062700  000006          ADD     #6,R0                   ;[S3]
032132  062701  000006          ADD     #6,R1                   ;[D3]
032136  161011                  SUB     (R0),(R1)               ;D3=(D3)-(S3)
032140  005641                  SBC     -(R1)                   ;D2=(D2)-(C)
032142  005641                  SBC     -(R1)                   ;D1=(D1)-(C)
032144  005641                  SBC     -(R1)                   ;D0=(D0)-(C)
032146  062701  000004          ADD     #4,R1                   ;[D2]
032152  164011                  SUB     -(R0),(R1)              ;D2=(D2)-(S2)
032154  005641                  SBC     -(R1)                   ;D1=(D1)-(C)
032156  005641                  SBC     -(R1)                   ;D0=(D0)-(C)
032160  062701  000002          ADD     #2,R1                   ;[D1]
032164  164011                  SUB     -(R0),(R1)              ;D1=(D1)-(S1)
032166  005641                  SBC     -(R1)                   ;D0=(D0)-(C)
032170  164011                  SUB     -(R0),(R1)              ;D0=(D0)-(S0)

                                ;DONE, RESTORE REGISTERS AND RETURN
032172  012601                  MOV     (SP)+,R1                ;RESTORE R0-R1
032174  012600                  MOV     (SP)+,R0                ;
032176  062716  000004          ADD     #4,(SP)                 ;FIX RETURN ADDRESS
032202  000002                  RTI                             ;AND RETURN


                        ;;**********,************************************************************

                        .SBTTL  ...MULTIPLE WORD COMPARISON ROUTINES (CMP64M, CMP32M, CMPXXM)

                                ;*      THESE ROUTINES COMPARE 2 MULTIPLE WORD VALUES ON AN
                                ;*      EQUAL/NOTEQUAL BASIS, SETTING THE CONDITION CODES TO
                                ;*      REFLECT THE RESULT.
                                ;*
                                ;*      CALLED BY:      CMP***          ;COMPARE
                                ;*                      ADDR(SRC)       ;1ST OPERAND
                                ;*                      ADDR(DST)       ;2ND OPERAND
                                ;*                      BEQ/BNE XXX     ;TEST RESULT
                                ;*

032204  113737  002610  002624 $CMPWD: MOVB    $FPS,FPLENF             ;$FPS-FD SPECIFIES 2.4 WORDS
032212  000406                  BR      $CMPX                   ;
032214  105037  002624          $CMP2W: CLRB    FPLENF                  ;FORCE FD=0
032220  000403                  BR      $CMPX                   ;
032222  112737  177777  002624 $CMP4W: MOVB    #-1,FPLENF              ;FORCE FD=1
```

```
 9698
 9699  032230  010046                $CMPX:  MOV     R0,-(SP)                ;SAVE R0, R1
 9700  032232  010146                        MOV     R1,-(SP)                ;
 9701
 9702  032234  016600  000004                MOV     4(SP),R0                ;RETURN PC
 9703  032240  012001                        MOV     (R0)+,R1                ;ADDR(SRC)
 9704  032242  011000                        MOV     (R0),R0                 ;ADDR(DST)
 9705
 9706  032244  042766  000017  000006        BIC     #17,6(SP)               ;ZAP RETURN CC FOR Z=0
 9707
 9708  032252  022021                        CMP     (R0)+,(R1)+             ;WORD-A
 9709  032254  001014                        BNE     10$                     ;BR IF DIFF
 9710  032256  022021                        CMP     (R0)+,(R1)+             ;WORD-B
 9711  032260  001012                        BNE     10$                     ;BR IF DIFF
 9712
 9713  032262  105737  002624                TSTB    FPLENF                  ;F(=0) OR D(=1) MODE ?
 9714  032266  100004                        BPL     9$                      ;BR IF F-MODE
 9715
 9716  032270  022021                        CMP     (R0)+,(R1)+             ;WORD-C
 9717  032272  001005                        BNE     10$                     ;BR IF DIFF
 9718  032274  021011                        CMP     (R0),(R1)               ;WORD-D
 9719  032276  001003                        BNE     10$                     ;BR IF DIFF
 9720
 9721  032300  052766  000004  000006  9$:   BIS     #4,6(SP)                ;SET Z=1 FOR EQUAL
 9722  032306  012601                10$:    MOV     (SP)+,R1                ;RESTORE R0, R1
 9723  032310  012600                        MOV     (SP)+,R0                ;
 9724  032312  062716  000004                ADD     #4,(SP)                 ;FIX RETURN ADDRESS
 9725  032316  000002                        RTI                             ;AND RETURN
 9726
 9727
 9728                                ;;*****************************************************************
```

```
                                .SBTTL  --SYSMAC SUPPORT ROUTINES--

                                ;;*****************************************************************

                                .SBTTL  SCOPE HANDLER ROUTINE

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

032320                          $SCOPE:
                                ;////////////////////////////////////////////////////////////////
032320  105037  002645                CLRB    LPTITE          ;DISABLE TIGHT LOOP SWITCH
                                ;////////////////////////////////////////////////////////////////
032324  105037  002625                CLRB    UBTPOK          ;CLEAR BM UBRK TRAP OK FLAG
032330  005037  002626                CLR     UBESCP          ;CLEAR BM UBRK ESCAPE ADDR
032334  076600  000144                MED     ,RFLAG          ;GET FLAGS
032340  042700  100000                BIC     #BIT15,R0       ;ZAP UBRK FLAG
032344  076600  000344                MED     ,WFLAG          ;WRITE FLAGS
                                ;////////////////////////////////////////////////////////////////
032350  005037  002620                CLR     FPESCP          ;CLEAR FP TRAP ESCAPE ADDRESS
032354  105037  002622                CLRB    FPTPOK          ;CLEAR FP TRAP OK
                                ;////////////////////////////////////////////////////////////////
032360  105037  002644                CLRB    DWFLAG          ;CLEAR LINE CLOCK ESC ENABLE
032364  005037  002634                CLR     DWESCP          ;CLEAR LINE CLOCK ESC ADDRESS
032370  005037  002632                CLR     DWLOPC          ;CLEAR LINE CLOCK LAST OLD PC
032374  005037  002640                CLR     DWLOOP          ;RESET LOOP/HUNG COUNTER
032400  005037  002642                CLR     DWOLDP          ;RESET OLD " " "
032404  012737  000006  002636        MOV     #DW$CNT,DWCNTR  ;RESET MATCH COUNTER TO DEFAULT
032412  012737  000006  003000        MOV     #DW$CNT,DWICNT  ;RESET MASTER MATCH COUNT
                                ;////////////////////////////////////////////////////////////////
032420  105737  002623                TSTB    NOFPIE          ;ABLE TO EXEC FP  ?
032424  001403                      BEQ     20$             ;BR IF NOT
032426  013703  001262              MOV     $FPBRK,R3       ;IF OK, GET HFP UBRK ADDR INTO R3
032432  170003                      LDUB                    ;AND THEN INTO HFP
032434                          20$:                        ;
                                ;////////////////////////////////////////////////////////////////
                                ;;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002
                                ;;OTHERWISE CONTINUE
032434  021627  001002                CMP     (SP),#1002      ;;UNEXPECTED TRAP OR INTERRRUPT
032440  101002                      BHI     1$              ;;ARE TRAPPED HERE VIA IOT
032442  000137  032724              JMP     $ERROR          ;;GO PROCESS UNEXPECTED TRAP
032446  032777  040000  146600  1$: BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
032454  001114                      BNE     $OVER           ;;YES IF SW14=1
                                ;*****START OF CODE FOR THE XOR TESTER*****
032456  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
032460  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
032464  012737  032504  000004        MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
```

```
9785  032472  005737  177060          TST     @#177060        ;;TIME OUT ON XOR?
9786  032476  012637  000004          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
9787  032502  000463                  BR      $SVLAD          ;;GO TO THE NEXT TEST
9788  032504  022626          5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
9789  032506  012637  000004          MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
9790  032512  000423                  BR      7$              ;;LOOP ON THE PRESENT TEST
9791  032514                  6$:;*****END OF CODE FOR THE XOR TESTER*****
9792  032514  032777  000400  146532  BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
9793  032522  001404                  BEQ     2$              ;;BR IF NO
9794  032524  023737  001260  001212  CMP     $LPTST,$TSTNM   ;;ON THE RIGHT TEST?
9795  032532  001465                  BEQ     $OVER           ;;BR IF YES
9796  032534  005737  001214  2$:     TST     $ERFLG          ;;HAS AN ERROR OCCURRED?
9797  032540  001421                  BEQ     3$              ;;BR IF NO
9798  032542  023737  001230  001214  CMP     $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
9799  032550  101015                  BHI     3$              ;;BR IF NO
9800  032552  032777  001000  146474  BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
9801  032560  001404                  BEQ     4$              ;;BR IF NO
9802  032562  013737  001222  001220  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
9803  032570  000446                  BR      $OVER
9804  032572  005037  001214  4$:     CLR     $ERFLG          ;;ZERO THE ERROR FLAG
9805  032576  005037  001342          CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
9806  032602  000415                  BR      1$              ;;ESCAPE TO THE NEXT TEST
9807  032604  032777  004000  146442  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
9808  032612  001011                  BNE     1$              ;;BR IF YES
9809  032614  005737  001364          TST     $PASS           ;;IF FIRST PASS OF PROGRAM
9810  032620  001406                  BEQ     1$              ;;      INHIBIT ITERATIONS
9811  032622  005237  001216          INC     $ICNT           ;;INCREMENT ITERATION COUNT
9812  032626  023737  001342  001216  CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
9813  032634  002024                  BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
9814  032636  012737  000001  001216  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
9815  032644  013737  032722  001342          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
9816  032652  005237  001212  $SVLAD: INC     $TSTNM          ;;COUNT TEST NUMBERS
9817  032656  013737  001212  001362          MOV     $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
9818  032664  011637  001220                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
9819  032670  011637  001222                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
9820  032674  005037  001344                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
9821  032700  012737  000001  001230          MOV     #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TES
9822  032706  013777  001212  146342  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
9823  032714  013716  001220                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
9824  032720  000002                  RTI                     ;;FIXES PS
9825  032722  000620          $MXCNT: 400.                    ;;MAX. NUMBER OF ITERATIONS
9826
9827
9828
9829                            ;;**************************************************************
9830
9831                            .SBTTL  ERROR HANDLER ROUTINE
9832
9833                            ;;**************************************************************
9834                            ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
9835                            ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
9836                            ;*AND GO TO $TYPERR ON ERROR
9837                            ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
9838                            ;*SW15=1         HALT ON ERROR
9839                            ;*SW13=1         INHIBIT ERROR TYPEOUTS
9840                            ;*SW10=1         BELL ON ERROR
```

```
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
                                        $ERROR:
032724                                          CLR     DW11LC          ;ZAP CLOCK
032724  005037  177546                          MOV     R0,EREG0        ;DISPLAY R0
032730  010037  002746                          MOV     R1,EREG1        ;        R1
032734  010137  002750                          MOV     R2,EREG2        ;        R2
032740  010237  002752                          MOV     R3,EREG3        ;        R3
032744  010337  002754                          MOV     R4,EREG4        ;        R4
032750  010437  002756                          MOV     R5,EREG5        ;        R5
032754  010537  002760                          MOV     R6,EREG6        ;GET R6(SP) BEFORE TRAP
032760  010637  002762                          MOV     #4,EREG6        ;
032764  062737  000004  002762                  ADD     (SP),EREG7      ;PC -> ERROR CALL
032772  011637  002764                  7$:     INC     $ERFLG          ;;SET THE ERROR FLAG
032776  005237  001214                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
033002  001775                                  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
033004  013777  001212  146244                  BIT     #BIT10,@SWR     ;;BELL ON ERROR?
033012  032777  002000  146234                  BEQ     1$              ;;NO - SKIP
033020  001402                                  TYPE    ,$BELL          ;;RING BELL
033022  104401  001346                  1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
033026  005237  001224                          MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
033032  011637  001232                          SUB     #2,$ERRPC
033036  162737  000002  001232                  MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
033044  117737  146162  001226                  BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
033052  032777  020000  146174                  BNE     20$             ;;SKIP TYPEOUTS
033060  001055                                  CMP     (SP),#1002      ;;IF RETURN PC LESS THAN 1002
033062  021627  001002                          BHI     12$             ;;ERROR IS ILLEGAL TRAP
033066  101046                          ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
                                                MOV     4(SP),$ERRPC    ;;GET PC AT TIME OF FALSE TRAP
033070  016637  000004  001232                  SUB     #2,$ERRPC       ;;ADJUST PC
033076  162737  000002  001232                  TYPE    ,10$            ;;TYPE HEADER
033104  104401  033150                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
033110  013746  001232                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
033114  104402                                  TYPE    ,11$
033116  104401  033156                          SUB     #4,(SP)         ;;GET FALSE TRAP VECTOR ADDR
033122  162716  000004                          MOV     (SP),$ERRPC
033126  011637  001232                          MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
033132  013746  001232                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
033136  104402                                  TYPE    ,$CRLF
033140  104401  001353                          CMP     (SP)+,(SP)+     ;;POP FALSE TRAP VECTOR PC&ADDR
033144  022626                                  BR      20$
033146  000422                          10$:    .ASCIZ  <200>'PC= '
033150  050200  036503  000040          11$:    .ASCIZ  '  UNEXPECTED TRAP TO '
033156  020040  047125  054105
033164  042520  052103  042105
033172  052040  040522  020120
033200  047524  000040
                                                .EVEN
033204                                  12$:
033204  004737  033422                          JSR     PC,$TYPERR      ;;GO TO USER ERROR ROUTINE
033210  104401  001353                          TYPE    ,$CRLF
033214                                  20$:
033214  122737  000001  001376                  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
033222  001007                                  BNE     2$              ;;NO,SKIP APT ERROR REPORT
033224  113737  001226  033236                  MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
```

```
 9897  033232  004737  034342            21$:    JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 9898  033236     000                             .BYTE   0
 9899  033237     000                             .BYTE   0
 9900  033240  000777                      22$:    BR      22$             ;;APT ERROR LOOP
 9901  033242  005777  146006             2$:     TST     @SWR            ;;HALT ON ERROR
 9902  033246  100001                             BPL     3$              ;;SKIP IF CONTINUE
 9903  033250  000000                             HALT                    ;;HALT ON ERROR!
 9904  033252  032777  001000  145774     3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
 9905  033260  001437                             BEQ     4$              ;;BR IF NO
 9906  033262  013716  001222                     MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
 9907  033266  032777  000040  145760             BIT     #SW05,@SWR      ;TIGHT LOOP SELECTED ?
 9908  033274  001436                             BEQ     5$              ;BR IF NOT
 9909  033276  112737  000377  002645             MOVB    #377,LPTITE     ;YES, SET SWITCH
 9910  033304  010046                             MOV     R0,-(SP)        ;SAVE R0
 9911  033306  076600  000144                     MED     ,RFLAG          ;SAVE OLD/EXISTING FLAGS
 9912  033312  010046                             MOV     R0,-(SP)        ;ON THE STACK
 9913  033314  005000                             CLR     R0              ;FLAGS FOR WFP*VALID
 9914  033316  076600  000344                     MED     ,WFLAG          ;INTO FLAG<5:4>
 9915  033322  170200                             STFPS   R0              ;GET OLD FPS
 9916  033324  042700  000020                     BIC     #BIT04,R0       ;SET FMM=0 (NO UBRK JAMS, PLEASE)
 9917  033330  052700  040000                     BIS     #BIT14,R0       ;SET FID=1 (NO FP TRAPS, PLEASE)
 9918  033334  170100                             LDFPS   R0              ;RESTORE FPS
 9919  033336  012600                             MOV     (SP)+,R0        ;RESTORE PREVIOUS FLAGS
 9920  033340  076600  000344                     MED     ,WFLAG          ;FROM STACK
 9921  033344  011600                             MOV     (SP),R0         ;RESTORE R0
 9922  033346  010316                             MOV     R3,(SP)         ;SAVE R3
 9923  033350  013703  001262                     MOV     $FPBRK,R3       ;GET UBRK ADDR, FOR SYNC PULSE
 9924  033354  170003                             LDUB                    ;INTO HFP UBRK
 9925  033356  012603                             MOV     (SP)+,R3        ;RESTORE R3
 9926  033360  005737  001344            4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
 9927  033364  001402                             BEQ     5$              ;;BR IF NONE
 9928  033366  013716  001344                     MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 9929  033372                             5$:
 9930  033372  022737  030464  000042             CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
 9931  033400  001001                             BNE     6$              ;;BRANCH IF NO
 9932  033402  000000                             HALT                    ;;YES
 9933  033404                             6$:
 9934  033404  105737  002645                     TSTB    LPTITE          ;ONLY IF NO TIGHT-LOOP SELECTED, THEN:
 9935  033410  001003                             BNE     13$             ;INTR ENABLE LINE CLOCK
 9936  033412  012737  000100  177546             MOV     #BIT6,DW11LC    ;
 9937  033420                             13$:                            ;
 9938  033420  000002                             RTI                     ;;RETURN
 9939
 9940
 9941
 9942                                     ;;*****************************************************************
 9943
 9944                                     .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE (MODIFIED SYSMAC)
 9945
 9946                                     ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
 9947                                     ;*ERROR IS TO BE REPORTED.  IT THEN OBTAINS, FROM THE "ERROR TABLE",
 9948                                     ;*($ERRTB) THE ERROR MESSAGE, DATA HEADER, AND DATA VALUES TO PRINT.
 9949                                     ;*THIS ROUTINE ALWAYS PRINTS $TESTN AND $ERRPC AS THE FIRST TWO DATA
 9950                                     ;*ELEMENTS (WITH APPROPRIATE HEADERS).
 9951
 9952  033422                             $TYPERR:
```

```
 9953  033422  010046                  MOV     R0,-(SP)        ;SAVE R0
 9954  033424  010146                  MOV     R1,-(SP)        ;SAVE R1
 9955  033426  005000                  CLR     R0              ;PICKUP ITEM INDEX
 9956  033430  153700  001226          BISB    @#$ITEMB,R0     ;
 9957  033434  001004                  BNE     1$              ;IF ITEM NUMBER FROM ERROR 0,
 9958                                                          ;JUST TYPE PC OF ERROR
 9959  033436  013746  001232          MOV     $ERRPC,-(SP)    ;GET ERROR PC FOR TYPEOUT
 9960  033442  104402                  TYPOC                   ;TYPE OCTAL, ALL DIGITS
 9961  033444  000473                  BR      15$             ;EXIT
 9962  033446  005300          1$:     DEC     R0              ;ADJUST 1-N TO 0-NM1
 9963  033450  006300                  ASL     R0              ;ADJUST ERROR # FOR TABLE INDEX
 9964  033452  006300                  ASL     R0              ;OF 8. BYTES/ENTRY
 9965  033454  006300                  ASL     R0              ;
 9966  033456  062700  001406          ADD     #$ERRTB,R0      ;FORM TABLE PTR
 9967  033462  012037  033472          MOV     (R0)+,2$        ;PICKUP "ERROR MESSAGE" PTR
 9968  033466  001404                  BEQ     3$              ;SKIP TYPEOUT IF NULL
 9969  033470  104401                  TYPE                    ;TYPE "ERROR MESSAGE"
 9970  033472  000000          2$:     .WORD   0               ;"ERROR MESSAGE" PTR HERE
 9971  033474  104401  001353          TYPE    ,$CRLF          ;CR & LF
 9972  033500  104401  033654  3$:     TYPE    ,11$            ;"TEST #    ERR PC" HEADER
 9973  033504  012037  033514          MOV     (R0)+,4$        ;PICKUP "DATA HEADER" PTR
 9974  033510  001402                  BEQ     5$              ;SKIP TYPEOUT IF NULL
 9975  033512  104401                  TYPE                    ;TYPE "DATA HEADER"
 9976  033514  000000          4$:     .WORD   0               ;"DATA HEADER" PTR HERE
 9977  033516  104401  001353  5$:     TYPE    ,$CRLF          ;CR & LF
 9978  033522  017746  000120          MOV     @8$,-(SP)       ;($TESTN)
 9979  033526  104402                  TYPOC                   ;OCTAL W/ LEADING ZEROS
 9980  033530  104401  033652          TYPE    ,10$            ;<HT>
 9981  033534  017746  000110          MOV     @9$,-(SP)       ;($ERRPC)
 9982  033540  104402                  TYPOC                   ;OCTAL W/ LEADING ZEROS
 9983  033542  104401  033652          TYPE    ,10$            ;<HT>
 9984  033546  012001                  MOV     (R0)+,R1        ;PICKUP "DATA TABLE" PTR
 9985  033550  001407                  BEQ     7$              ;EXIT IF NULL
 9986  033552  013146          6$:     MOV     @(R1)+,-(SP)    ;SAVE ... FOR TYPEOUT
 9987  033554  104402                  TYPOC                   ;TYPE OCTAL, ALL DIGITS
 9988  033556  005711                  TST     (R1)            ;ANOTHER NUMBER ?
 9989  033560  001403                  BEQ     7$              ;NO - EXIT
 9990  033562  104401  033652          TYPE    ,10$            ;TAB BETWEEN ELEMENTS
 9991  033566  000771                  BR      6$              ;LOOP ON DATA TABLE VECTOR
 9992  033570  104401  001353  7$:     TYPE    ,$CRLF          ;CR & LF
 9993  033574  011000                  MOV     (R0),R0         ;GET OPERAND PTR
 9994  033576  001420                  BEQ     12$             ;IF ZERO, SKIP IT
 9995  033600  012001          17$:    MOV     (R0)+,R1        ;POINT TO MESSAGE
 9996  033602  001416                  BEQ     12$             ;DONE IF ZERO
 9997  033604  010137  033612          MOV     R1,14$          ;FOR TYPEOUT
 9998  033610  104401                  TYPE                    ;ASCII TYPER
 9999  033612  000000          14$:    .WORD   0               ;FROM HERE
10000  033614  012001                  MOV     (R0)+,R1        ;POINT TO DATA
10001  033616  001406                  BEQ     15$             ;DONE IF ZERO
10002  033620  010137  033630          MOV     R1,16$          ;FOR TYPEOUT
10003  033624  004537  033674          JSR     R5,TYPMAC       ;FLT PT TYPER
10004  033630  000000          16$:    .WORD   0               ;FROM HERE
10005  033632  000762                  BR      17$             ;NEXT
10006  033634  104401  001353  15$:    TYPE    ,$CRLF          ;A CR/LF
10007  033640  012601          12$:    MOV     (SP)+,R1        ;RESTORE R1
10008  033642  012600                  MOV     (SP)+,R0        ;RESTORE R0
```

```
10009  033644  000207                          RTS     PC              ;RETURN
10010  033646  001362                  8$:     .WORD   $TESTN          ;
10011  033650  001232                  9$:     .WORD   $ERRPC          ;
10012  033652  000011                  10$:    .ASCIZ  <11>            ;<HT>
10013  033654  042524  052123  021455 11$:    .ASCIZ  "TEST-# ERR-PC "
10014  033662  042411  051122  050055
10015  033670  004503     000
10016          033674                          .EVEN
10017
10018
10019                                  ;;*****************************************************************
10020
10021                                  .SBTTL  FLOATING POINT DATA TYPEOUT ROUTINE
10022
10023                                  ;*
10024                                  ;*      THIS ROUTINE TYPES OUT FLOATING POINT DATA (F OR D MODES)
10025                                  ;*      IN THE FOLLOWING FORMAT:
10026                                  ;*
10027                                  ;*      IF SW07=1:      SEEEFF.FFFFFF.FFFFFF.FFFFFF     [D-MODE]
10028                                  ;*                      SEEEFF.FFFFFF                   [F-MODE]
10029                                  ;*
10030                                  ;*      IF SW07=0:      S/EEE/FFF.FFFFFF.FFFFFF.FFFFFF  [D-MODE]
10031                                  ;*                      S/EEE/FFF.FFFFFF                [F-MODE]
10032                                  ;*
10033                                  ;*      MODE IS DETERMINED BY BIT<07> OF "FPLENF":
10034                                  ;*              1=D-MODE (4 WORDS)
10035                                  ;*              0=F-MODE (2 WORDS)
10036                                  ;*
10037                                  ;*      CALLED BY:      JSR     R5,TYPMAC
10038                                  ;*                      ADDR(DATA)
10039                                  ;*
10040
10041  033674  010046          TYPMAC: MOV     R0,-(SP)                ;SAVE R0
10042  033676  012500                  MOV     (R5)+,R0                ;GET ADDR(DATA), BUMP RETURN
10043
10044  033700  104401  003234          TYPE    ,$HT                    ;START WITH A TAB
10045  033704  012046                  MOV     (R0)+,-(SP)             ;WORD-A ONTO STACK FOR TYPEOUT
10046  033706  032777  000200  145340  BIT     #SW07,@SWR              ;CHECK TYPEOUT MODE
10047  033714  001402                  BEQ     10$                     ;BR IF = 0, S/EEE/FFF MODE
10048
10049  033716  104402                  TYPOC                           ;MODE=1, SEEEFF 16. BIT MODE
10050  033720  000425                  BR      20$                     ;
10051
10052                                                                  ;MODE=1, S/EEE/FFF MODE
10053  033722  006116          10$:    ROL     (SP)                    ;GET SIGN
10054  033724  006116                  ROL     (SP)                    ;  IN BIT00
10055  033726  042716  177776          BIC     #^C1,(SP)               ;ONLY SIGN
10056  033732  104403  000401          TYPOS   ,401                    ;1 DIGIT
10057  033736  104401  003240          TYPE    ,$SL                    ;"/"
10058  033742  014046                  MOV     -(R0),-(SP)             ;RETRIEVE WORD-A AGAIN
10059  033744  006316                  ASL     (SP)                    ;EXP IN UPPER BYTE
10060  033746  105016                  CLRB    (SP)                    ;ZAP OTHER BITS
10061  033750  000316                  SWAB    (SP)                    ;NOW IN LOWER BYTE
10062  033752  104403  000403          TYPOS   ,403                    ;EXPONENT, 3 OCTAL
10063  033756  104401  003240          TYPE    ,$SL                    ;"/"
10064  033762  012046                  MOV     (R0)+,-(SP)             ;RETRIEVE WORD-A AGAIN
```

```
10065  033764  042716  177600                  BIC     #^C177,(SP)             ;ZAP SIGN, EXP
10066  033770  104403  000403                  TYPOS   ,403                    ;FRACTION-UPPER, 3 OCTAL
10067
10068  033774  104401  003236          20$:    TYPE    ,$DT                    ;" "
10069  034000  012046                          MOV     (R0)+,-(SP)             ;WORD-B
10070  034002  104402                          TYPOC                           ;6 OCTAL
10071
10072  034004  105737  002624                  TSTB    FPLEN                   ;F(=0) OR D(=1) MODE ?
10073  034010  100010                          BPL     21$                     ;BR IF F-MODE
10074  034012  104401  003236                  TYPE    ,$DT                    ;" "
10075  034016  012046                          MOV     (R0)+,-(SP)             ;WORD-C
10076  034020  104402                          TYPOC                           ;6 OCTAL
10077  034022  104401  003236                  TYPE    ,$DT                    ;" "
10078  034026  011046                          MOV     (R0),-(SP)              ;WORD-D
10079  034030  104402                          TYPOC                           ;6 OCTAL
10080
10081  034032  104401  001353          21$:    TYPE    ,$CRLF                  ;END THE LINE
10082  034036  012600                          MOV     (SP)+,R0                ;RESTORE R0
10083  034040  000205                          RTS     R5                      ;AND RETURN
10084
10085
10086                                  ;;**********************************************************************
10087
10088                                  .SBTTL  TYPE ROUTINE
10089
10090                                  ;;**********************************************************************
10091                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
10092                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
10093                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
10094                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
10095                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
10096                                  ;*
10097                                  ;*CALL:
10098                                  ;*1) USING A TRAP INSTRUCTION
10099                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
10100                                  ;*OR
10101                                  ;*      TYPE
10102                                  ;*      MESADR
10103                                  ;*
10104
10105  034042  105737  001277          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
10106  034046  100002                          BPL     1$              ;;BR IF YES
10107  034050  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
10108  034052  000430                          BR      3$              ;;LEAVE
10109  034054  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
10110  034056  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
10111  034062  122737  000001  001376          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
10112  034070  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
10113  034072  132737  000100  001377          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
10114  034100  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
10115  034102  010037  034112                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
10116  034106  004737  034332                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
10117  034112  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
10118  034114  132737  000040  001377  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
10119  034122  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
10120  034124  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
```

```
10121  034126  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
10122  034130  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
10123  034132  012600            60$:  MOV     (SP)+,R0        ;;RESTORE R0
10124  034134  062716  000002    3$:   ADD     #2,(SP)         ;;ADJUST RETURN PC
10125  034140  000002                  RTI                     ;;RETURN
10126  034142  122716  000011    4$:   CMPB    #HT,(SP)        ;;BRANCH IF <HT>
10127  034146  001430                  BEQ     8$
10128  034150  122716  000200          CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
10129  034154  001006                  BNE     5$
10130  034156  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
10131  034160  104401                  TYPE                    ;;TYPE A CR AND LF
10132  034162  001353                  $CRLF
10133  034164  105037  034320          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
10134  034170  000755                  BR      2$              ;;GET NEXT CHARACTER
10135  034172  004737  034254    5$:   JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
10136  034176  123726  001276    6$:   CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
10137  034202  001350                  BNE     2$              ;;IF NO GO GET NEXT CHAR.
10138  034204  013746  001274          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
10139                                                          ;;AND THE NULL CHAR.
10140  034210  105366  000001    7$:   DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
10141  034214  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
10142  034216  004737  034254          JSR     PC,$TYPEC       ;;GO TYPE A NULL
10143  034222  105337  034320          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
10144  034226  000770                  BR      7$              ;;LOOP
10145
10146                                  ;HORIZONTAL TAB PROCESSOR
10147
10148  034230  112716  000040    8$:   MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
10149  034234  004737  034254    9$:   JSR     PC,$TYPEC       ;;TYPE A SPACE
10150  034240  132737  000007  034320  BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
10151  034246  001372                  BNE     9$              ;;TAB STOP
10152  034250  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
10153  034252  000724                  BR      2$              ;;GET NEXT CHARACTER
10154  034254  105777  145010    $TYPEC: TSTB  @$TPS           ;;WAIT UNTIL PRINTER IS READY
10155  034260  100375                  BPL     $TYPEC
10156  034262  116677  000002  145002  MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
10157  034270  122766  000015  000002  CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
10158  034276  001003                  BNE     1$              ;;BRANCH IF NO
10159  034300  105037  034320          CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
10160  034304  000406                  BR      $TYPEX          ;;EXIT
10161  034306  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
10162  034314  001402                  BEQ     $TYPEX          ;;BRANCH IF YES
10163  034316  105227                  INCB    (PC)+           ;;COUNT THE CHARACTER
10164  034320  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
10165  034322  000207          $TYPEX: RTS     PC
10166
10167
10168
10169
10170                                  ;;******************************************************************
10171
10172                                  .SBTTL  APT COMMUNICATIONS ROUTINE
10173
10174                                  ;;******************************************************************
10175  034324  112737  000001  034570  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
10176  034332  112737  000001  034566  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
```

```
10177  034340  000403                          BR      $ATYC
10178  034342  112737  000001  034570  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
10179  034350                          $ATYC:
10180  034350  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
10181  034352  010145                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
10182  034354  105737  034566                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
10183  034360  001450                          BEQ     5$              ;;IF NOT:  BR
10184  034362  122737  000001  001376          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
10185  034370  001031                          BNE     3$              ;;IF NOT:  BR
10186  034372  132737  000100  001377          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
10187  034400  001425                          BEQ     3$              ;;IF NOT:  BR
10188  034402  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
10189  034406  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
10190  034414  005737  001356          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
10191  034420  001375                          BNE     1$              ;;IF NOT:  WAIT
10192  034422  010037  001372                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
10193  034426  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
10194  034430  001376                          BNE     2$
10195  034432  163700  001372                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
10196  034436  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
10197  034440  010037  001374                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
10198  034444  012737  000004  001356          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
10199  034452  000413                          BR      5$
10200  034454  017637  000004  034500  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
10201  034462  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
10202  034470  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
10203  034474  004737  034042                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
10204  034500  000000                  4$:     .WORD   0
10205  034502                          5$:
10206  034502  105737  034570          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
10207  034506  001416                          BEQ     12$             ;;IF NOT:  BR
10208  034510  005737  001376                  TST     $ENV            ;;RUNNING UNDER APT?
10209  034514  001413                          BEQ     12$             ;;IF NOT:  BR
10210  034516  005737  001356          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
10211  034522  001375                          BNE     11$             ;;IF NOT:  WAIT
10212  034524  017637  000004  001360          MOV     @4(SP),$FATAL   ;;GET ERROR #
10213  034532  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
10214  034540  005237  001356                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
10215  034544  105037  034570          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
10216  034550  105037  034567                  CLRB    $LFLG           ;;CLEAR LOG FLAG
10217  034554  105037  034566                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
10218  034560  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
10219  034562  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
10220  034564  000207                          RTS     PC              ;;RETURN
10221  034566     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
10222  034567     000          $LFLG:  .BYTE   0               ;;LOG FLAG
10223  034570     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
10224          034572                  .EVEN
10225          000200          APTSIZE=200
10226          000001          APTENV=001
10227          000100          APTSPOOL=100
10228          000040          APTCSUP=040
10229
10230
10231
10232                          ;;*****************************************************************
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

                                ;;*****************************************************************
                                ;*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

034572  017646  000000       $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
034576  116637  000001 035015         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
034604  112637  035017               MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
034610  062716  000002               ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
034614  000406                       BR      $TYPON
034616  112737  000001 035015 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
034624  112737  000006 035017         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
034632  112737  000005 035014 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
034640  010346                       MOV     R3,-(SP)        ;;SAVE R3
034642  010446                       MOV     R4,-(SP)        ;;SAVE R4
034644  010546                       MOV     R5,-(SP)        ;;SAVE R5
034646  113704  035017               MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
034652  005404                       NEG     R4
034654  062704  000006               ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
034660  110437  035016               MOVB    R4,$OMODE       ;;SAVE IT FOR USE
034664  113704  035015               MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
034670  016605  000012               MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
034674  005003                       CLR     R3              ;;CLEAR THE OUTPUT WORD
034676  006105                 1$:   ROL     R5              ;;ROTATE MSB INTO "C"
034700  000404                       BR      3$              ;;GO DO MSB
034702  006105                 2$:   ROL     R5              ;;FORM THIS DIGIT
034704  006105                       ROL     R5
034706  006105                       ROL     R5
034710  010503                       MOV     R5,R3
034712  006103                 3$:   ROL     R3              ;;GET LSB OF THIS DIGIT
034714  105337  035016               DECB    $OMODE          ;;TYPE THIS DIGIT?
034720  100016                       BPL     7$              ;;BR IF NO
034722  042703  177770               BIC     #177770,R3      ;;GET RID OF JUNK
034726  001002                       BNE     4$              ;;TEST FOR 0
034730  005704                       TST     R4              ;;SUPPRESS THIS 0?
```

```
10289  034732  001403                          BEQ     5$              ;;BR IF YES
10290  034734  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
10291  034736  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
10292  034742  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
10293  034746  110337  035012                  MOVB    R3,8$           ;;SAVE FOR TYPING
10294  034752  104401  035012                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
10295  034756  105337  035014          7$:     DECB    $OCNT           ;;COUNT BY 1
10296  034762  003347                          BGT     2$              ;;BR IF MORE TO DO
10297  034764  002402                          BLT     6$              ;;BR IF DONE
10298  034766  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
10299  034770  000744                          BR      2$              ;;GO DO THE LAST DIGIT
10300  034772  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
10301  034774  012604                          MOV     (SP)+,R4        ;;RESTORE R4
10302  034776  012603                          MOV     (SP)+,R3        ;;RESTORE R3
10303  035000  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
10304  035006  012616                          MOV     (SP)+,(SP)
10305  035010  000002                          RTI                     ;;RETURN
10306  035012     000          8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
10307  035013     000                  .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
10308  035014     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
10309  035015     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
10310  035016  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
10311
10312
10313                          ;;**************************************************************
```

```
                                .SBTTL  "TRAP" INSTRUCTION DECODER


                                ;;*****************************************************************
                                ;*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 AT THE DESIRED PRIORITY LEVEL.

035020  024646          $TRAP:  CMP     -(SP),-(SP)             ;;RESERVE TWO WORDS ON STACK
035022  010046                  MOV     R0,-(SP)                ;;SAVE R0
035024  016600  000006          MOV     6(SP),R0                ;;COPY TRAP ADDRESS
035030  005740                  TST     -(R0)                   ;;BACKUP BY TWO
035032  111000                  MOVB    (R0),R0                 ;;GET RITE BYTE OF TRAP
035034  006300                  ASL     R0                      ;;POSITION FOR INDEXING
035036  006300                  ASL     R0                      ;;
035040  016066  035060  000002  MOV     $TRPAD+0(R0),2(SP)      ;;INDEX TO TABLE, NEW PC
035046  016066  035062  000004  MOV     $TRPAD+2(R0),4(SP)      ;;AND PS
035054  012600                  MOV     (SP)+,R0                ;;RESTORE R0
035056  000002                  RTI                             ;;AND GO TO THE ROUTINE


                                .SBTTL  TRAP TABLE

                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.

                                ;ROUTINE/PRIO
                                ;-------/----
035060  000000  000000  $TRPAD: .WORD   0,0
035064  034042  000340  $TYPE,PR7         ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
035070  034616  000340  $TYPOC,PR7        ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
035074  034572  000340  $TYPOS,PR7        ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
035100  034632  000340  $TYPON,PR7        ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
                                :NOTE:  THE FOLLOWING TRAP ENTRIES ARE ADDITIONALLY DEFINED USER ENTRIES
035104  035426  000340  $CNDES,PR7       ;;CALL=CNDSES   TRAP+5(104405)  LOAD $ESCAPE, IF SW6=1
035110  035452  000340  $ERPNT,PR7       ;;CALL=ERRPNT   TRAP+6(104406)  SETUP $LPERR TO THIS CALL
035114  035460  000340  $LPPNT,PR7       ;;CALL=LOOPNT   TRAP+7(104407)  SETUP $LPADR TO THIS CALL
035120  035246  000340  $SETDW,PR7       ;;CALL=SETDW    TRAP+10(104410) SETUP LINE-CLOCK - PROC HUNG ENABLE
035124  035234  000340  $CLRDW,PR7       ;;CALL=CLRDW    TRAP+11(104411) CLEAR LINE-CLOCK - PROC HUNG ENABLE
035130  035404  000340  $SETFP,PR7       ;;CALL=SETFP    TRAP+12(104412) SETUP FP TRAP ESCAPE
035134  035372  000340  $CLRFP,PR7       ;;CALL=CLRFP    TRAP+13(104413) CLEAR FP TRAP ESCAPE
035140  035330  000340  $SETUB,PR7       ;;CALL=SETUB    TRAP+14(104414) SETUP BM UBRK ESCAPE ENABLE
035144  035304  000340  $CLRUB,PR7       ;;CALL=CLRUB    TRAP+15(104415) CLEAR BM UBRK ESCAPE ENABLE
035150  031406  000240  $ZPHFP,PR5       ;;CALL=ZAPHFP   TRAP+16(104416) INIT HFP-FPS-FEC-FEA  HFP ENAB
035154  031400  000240  $ZPWFP,PR5       ;;CALL=ZAPWFP   TRAP+17(104417) INIT HFP-FPS-FEC-FEA, HFP DISAB
035160  031230  000240  $DUBL,PR5        ;;CALL=DBLDAT   TRAP+20(104420) 4 WORDS OF RANDOM DATA
035164  031220  000240  $SNGL,PR5        ;;CALL=SGLDAT   TRAP+21(104421) 2 WORDS OF RANDOM DATA
035170  031150  000240  $RNFPS,PR5       ;;CALL=RANFPS   TRAP+22(104422) RANDOM FPS IN "$FPS"
035174  031564  000240  $FIXFR,PR5       ;;CALL=FIXFRA   TRAP+23(104423) FIX FRACTION, USING EXP=EADJ
035200  031476  000240  $EADJ,PR5        ;;CALL=EADJ     TRAP+24(104424) CALCULATE NORMK-EADJ
035204  032222  000240  $CMP4W,PR5       ;;CALL=CMP64M   TRAP+25(104425) 64. BIT COMPARE EQ-NE
035210  032214  000240  $CMP2W,PR5       ;;CALL=CMP32M   TRAP+26(104426) 32. BIT COMPARE EQ-NE
035214  032204  000240  $CMPWD,PR5       ;;CALL=CMPXXM   TRAP+27(104427) 32.-64. BIT COMPARE EQ-NE
035220  031660  000240  $AS64I,PR5       ;;CALL=ASH64I   TRAP+30(104430) 64. BIT IMMEDIATE "ASHC"
035224  031636  000240  $AS64M,PR5       ;;CALL=ASH64M   TRAP+31(104431) 64. BIT MEMORY "ASHC"
```

```
10370  035230  032112  000240          $SB64M,PR5      ;;CALL=SUB64M   TRAP+32(104432) 64. BIT MEMORY "SUB"
10371
10372
10373                                  ;;*****************************************************************
10374
10375                                  .SBTTL  ...SETUP LINE-CLOCK/PROCESSOR HUNG ESCAPE (SETDW, CLRDW)
10376
10377                                  ;*      THIS ROUTINE SETS UP "DWESCP" WITH THE CONTENTS OF THE
10378                                  ;*      FOLLOWING WORD (AN ADDRESS), AND ALSO SETS THE "DWFLAG"
10379                                  ;*      TO (377), INDICATING ESCAPE IS ENABLED, IF AND WHEN THE
10380                                  ;*      RETURN ADDRESS FROM THE LINE CLOCK INTERRUPT SERVICE
10381                                  ;*      ROUTINE IS SEEN TO BE THE SAME VALUE FOR (DWICNT)
10382                                  ;*      CONSECUTIVE CLOCK TICKS.
10383                                  ;*
10384                                  ;*      CALLED BY:      SETDW           ;ENTER ROUTINE
10385                                  ;*                      ESCAPE          ;ESCAPE ADDRESS FOR TIMEOUT
10386                                  ;*              OR
10387                                  ;*                      CLRDW           ;CLEAR PREV ENABLE
10388                                  ;*
10389
10390  035234  005037  002634          $CLRDW: CLR     DWESCP                  ;CLEAR ESCAPE ADDR
10391  035240  105037  002644                  CLRB    DWFLAG                  ;CLEAR ENABLE FLAG
10392  035244  000410                          BR      $DW                     ;
10393  035246  017637  000000  002634  $SETDW: MOV     @(SP),DWESCP            ;GET ESCAPE ADDRESS
10394  035254  112737  000377  002644          MOVB    #377,DWFLAG             ;SETUP FLAG FOR ENABLE
10395  035262  062716  000002                  ADD     #2,(SP)                 ;FIX RETURN ADDRESS
10396  035266  013737  003000  002636  $DW:    MOV     DWICNT,DWCNTR           ;ALWAYS RESET COUNTER FOR MATCHES
10397  035274  013737  002640  002642          MOV     DWLOOP,DWOLOP           ;RESET LOOP COUNT
10398  035302  000002                          RTI                             ;AND RETURN
10399
10400                                  ;;*****************************************************************
10401
10402                                  .SBTTL  ...SETUP PROCESSOR MICROBREAK ESCAPE (SETUB, CLRUB)
10403
10404                                  ;*      THIS ROUTINE SETS UP "UBESCP" WITH THE CONTENTS OF THE
10405                                  ;*      FOLLOWING WORD (AN ADDRESS), AND ALSO SETS THE "UBTPOK"
10406                                  ;*      TO (377), INDICATING ESCAPE IS ENABLED, IF AND WHEN THE
10407                                  ;*      PROCESSOR MICROBREAK OCCURS.  THE COUNT IN "UBCNTR" IS
10408                                  ;*      BUMPED EACH TIME ALSO.  "FLAG<8>" IS ALSO SETUP
10409                                  ;*      TO ENABLE THE NEXT BREAK.  RETURN IS MADE TO THE ADDRESS
10410                                  ;*      IN "UBESCP" IF IT IS NONZERO.
10411                                  ;*
10412                                  ;*      CALLED BY:      SETUB           ;ENTER ROUTINE
10413                                  ;*                      ESCAPE          ;ESCAPE ADDRESS FOR BREAK
10414                                  ;*              OR
10415                                  ;*                      CLRUB           ;CLEAR PREV ENABLE
10416                                  ;*
10417
10418  035304  005037  002626          $CLRUB: CLR     UBESCP                  ;CLEAR ESCAPE ADDR
10419  035310  105037  002625                  CLRB    UBTPOK                  ;CLEAR ENABLE FLAG
10420  035314  010046                          MOV     R0,-(SP)                ;SAVE
10421  035316  076600  000144                  MFD     PFLAG                   ;GET BM FLAGS IN R0
10422  035322  042700  100000                  BIC     #BIT15,R0               ;ZAP UBRK ENABLE
10423  035326  000415                          BR      $UB                     ;
10424  035330  017637  000000  002626  $SETUB: MOV     @(SP),UBESCP            ;GET ESCAPE ADDRESS
10425  035336  112737  000377  002625          MOVB    #377,UBTPOK             ;SETUP FLAG FOR ENABLE
```

```
10426  035344  062716  000002          ADD     #2,(SP)                 ;FIX RETURN ADDRESS
10427  035350  010046                  MOV     R0,-(SP)                ;SAVE
10428  035352  076600  000144          MFD     @RFLAG                  ;GET FLAGS IN R0
10429  035356  052700  100000          BIS     #BIT15,R0               ;ENABLE BM UBRK FLAG(8)
10430  035362  076600  000344  $UB:    MFD     @WFLAG                  ;RESET FLAGS
10431  035366  012600                  MOV     (SP)+,R0                ;RESTORE R0
10432  035370  000002                  RTI                             ;AND RETURN
10433
10434                                  ;;*****************************************************************
10435
10436                                  .SBTTL  ...SETUP FLOATING POINT TRAP ESCAPE (SETFP, CLRFP)
10437
10438                                  ;*      THIS ROUTINE SETS UP "FPESCP" WITH THE CONTENTS OF THE
10439                                  ;*      FOLLOWING WORD (AN ADDRESS), AND ALSO SETS THE "FPTPOK"
10440                                  ;*      TO (377), INDICATING ESCAPE IS ENABLED, IF AND WHEN THE
10441                                  ;*      FLOATING POINT TRAP TO (244) OCCURS.  THE SERVICE ROUTINE
10442                                  ;*      CLEARS "FPTPOK" AFTER THE TRAP, SO MORE THAN ONE
10443                                  ;*      IN A ROW WILL GENERATE AN ERROR.
10444                                  ;*
10445                                  ;*      CALLED BY:      SETFP           ;ENTER ROUTINE
10446                                  ;*                      ESCAPE          ;ESCAPE ADDRESS
10447                                  ;*              OR
10448                                  ;*                      CLRFP           ;CLEAR PREV ENABLE
10449                                  ;*
10450
10451  035372  005037  002620  $CLRFP: CLR     FPESCP                  ;CLEAR ESCAPE ADDR
10452  035376  105037  002622          CLRB    FPTPOK                  ;CLEAR ENABLE FLAG
10453  035402  000410                  BR      $FP                     ;
10454  035404  017637  000000  002620  $SETFP: MOV     @(SP),FPESCP            ;GET ESCAPE ADDRESS
10455  035412  112737  000377  002622          MOVB    #377,FPTPOK             ;SETUP FLAG FOR ENABLE
10456  035420  062716  000002                  ADD     #2,(SP)                 ;FIX RETURN ADDRESS
10457  035424  000002          $FP:    RTI                             ;AND RETURN
10458
10459                                  ,;*****************************************************************
10460
10461                                  .SBTTL  ...CONDITIONALLY LOAD $ESCAPE (CND$ES)
10462
10463                                  ;*      THIS ROUTINE LOADS THE NEXT WORD AFTER ITS CALL INTO
10464                                  ;*      THE SYSMAC $ESCAPE WORD, WHICH IS USED AS THE EXIT
10465                                  ;*      ADDRESS WHEN "ERROR X." IS CALLED AND $ESCAPE IS NONZERO.
10466                                  ;*      $ESCAPE IS ZEROED IN THE "SCOPE" ROUTINE.
10467                                  ;*
10468                                  ;*      NOTE:  THE LOADING ONLY TAKES PLACE IF SW6=1.
10469                                  ;*
10470                                  ;*      CALLED BY:      CND$ES          ;CONDITIONAL LOAD $ESCAPE
10471                                  ;*                      ESCAPE          ;"ESCAPE"=ADDR TO PUT IN $ESCAPE
10472                                  ;*
10473
10474  035426  032777  000100  143620  $CNDES: BIT     #SW6,@SWR               ;BIT SET ?
10475  035434  001403                  BEQ     1$                      ;BR IF NOT
10476  035436  017637  000000  001344          MOV     @(SP),$ESCAPE           ;LOAD FROM NEXT WORD
10477  035444  062716  000002          1$:     ADD     #2,(SP)                 ;FIX RETURN ADDRESS
10478  035450  000002                  RTI                             ;AND RETURN
10479
10480                                  ;;*****************************************************************
10481
```

```
10482                                   .SBTTL  ...SETUP ERROR LOOP ($LPERR) POINT (ERRPNT)
10483
10484                                   ;*      SETS UP $LPERR LOCATION TO AFTER THE CALL
10485                                   ;*
10486                                   ;*      CALLED BY:      ERRPNT
10487
10488  035452  011637  001222          $ERPNT: MOV     (SP),$LPERR             ;POINT $LPERR TO RETURN LOCATION
10489  035456  000002                          RTI                             ;AND RETURN
10490
10491                                   ;;**********************************************************************
10492
10493                                   .SBTTL  ...SETUP SCOPE LOOP ($LPADR) POINT (LOOPNT)
10494
10495                                   ;*      SETS UP $LPADR LOCATION TO AFTER THE CALL
10496                                   ;*
10497                                   ;*      CALLED BY:      LOOPNT
10498
10499  035460  011637  001220          $LPPNT: MOV     (SP),$LPADR             ;POINT $LPADR TO RETURN LOCATION
10500  035464  000002                          RTI                             ;AND RETURN
10501
10502
10503                                   ;;**********************************************************************
10504                                   ;;**********************************************************************
10505
10506                                   .SBTTL  POWER DOWN AND UP ROUTINES
10507
10508                                   ;;**********************************************************************
10509                                   ;POWER DOWN ROUTINE
10510  035466  012737  035640  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC        ;;SET FOR FAST UP
10511  035474  012737  000340  000026          MOV     #340,@#PWRVEC+2         ;;PRIO:7
10512  035502  010046                         MOV     R0,-(SP)                ;;PUSH R0 ON STACK
10513  035504  010146                         MOV     R1,-(SP)                ;;PUSH R1 ON STACK
10514  035506  010246                         MOV     R2,-(SP)                ;;PUSH R2 ON STACK
10515  035510  010346                         MOV     R3,-(SP)                ;;PUSH R3 ON STACK
10516  035512  010446                         MOV     R4,-(SP)                ;;PUSH R4 ON STACK
10517  035514  010546                         MOV     R5,-(SP)                ;;PUSH R5 ON STACK
10518  035516  017746  143532                 MOV     @SWR,-(SP)              ;;PUSH @SWR ON STACK
10519  035522  010637  035644                 MOV     SP,$SAVR6               ;;SAVE SP
10520  035526  012737  035540  000024         MOV     #$PWRUP,@#PWRVEC        ;;SET UP VECTOR
10521  035534  000000                         HALT
10522  035536  000776                         BR      .-2                     ;;HANG UP
10523
10524                                   ;;**********************************************************************
10525                                   ;POWER UP ROUTINE
10526  035540  012737  035640  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC        ;;SET FOR FAST DOWN
10527  035546  013706  035644                 MOV     $SAVR6,SP               ;;GET SP
10528  035552  005037  035644                 CLR     $SAVR6                  ;;WAIT LOOP FOR THE TTY
10529  035556  005237  035644          1$:    INC     $SAVR6                  ;;WAIT FOR THE INC
10530  035562  001375                         BNE     1$                      ;;OF  WORD
10531  035564  011600                         MOV     (SP),R0                 ;GET SAVED SWR OFF STACK
10532  035566  076600  000226                 MED     ,226                    ;RESTORE SWR CONTENTS (CNSL.SW IN ASPHI[06])
10533  035572  012677  143456                 MOV     (SP)+,@SWR              ;;POP STACK INTO @SWR
10534  035576  012605                         MOV     (SP)+,R5                ;;POP STACK INTO R5
10535  035600  012604                         MOV     (SP)+,R4                ;;POP STACK INTO R4
10536  035602  012603                         MOV     (SP)+,R3                ;;POP STACK INTO R3
10537  035604  012602                         MOV     (SP)+,R2                ;;POP STACK INTO R2
```

```
 10538  035606  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 10539  035610  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 10540  035612  012737  035466  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
 10541  035620  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
 10542  035626  104401                          TYPE                    ;REPORT THE POWER FAILURE
 10543  035630  035646                $PWRMG:   .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
 10544  035632  012716                          MOV     (PC)+,(SP)      ;;RESTART AT RESTRT
 10545  035634  003664                $PWRAD:   .WORD   RESTRT          ;;RESTART ADDRESS
 10546  035636  000002                          RTI
 10547  035640  000000                $ILLUP:   HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
 10548  035642  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
 10549  035644  000000                $SAVR6:   0                       ;;PUT THE SP HERE
 10550  035646  005015  047520  042527  $POWER:  .ASCIZ  <15><12>"POWER"
 10551  035654  000122
 10552                                          .EVEN
 10553
 10554                                ;;*****************************************************************
```

```
10555                                          .SBTTL  ERR MESSAGES, DATA HEADERS, DATA VECTORS, ETC
10556
10557                                                  ;ERR MESSAGES HERE
10558  035656                                  EMA:
10559  035656  047125  054105  023520          EMB:    .ASCIZ  "UNEXP'D FPP TRAP TO (244)"
10560  035664  020104  050106  020120
10561  035672  051124  050101  052040
10562  035700  020117  031050  032064
10563  035706  000051
10564  035710  047125  054105  023520          EMC:    .ASCIZ  "UNEXP'D TRAP TO (4)"
10565  035716  020104  051124  050101
10566  035724  052040  020117  032050
10567  035732  000051
10568  035734  047125  054105  023520          EMD:    .ASCIZ  "UNEXP'D TRAP TO (10)"
10569  035742  020104  051124  050101
10570  035750  052040  020117  030450
10571  035756  024460     000
10572  035761     125  042516  050130          EME:    .ASCIZ  "UNEXP'D TRAP TO (114)"
10573  035766  042047  052040  040522
10574  035774  020120  047524  024040
10575  036002  030461  024464     000
10576  036007     115  044501  052116          EME1:   .ASCIZ  "MAINT INSTR ALTERED AC0"
10577  036014  044440  051516  051124
10578  036022  040440  052114  051105
10579  036030  042105  040440  030103
10580  036036     000
10581  036037     115  044501  052116          EME2:   .ASCIZ  "MAINT INSTR ALTERED FPS"
10582  036044  044440  051516  051124
10583  036052  040440  052114  051105
10584  036060  042105  043040  051520
10585  036066     000
10586  036067     115  050120  020054          EME3:   .ASCIZ  "MPP, AC2 MNET(S+C) ERR"
10587  036074  041501  020062  047115
10588  036102  052105  051450  041453
10589  036110  020051  051105  000122
10590  036116  047115  026123  040440          EME4:   .ASCIZ  "MNS, AC1 NORM ERR"
10591  036124  030503  047040  051117
10592  036132  020115  051105  000122
10593  036140  040515  026123  040440          EME5:   .ASCIZ  "MAS, AC1 PRE-SHFT ERR"
10594  036146  030503  050040  042522
10595  036154  051455  043110  020124
10596  036162  051105  000122
10597  036166  040515  026123  040440          EME6:   .ASCIZ  "MAS, AC2 CNTR INCR ERR"
10598  036174  03_103  041440  052116
10599  036202  020122  047111  051103
10600  036210  042440  051122     000
10601  036215     115  046125  042516          EMF:    .ASCIZ  "MULNET MULT-ROM CONTENTS ERR"
10602  036222  020124  052515  052114
10603  036230  051055  046517  041440
10604  036236  047117  042524  052116
10605  036244  020123  051105  000122
10606  036252  052515  047114  052105          EMG:    .ASCIZ  "MULNET CNTR-ROM CONTENTS ERR"
10607  036260  041440  052116  020522
10608  036266  047522  020115  047503
10609  036274  052116  047105  051524
10610  036302  042440  051122     000
```

```
10611  036307     115  046125  042516  EMH:    .ASCIZ  "MULNET MULT-ROM ERR"
10612  036314  020124  052515  052114
10613  036322  051055  046517  042440
10614  036330  051122     000
10615  036333     110  050106  044457  EMI:    .ASCIZ  "HFP/IFORK/-[(ADD+SUB)*MD]; BAD IFORK DECODE"
10616  036340  047506  045522  026457
10617  036346  024133  042101  025504
10618  036354  052523  024502  046452
10619  036362  056460  020073  040502
10620  036370  020104  043111  051117
10621  036376  020113  042504  047503
10622  036404  042504     000
10623  036407     110  050106  044457  EMJ:    .ASCIZ  "HFP/IFORK/-[(ADD+SUB)*MD]; UNEXP'D FEC/FEA"
10624  036414  047506  045522  026457
10625  036422  024133  042101  025504
10626  036430  052523  024502  046452
10627  036436  056460  020073  047125
10628  036444  054105  023520  020104
10629  036452  042506  027503  042506
10630  036460  000101
10631  036462  051506  040520  020104  EMK1:   .ASCIZ  "FSPAD ADDRS ERR, R[DF]"
10632  036470  042101  051104  020123
10633  036476  051105  026122  051040
10634  036504  042133  056506     000
10635  036511     106  050123  042101  EMK2:   .ASCIZ  "FSPAD ADDRS ERR, R[SF]"
10636  036516  040440  042104  051522
10637  036524  042440  051122  020054
10638  036532  055522  043123  000135
10639  036540  043110  020120  051120  EML:    .ASCIZ  "HFP PREPD/1/2, UBRK, SRVC, */+ ENABL ERR"
10640  036546  050105  027460  027461
10641  036554  026062  052440  051102
10642  036562  026113  051440  053122
10643  036570  026103  025040  025457
10644  036576  042440  040516  046102
10645  036604  042440  051122     000
10646  036611     120  047522  020103  EMM:    .ASCIZ  "PROC HUNG: LINE CLOCK TIMEOUT"
10647  036616  052510  043516  020072
10648  036624  044514  042516  041440
10649  036632  047514  045503  052040
10650  036640  046511  047505  052125
10651  036646     000
10652  036647     102  020115  044127  EMN:    .ASCIZ  "BM WHAMI/FLAGS INIT ERR"
10653  036654  046501  027511  046106
10654  036662  043501  020123  047111
10655  036670  052111  042440  051122
10656  036676     000
10657  036677     102  020115  046106  EMO:    .ASCIZ  "BM FLAGS R/W ERR"
10658  036704  043501  020123  027522
10659  036712  020127  051105  000122
10660  036720  046502  044440  051516  EMP:    .ASCIZ  "BM INSTR1/FP DECODE ERR; FLAGS OK"
10661  036726  051124  027461  050105
10662  036734  042040  041505  042117
10663  036742  020105  051105  035522
10664  036750  043040  040514  051507
10665  036756  047440  000113
10666  036762  046502  044440  051516  EMQ:    .ASCIZ  "BM INSTR1/FP DECODE OR FLAGS ERR"
```

```
10667  036770  051124  027461  050106
10668  036776  042040  041505  042117
10669  037004  020105  051117  043040
10670  037012  040514  051507  042440
10671  037020  051122     000
10672  037023     102  020115  046106  EMR:    .ASCIZ  "BM FLAG4=1 AFTER CSP FP CNST RESTORE"
10673  037030  043501  036464  020061
10674  037036  043101  042524  020122
10675  037044  051503  020120  050106
10676  037052  041440  051516  020124
10677  037060  042522  052123  051117
10678  037066  000105
10679  037070  046502  041040  042101  EMS:    .ASCIZ  "BM BAD FP-CNST IN CSP"
10680  037076  043040  026520  047103
10681  037104  052123  044440  020116
10682  037112  051503  000120
10683  037116  043110  020120  051123  EMAA:   .ASCIZ  "HFP SRVC GRANT ERR"
10684  037124  041526  043440  040522
10685  037132  052116  042440  051122
10686  037140     000
10687  037141     102  042101  052440  EMAB:   .ASCIZ  "BAD UBRK CODE FROM HFP [CODE#07/FEC#16]"
10688  037146  051102  020113  047503
10689  037154  042504  043040  047522
10690  037162  020115  043110  020120
10691  037170  041533  042117  021505
10692  037176  033460  043057  041505
10693  037204  030443  056466     000
10694  037211     102  020115  052510  EMAC:   .ASCIZ  "BM HUNG DURING HFP FLPGO/FPACK SEQ"
10695  037216  043516  042040  051125
10696  037224  047111  020107  043110
10697  037232  020120  046106  043520
10698  037240  027517  050106  041501
10699  037246  020113  042523  000121
10700  037254  052515  047114  052105  EMAD:   .ASCIZ  "MULNET-DATA STUCK/H OR REG. LOAD ERR; 0*0=0"
10701  037262  042055  052101  020101
10702  037270  052123  041525  027513
10703  037276  020110  051117  051040
10704  037304  043505  020056  047514
10705  037312  042101  042440  051122
10706  037320  020073  025060  036460
10707  037326  000060
10708  037330  052515  047114  052105  EMAE:   .ASCIZ  "MULNET-RIPPLE 0/1 ERR"
10709  037336  051055  050111  046120
10710  037344  020105  027460  020061
10711  037352  051105  000122
10712  037356  052515  047114  052105  EMAF:   .ASCIZ  "MULNET-RIPPLE 0/1 ERR; S#S+C"
10713  037364  051055  050111  046120
10714  037372  020105  027460  020061
10715  037400  051105  035522  051440
10716  037406  051443  041453     000
10717  037413     115  046125  042516  EMAG:   .ASCIZ  "MULNET MULD RIPPLE-A-1 THRU MIER ERR"
10718  037420  020124  052515  042114
10719  037426  051040  050111  046120
10720  037434  026505  026501  020061
10721  037442  044124  052522  046440
10722  037450  042511  020122  051105
```

```
10723  037456  000122
10724  037460  052515  047114  052105  EMAH:   .ASCIZ  "MULNET MULD RIPPLE-A-1 THRU MAND ERR"
10725  037466  046440  046125  020104
10726  037474  044522  050120  042514
10727  037502  040455  030455  052040
10728  037510  051110  020125  040515
10729  037516  042116  042440  051122
10730  037524     000
10731  037525     110  050106  020120  EMAI:   .ASCIZ  "HFPP STATUS INSTR EXEC ALTERED FPS"
10732  037532  052123  052101  051525
10733  037540  044440  051516  051124
10734  037546  042440  042530  020103
10735  037554  046101  042524  042522
10736  037562  020104  050106  000123
10737  037570  047516  046522  026513  EMAJ:   .ASCIZ  "NORMK-EADJ/SHFTR ERR"
10738  037576  040505  045104  051457
10739  037604  043110  051124  042440
10740  037612  051122     000
10741  037615     123  043110  051124  EMAK:   .ASCIZ  "SHFTR L(2+4) OF ZERO ERR"
10742  037622  046040  031050  032053
10743  037630  020051  043117  055040
10744  037636  051105  020117  051105
10745  037644  000122
10746  037646  044123  052106  020122  EMAL:   .ASCIZ  "SHFTR R(11.-1) OF ZERO ERR"
10747  037654  024122  030461  026456
10748  037662  024461  047440  020106
10749  037670  042532  047522  042440
10750  037676  051122     000
10751  037701     123  043110  051124  EMAM:   .ASCIZ  "SHFTR, MAS-RITE RIPPLE-A-1 ERR"
10752  037706  020054  040515  026523
10753  037714  044522  042524  051040
10754  037722  050111  046120  026505
10755  037730  026501  020061  051105
10756  037736  000122
10757  037740  044123  052106  026122  EMAN:   .ASCIZ  "SHFTR, MNS-LEFT/RITE RIPPLE-A-1 ERR"
10758  037746  046440  051516  046055
10759  037754  043105  027524  044522
10760  037762  042524  051040  050111
10761  037770  046120  026505  026501
10762  037776  020061  051105  000122
10763  040004  042114  027506  052123  EMAO:   .ASCIZ  "LDF/STF FRAC DATAPATH ERR"
10764  040012  020106  051106  041501
10765  040020  042040  052101  050101
10766  040026  052101  020110  051105
10767  040034  000122
10768  040036  042114  027504  052123  EMAP:   .ASCIZ  "LDD/STD FRAC DATAPATH ERR"
10769  040044  020104  051106  041501
10770  040052  042040  052101  050101
10771  040060  052101  020110  051105
10772  040066  000122
10773  040070  051506  040520  020104  EMAQ:   .ASCIZ  "FSPAD DATA ERR"
10774  040076  040504  040524  042440
10775  040104  051122     000
10776  040107     106  050123  042101  EMAR:   .ASCIZ  "FSPAD FP-INSTR MODIFIED SRC-ACC ERR"
10777  040114  043040  026520  047111
10778  040122  052123  020122  047515
```

```
10779  040130  044504  044506  042105
10780  040136  051440  041522  040455
10781  040144  041503  042440  051122
10782  040152     000
10783  040153     106  050123  042101  EMAS:   .ASCIZ  "FSPAD-SECT[CD] ADDRS/WRITE-ENABL ERR"
10784  040160  051455  041505  055524
10785  040166  042103  020135  042101
10786  040174  051104  027523  051127
10787  040202  052111  026505  047105
10788  040210  041101  020114  051105
10789  040216  000122
10790  040220  051505  040520  027104  EMAT:   .ASCIZ  "ESPAD.B LDX/STX DATAPATH ERR"
10791  040226  020102  042114  027530
10792  040234  052123  020130  040504
10793  040242  040524  040520  044124
10794  040250  042440  051122     000
10795  040255     105  050123  042101  EMAU:   .ASCIZ  "ESPAD.A LDX/STX DATAPATH ERR"
10796  040262  040456  046040  054104
10797  040270  051457  054124  042040
10798  040276  052101  050101  052101
10799  040304  020110  051105  000122
10800  040312  040506  052514  040440  EMAV:   .ASCIZ  "FALU ADDD/CARRY RESULT ERR"
10801  040320  042104  027504  040503
10802  040326  051122  020131  042522
10803  040334  052523  052114  042440
10804  040342  051122     000
10805  040345     106  050123  042101  EMAW:   .ASCIZ  "FSPAD.IN.MUX INBUF-PORT ERR"
10806  040352  044456  027116  052515
10807  040360  020130  047111  052502
10808  040366  026506  047520  052122
10809  040374  042440  051122     000
10810  040401     106  051111  020102  EMAX:   .ASCIZ  "FIRB IMMED-H MODE DECODE ERR"
10811  040406  046511  042515  026504
10812  040414  020110  047515  042504
10813  040422  042040  041505  042117
10814  040430  020105  051105  000122
10815  040436  043111  051117  027513  EMAY:   .ASCIZ  "IFORK/(ADD+SUB)*MO EXECUTE ERR"
10816  040444  040450  042104  051453
10817  040452  041125  025051  030115
10818  040460  042440  042530  052503
10819  040466  042524  042440  051122
10820  040474     000
10821  040475     105  050130  052116  EMBA:   .ASCIZ  "EXPNT EA=0, EB=0 DATAPATH ERR"
10822  040502  042440  036501  026060
10823  040510  042440  036502  020060
10824  040516  040504  040524  040520
10825  040524  044124  042440  051122
10826  040532     000
10827  040533     105  050130  052116  EMBB:   .ASCIZ  "EXPNT EA=0+EB=0 DATAPATH ERR"
10828  040540  042440  036501  025460
10829  040546  041105  030075  042040
10830  040554  052101  050101  052101
10831  040562  020110  051105  000122
10832  040570  054105  047120  020124  EMBC:   .ASCIZ  "EXPNT ER=0 DATAPATH ERR"
10833  040576  051105  030075  042040
10834  040604  052101  050101  052101
```

```
10835  040612  020110  051105  000122
10836  040620  054105  047120  020124  EMBD:   .ASCIZ  "EXPNT EALU EA-PLUS-EB RESULT ERR"
10837  040626  040505  052514  042440
10838  040634  026501  046120  051525
10839  040642  042455  020102  042522
10840  040650  052523  052114  042440
10841  040656  051122     000
10842  040661     105  050130  052116  EMBK:   .ASCIZ  "EXPNT EALU EA-PLUS-EB MULF/SEQ ERR"
10843  040666  042440  046101  020125
10844  040674  040505  050055  052514
10845  040702  026523  041105  046440
10846  040710  046125  027506  042523
10847  040716  020121  051105  000122
10848  040724  054105  047120  020124  EMBE:   .ASCIZ  "EXPNT CNTR/PRE-SHFT-QUOT-ROM ERR"
10849  040732  047103  051124  050057
10850  040740  042522  051455  043110
10851  040746  026524  052521  052117
10852  040754  051055  046517  042440
10853  040762  051122     000
10854  040765     111  047506  045522  EMBF:   .ASCIZ  "IFORK/(ADD+SUB)*MO SUMPATH/MO*R6 ERR"
10855  040772  024057  042101  025504
10856  041000  052523  024502  046452
10857  041006  020060  052523  050115
10858  041014  052101  027510  030115
10859  041022  051052  020066  051105
10860  041030  000122
10861  041032  043111  051117  027513  EMBG:   .ASCIZ  "IFORK/(ADD+SUB)*MO [EA+EB]=0/MO*R6 ERR"
10862  041040  040450  042104  051453
10863  041046  041125  025051  030115
10864  041054  055440  040505  042453
10865  041062  056502  030075  046457
10866  041070  025060  033122  042440
10867  041076  051122     000
10868  041101     111  047506  045522  EMBH:   .ASCIZ  "IFORK/(ADD+SUB)*MO EXPNT.RANGE.CODE.ROM ERR"
10869  041106  024057  042101  025504
10870  041114  052523  024502  046452
10871  041122  020060  054105  047120
10872  041130  027124  040522  043516
10873  041136  027105  047503  042504
10874  041144  051056  046517  042440
10875  041152  051122     000
10876  041155     104  053111  042111  EMBI:   .ASCIZ  "DIVIDE INBUF/AR-SHIFT FSPAD-SELECT ERR"
10877  041162  020105  047111  052502
10878  041170  027506  051101  051455
10879  041176  044510  052106  043040
10880  041204  050123  042101  051455
10881  041212  046105  041505  020124
10882  041220  051105  000122
10883  041224  042114  050103  044450  EMBJ:   .ASCIZ  "LDCP(I->F  FPINMUX/DOUT CONVERT ERR"
10884  041232  037055  024506  043040
10885  041240  044520  046516  054125
10886  041246  042057  052517  020124
10887  041254  047503  053116  051105
10888  041262  020124  051105  000122
10889  041270  042506  050130  043057  EMBL:   .ASCIZ  "FEXP/FALU FPS F-D &/+ R-T MODE ERR"
10890  041276  046101  020125  050106
```

```
10891  041304  020123  026506  020104
10892  041312  027446  020053  026522
10893  041320  020124  047515  042504
10894  041326  042440  051122     000
10895  041333     106  040522  052103  EMBM:   .ASCIZ  "FRACTION/CLRD-EXEC/FP.EMIT.F DATA ERR"
10896  041340  047511  027516  046103
10897  041346  042122  042455  042530
10898  041354  027503  050106  042456
10899  041362  044515  027124  020106
10900  041370  040504  040524  042440
10901  041376  051122     000
10902  041401     106  044520  044516  EMBN:   .ASCIZ  "FPINIT/CLRD/LDEXP BIT<59:58> INSERT ERR"
10903  041406  027524  046103  042122
10904  041414  046057  042504  050130
10905  041422  041040  052111  032474
10906  041430  035071  034065  020076
10907  041436  047111  042523  052122
10908  041444  042440  051122     000
10909  041451     105  051103  043057  EMBO:   .ASCIZ  "ECR/FCCR EXCEPTION+FCC ERR"
10910  041456  041503  020122  054105
10911  041464  042503  052120  047511
10912  041472  025516  041506  020103
10913  041500  051105  000122
10914  041504  050106  047111  052111  EMCA:   .ASCIZ  "FPINIT FLOW, UNEXP'D FPSHI#FEC ERR"
10915  041512  043040  047514  026127
10916  041520  052440  042516  050130
10917  041526  042047  043040  051520
10918  041534  044510  043043  041505
10919  041542  042440  051122     000
10920  041547     106  044520  044516  EMCB:   .ASCIZ  "FPINIT FLOW, HFP DIDN'T UBREAK ERR"
10921  041554  020124  046106  053517
10922  041562  020054  043110  020120
10923  041570  044504  047104  052047
10924  041576  052440  051102  040505
10925  041604  020113  051105  000122
10926  041612  046502  053457  050106  EMCC:   .ASCIZ  "BM/WFP ILLEGL.INTRNL.ADDR ERR"
10927  041620  044440  046114  043505
10928  041626  027114  047111  051124
10929  041634  046116  040456  042104
10930  041642  020122  051105  000122
10931  041650  051505  040520  027104  EMCD:   .ASCIZ  "ESPAD.B ADDRS ERR; R[DF]"
10932  041656  020102  042101  051104
10933  041664  020123  051105  035522
10934  041672  051040  042133  056506
10935  041700     000
10936  041701     105  050123  042101  EMCE:   .ASCIZ  "ESPAD.B ADDRS ERR; R[SF]"
10937  041706  041056  040440  042104
10938  041714  051522  042440  051122
10939  041722  020073  055522  043123
10940  041730  000135
10941  041732  051505  040520  027104  EMCF:   .ASCIZ  "ESPAD.A ADDRS ERR; R[DF]"
10942  041740  020101  042101  051104
10943  041746  020123  051105  035522
10944  041754  051040  042133  056506
10945  041762     000
10946  041763     105  050123  042101  EMCG:   .ASCIZ  "ESPAD.A ADDRS ERR; R[SF]"
```

```
10947  041770  040456  040440  042104
10948  041776  051522  042440  051122
10949  042004  020073  055522  043123
10950  042012  000135
10951  042014  042114  054105  027520  EM^H:    .ASCIZ  "LDEXP/STEXP FPINMUX(DOUT) ERR"
10952  042022  052123  054105  020120
10953  042030  050106  047111  052515
10954  042036  024130  047504  052125
10955  042044  020051  051105  000122
10956
10957
10958                                           ;DATA HEADERS HERE
10959  042052                                  DHA:
10960  042052  026455  050106  026523  DHB:    .ASCIZ  "--FPS- --FEC-  --FEA-  OLD-SP  OLD-PC  OLD-PS"
10961  042060  026411  043055  041505
10962  042066  004455  026455  042506
10963  042074  026501  047411  042114
10964  042102  051455  004520  046117
10965  042110  026504  041520  047411
10966  042116  042114  050055  000123
10967  042124  050103  042525  051122  DHC:    .ASCII  "CPUERR MEMERR  "                ;CONT ON NEXT LINE
10968  042132  046411  046505  051105
10969  042140  004522
10970  042142  046117  026504  050123  DHAC:   .ASCIZ  "OLD-SP OLD-PC  OLD-PS"
10971  042150  047411  042114  050055
10972  042156  004503  046117  026504
10973  042164  051520     000
10974  042167     055  044515  051105  DHF:    .ASCIZ  "-MIER- -MAND-  E-DATA  R-DATA"
10975  042174  004455  046455  047101
10976  042202  026504  042411  042055
10977  042210  052101  004501  026522
10978  042216  040504  040524     000
10979  042223     122  046517  021443  DHG:    .ASCIZ  "ROM### ROMADR  E-DATA  R-DATA"
10980  042230  004443  047522  040515
10981  042236  051104  042411  042055
10982  042244  052101  004501  026522
10983  042252  040504  040524     000
10984  042257     122  046517  021443  DHH:    .ASCIZ  "ROM### TOTERR  S<>S+C  BD-IOR  BD-AND"
10985  042264  004443  047524  042524
10986  042272  051122  051411  037074
10987  042300  025523  004503  042102
10988  042306  044455  051117  041011
10989  042314  026504  047101  000104
10990  042322  047522  040515  051104  DHI:    .ASCIZ  "ROMADR -FIR--  FPUBRK  PRVFPS  FPSFEC  -FEA--"
10991  042330  026411  044506  026522
10992  042336  004455  050106  041125
10993  042344  045522  050011  053122
10994  042352  050106  004523  050106
10995  042360  043123  041505  026411
10996  042366  042506  026501  000055
10997  042374  026522  050106  044123  DHL:    .ASCIZ  "R-FPSHI/FEC-E  FLG<5>  -FIR--"
10998  042402  027511  042506  026503
10999  042410  004505  046106  036107
11000  042416  037065  026411  044506
11001  042424  026522  000055
11002  042430  047111  052123  027122  DHM:    .ASCIZ  "INSTR. OLD-SP  OLD-PC  OLD-PS"
```

```
11003  042436  047411  042114  051455
11004  042444  004520  046117  026504
11005  042452  041520  047411  042114
11006  042460  050055  000123
11007  042464  026522  026455  046106  DHN:    .ASCIZ  "R---FLAGS----E R---WHAMI----E"
11008  042472  043501  026523  026455
11009  042500  042455  051011  026455
11010  042506  053455  040510  044515
11011  042514  026455  026455  000105
11012  042522  026505  026455  046106  DHO:    .ASCIZ  "E---FLAGS----R"
11013  042530  043501  026523  026455
11014  042536  051055     000
11015  042541     102  052515  051102  DHP:    .ASCII  "BMUBRK "       ;CONT
11016  042546  004513
11017  042550  026522  046106  043501  DHR:    .ASCIZ  "R-FLAGS"
11018  042556  000123
11019  042560  026522  026455  046106  DHQ:    .ASCIZ  "R---FLAGS----E R---UADDR----E"
11020  042566  043501  026523  026455
11021  042574  042455  051011  026455
11022  042602  052455  042101  051104
11023  042610  026455  026455  000105
11024  042616  051503  040520  051104  DHS:    .ASCIZ  "CSPADR E---FPCNST---R"
11025  042624  042411  026455  043055
11026  042632  041520  051516  026524
11027  042640  026455  000122
11028  042644  054105  023520  004504  DHW:    .ASCIZ  "EXP'D  RCV'D"
11029  042652  041522  023526  000104
11030  042660  022055  050106  026523  DHX:    .ASCIZ  "-$FPS-"
11031  042666     000
11032  042667     055  026455  026455  DHY:    .ASCIZ  "-------------EXP-------------- -------------RCV--------------"
11033  042674  026455  026455  026455
11034  042702  026455  054105  026520
11035  042710  026455  026455  026455
11036  042716  026455  026455  026455
11037  042724  004455  026455  026455
11038  042732  026455  026455  026455
11039  042740  026455  051055  053103
11040  042746  026455  026455  026455
11041  042754  026455  026455  026455
11042  042762  026455     000
11043  042765     105  050130  026455  DHZ:    .ASCIZ  "EXP---FPS--RCV -FEC--  -FEA--"
11044  042772  043055  051520  026455
11045  043000  041522  004526  043055
11046  043006  041505  026455  026411
11047  043014  042506  026501  000055
11048  043022  021443  051123  041526  DHAA:   .ASCIZ  "**SRVC BMSRVC"
11049  043030  041011  051515  053122
11050  043036  000103
11051  043040  026522  050106  044123  DHAB:   .ASCIZ  "R-FPSHI/FEC"
11052  043046  027511  042506  000103
11053  043054                          DHAL:
11054  043054                          DHAM:
11055  043054  044123  043111  026524  DHAN:   .ASCIZ  "SHIFT-VALUE"
11056  043062  040526  052514  000105
11057  043070  041501  021503  021443  DHAQ:   .ASCIZ  "ACC***"
11058  043076     000
```

```
11059  043077                                  DHAT:
11060  043077     101  041503  021443  DHAU:    .ASCIZ  "ACC### E-DATA  R-DATA"
11061  043104  004443  026505  040504
11062  043112  040524  051011  042055
11063  043120  052101  000101
11064  043124  050106  047111  052123  DHAX:    .ASCIZ  "FPINST E-UBRK  R-FPSHI/FEC"
11065  043132  042411  052455  051102
11066  043140  004513  026522  050106
11067  043146  044123  027511  042506
11068  043154  000103
11069  043156                          DHBI:
11070  043156  040504  040524  051455  DHAY:    .ASCIZ  "DATA-SET"
11071  043164  052105     000
11072  043167                          DHBA: DHBB: DHBC: DHBD: DHBF: DHCA:
11073  043167     105  052455  051102  DHBG:    .ASCIZ  "E-UBRK R-FPSHI/FEC"
11074  043174  004513  026522  050106
11075  043202  044123  027511  042506
11076  043210  000103
11077  043212  051105  032474  030072  DHBE:    .ASCIZ  "ER<5:0> E-CNTR/EXPNT-R"
11078  043220  020076  026505  047103
11079  043226  051124  042457  050130
11080  043234  052116  051055     000
11081  043241     105  052455  051102  DHBH:    .ASCIZ  "E-UBRK -FPS--  ER<8:0> R-FPSHI/FEC"
11082  043246  004513  043055  051520
11083  043254  026455  042411  036122
11084  043262  035070  037060  051040
11085  043270  043055  051520  044510
11086  043276  043057  041505     000
11087  043303     111  052116  043505  DHBJ:    .ASCIZ  "INTEGER"
11088  043310  051105     000
11089  043313     055  050106  026523  DHBL:    .ASCIZ  "-FPS--"
11090  043320  000055
11091  043322  051120  027126  050106  DHBO:    .ASCIZ  "PRV.FPS EXPD--FPS--RCVD EXPD--FEC--RCVD"
11092  043330  020123  054105  042120
11093  043336  026455  050106  026523
11094  043344  051055  053103  020104
11095  043352  054105  042120  026455
11096  043360  042506  026503  051055
11097  043366  053103  000104
11098  043372  050106  047111  052123  DHCC:    .ASCIZ  "FPINST E---CPUERR---R  0=TRAP/-1=NO.TRAP"
11099  043400  042411  026455  041455
11100  043406  052520  051105  026522
11101  043414  026455  004522  036460
11102  043422  051124  050101  026457
11103  043430  036461  047516  052056
11104  043436  040522  000120
11105  043442  026505  026455  054105  DHCH:    .ASCIZ  "E---EXPNT---R"
11106  043450  047120  026524  026455
11107  043456  000122
11108
11109
11110                                           ;DATA ADDRESS VECTOR
11111                                           .EVEN
11112  043460                          DTA:
11113  043460  002612  002614  002616  DTB:     .WORD   FPS,FEC,FEA,OLDSP,OLDPC,OLDPS,0
11114  043466  002772  002766  002770
```

```
11115  043474  000000
11116  043476  177766  177744          DTC:    .WORD   CPUERR,MEMERR   ;CONT ON NEXT LINE
11117  043502  002772  002766  002770  DTAC:   .WORD   OLDSP,OLDPC,OLDPS,0
11118  043510  000000
11119  043512                          DTL:
11120  043512                          DTQ:
11121  043512                          DTN:
11122  043512  002746  002750  002754  DTF:    .WORD   EREG0,EREG1,EREG3,EREG2,0
11123  043520  002752  000000
11124  043524  002756  002750  002752  DTG:    .WORD   EREG4,EREG1,EREG2       ;CONT ON NEXT LINE
11125  043532  002754  000000          .WORD   EREG3,0
11126  043536  001302  001316  001320  DTH:    .WORD   $REG0,$REG6,$REG7,$REG11,$REG12,0
11127  043544  001324  001326  000000
11128  043552  002756  002760  002754  DTI:    .WORD   EREG4,EREG5,EREG3,$FPS  ;CONT ON NEXT LINE
11129  043560  002610
11130  043562  002750                  DTO:    .WORD   EREG1   ;CONT ON NEXT LINE
11131  043564                          DTAB:
11132  043564                          DTAQ:
11133  043564  002746  000000          DTR:    .WORD   EREG0,0
11134  043570  002750                  DTS:    .WORD   EREG1   ;CONT ON NEXT LINE
11135  043572  002752  002746  000000  DTP:    .WORD   EREG2,EREG0,0
11136  043600  002610  000000          DTX:    .WORD   $FPS,0
11137  043604  002610  002612  002614  DTZ:    .WORD   $FPS,FPS,FEC,FEA,0
11138  043612  002616  000000
11139  043616  002630  002746  000000  DTAA:   .WORD   UBCNTR,EREG0,0
11140  043624                          DTAY: DTBI: DTBJ: DTBL:
11141  043624  002750  000000          DTAL:   .WORD   EREG1,0
11142  043630                          DTAM:
11143  043630  002756  000000          DTAN:   .WORD   EREG4,0
11144  043634  002774  002772  002766  DTM:    .WORD   FPINST,OLDSP,OLDPC,OLDPS,0
11145  043642  002770  000000
11146  043646                          DTAT:
11147  043646  002746  002646  002656  DTAU:   .WORD   EREG0,MFAC0,MFAC1,0
11148  043654  000000
11149  043656  002752  002750  002746  DTAX:   .WORD   EREG2,EREG1,EREG0,0
11150  043664  000000
11151  043666  002754  002752  002750  DTBH:   .WORD   EREG3,EREG2,EREG1,EREG0,0
11152  043674  002746  000000
11153  043700  002754  002746  002752  DTBE:   .WORD   EREG3,EREG0,EREG2,0
11154  043706  000000
11155  043710                          DTBA: DTBB: DTBC: DTBD: DTBF: DTCA:
11156  043710  002754  002746  000000  DTBG:   .WORD   EREG3,EREG0,0
11157  043716  002752  002750  002746  DTCC:   .WORD   EREG2,EREG1,EREG0,EREG3,0
11158  043724  002754  000000
11159  043730  002746  002750  002752  DTBO:   .WORD   EREG0,EREG1,EREG2,EREG3,EREG4,0
11160  043736  002754  002756  000000
11161  043744  002746  002750  000000  DTCH:   .WORD   EREG0,EREG1,0
11162
11163
11164                                   ;FLOATING POINT DATA VECTORS
11165                                   .EVEN
11166  043752  044370  002646  044476  DV1:    .WORD   EAC0,MFAC0,  RAC0,MFAC2,  0
11167  043760  002666  000000
11168  043764  044572  002646  044414  DV2:    .WORD   HAC0,MFAC0,  EAC2,MFAC6   RAC2,MFAC4,  0
11169  043772  002726  044522  002706
11170  044000  000000
```

```
11171  044002  044572  002646  044402  DV3:    .WORD   HAC0,MFAC0,  EAC1,MFAC5,  RAC1,MFAC3,  0
11172  044010  002716  044510  002676
11173  044016  000000
11174  044020  044510  002656  044522  DVAD:   .WORD   RAC1,MFAC1,  RAC2,MFAC2,  0
11175  044026  002666  000000
11176  044032  044572  002646  044402  DVAE:   .WORD   HAC0,MFAC0,  EAC1,MFAC1,  RAC1,MFAC2,  RAC2,MFAC3,  0
11177  044040  002656  044510  002666
11178  044046  044522  002676  000000
11179  044054                          DVBJ:
11180  044054  044370  002646  044476  DVAF:   .WORD   EAC0,MFAC0,  RAC0,MFAC1,  0
11181  044062  002656  000000
11182  044066  044604  002706  044630  DVAG:   .WORD   HAC1,MFAC4,  HAC3,MFAC3,  RAC2,MFAC5,  EAC2,MFAC2,  0
11183  044074  002676  044522  002716
11184  044102  044414  002666  000000
11185  044110  044572  002646  044402  DVAH:   .WORD   HAC0,MFAC0,  EAC1,MFAC1,  RAC1,MFAC2,  0
11186  044116  002656  044510  002666
11187  044124  000000
11188  044126  044510  002646  000000  DVAI:   .WORD   RAC1,MFAC0,  0
11189  044134  044356  002646          DVAJ:   .WORD   EACX,MFAC0      ;CONT
11190  044140  044464  002656  000000  DVAK:   .WORD   RACX,MFAC1,  0
11191  044146  044572  002646  044534  DVAL:   .WORD   HAC0,MFAC0,  RAC3,MFAC1,  0
11192  044154  002656  000000
11193  044160                          DVB0:
11194  044160  044572  002646  044604  DVAV:   .WORD   HAC0,MFAC0,  HAC1,MFAC1,  EAC2,MFAC2,  RAC2,MFAC3,  0
11195  044166  002656  044414  002666
11196  044174  044522  002676  000000
11197  044202  044772  002646  044762  DVBH:   .WORD   EBSF,MFAC0,  EADF,MFAC1,  0
11198  044210  002656  000000
11199  044214                          DVBI:
11200  044214  044666  002646  044677  DVAM:   .WORD   AC6SF,MFAC0,  ACODF,MFAC1,  EXPRES,MFAC2,  RCVRES,MFAC3,  0
11201  044222  002656  044710  002666
11202  044230  044725  002676  000000
11203  044236                          DVB8:
11204  044236  044742  002646  044752  DVBA:   .WORD   EASF,MFAC0,  EBDF,MFAC1,  0
11205  044244  002656  000000
11206  044250  044742  002646  000000  DVBC:   .WORD   EASF,MFAC0,  0
11207  044256  044762  002646  044772  DVBD:   .WORD   EADF,MFAC0,  EBSF,MFAC1,  EXEAEB,MFAC2,  RCEAEB,MFAC3,  0
11208  044264  002656  045002  002666
11209  044272  045016  002676  000000
11210  044300  045032  002646  045045  DVBF:   .WORD   SDEADF,MFAC0,  SSEBSF,MFAC1,  0
11211  044306  002656  000000
11212  044312  044402  002646  044510  DVBL:   .WORD   EAC1,MFAC0,  RAC1,MFAC1,  0
11213  044320  002656  000000
11214  044324  044476  002646  000000  DVBM:   .WORD   RAC0,MFAC0,  0
11215  044332  045060  002646  045100  DVBN:   .WORD   EAC1R3,MFAC0,  RAC1R3,MFAC1,  0
11216  044340  002656  000000
11217  044344                          DVCD: DVCF: DVCG:
11218  044344  044356  003164  044464  DVCE:   .WORD   EACX,FPSEFZ,  RACX,MFAC0,  0
11219  044352  002646  000000
11220
11221                                   ;RANDOM ASCII MESSAGES FOR ABOVE:
11222  044356  054105  042120  040440  EACX:   .ASCIZ  "EXPD ACC:"
11223  044364  041503  000072
11224  044370  054105  042120  040440  EAC0:   .ASCIZ  "EXPD AC0:"
11225  044376  030103  000072
11226  044402  054105  042120  040440  EAC1:   .ASCIZ  "EXPD AC1:"
```

```
11227  044410  030503  000072
11228  044414  054105  042120  040440  EAC2:   .ASCIZ  "EXPD AC2:"
11229  044422  031103  000072
11230  044426  054105  042120  040440  EAC3:   .ASCIZ  "EXPD AC3:"
11231  044434  031503  000072
11232  044440  054105  042120  040440  EAC4:   .ASCIZ  "EXPD AC4:"
11233  044446  032103  000072
11234  044452  054105  042120  040440  EAC5:   .ASCIZ  "EXPD AC5:"
11235  044460  032503  000072
11236  044464  041522  042126  040440  RACX:   .ASCIZ  "RCVD ACC:"
11237  044472  041503  000072
11238  044476  041522  042126  040440  RAC0:   .ASCIZ  "RCVD AC0:"
11239  044504  030103  000072
11240  044510  041522  042126  040440  RAC1:   .ASCIZ  "RCVD AC1:"
11241  044516  030503  000072
11242  044522  041522  042126  040440  RAC2:   .ASCIZ  "RCVD AC2:"
11243  044530  031103  000072
11244  044534  041522  042126  040440  RAC3:   .ASCIZ  "RCVD AC3:"
11245  044542  031503  000072
11246  044546  041522  042126  040440  RAC4:   .ASCIZ  "RCVD AC4:"
11247  044554  032103  000072
11248  044560  041522  042126  040440  RAC5:   .ASCIZ  "RCVD AC5:"
11249  044566  032503  000072
11250  044572  043110  050120  040440  HAC0:   .ASCIZ  "HFPP AC0:"
11251  044600  030103  000072
11252  044604  043110  050120  040440  HAC1:   .ASCIZ  "HFPP AC1:"
11253  044612  030503  000072
11254  044616  043110  050120  040440  HAC2:   .ASCIZ  "HFPP AC2:"
11255  044624  031103  000072
11256  044630  043110  050120  040440  HAC3:   .ASCIZ  "HFPP AC3:"
11257  044636  031503  000072
11258  044642  043110  050120  040440  HAC4:   .ASCIZ  "HFPP AC4:"
11259  044650  032103  000072
11260  044654  043110  050120  040440  HAC5:   .ASCIZ  "HFPP AC5:"
11261  044662  032503  000072
11262  044666  041501  055466  043123  AC6SF:  .ASCIZ  "AC6[SF]:"
11263  044674  035135     000
11264  044677     101  030103  042133  ACODF:  .ASCIZ  "AC0[DF]:"
11265  044704  056506  000072
11266  044710  054105  042120  051040  EXPRES: .ASCIZ  "EXPD RESULT:"
11267  044716  051505  046125  035124
11268  044724     000
11269  044725     122  053103  020104  RCVRES: .ASCIZ  "RCVD RESULT:"
11270  044732  042522  052523  052114
11271  044740  000072
11272  044742  040505  051533  056506  EASF:   .ASCIZ  "EA[SF]:"
11273  044750  000072
11274  044752  041105  042133  056506  EBDF:   .ASCIZ  "EB[DF]:"
11275  044760  000072
11276  044762  040505  042133  056506  EADF:   .ASCIZ  "EA[DF]:"
11277  044770  000072
11278  044772  041105  051533  056506  EBSF:   .ASCIZ  "EB[SF]:"
11279  045000  000072
11280  045002  054105  042120  042440  EXEAEB: .ASCIZ  "EXPD EA+EB:"
11281  045010  025501  041105  000072
11282  045016  041522  042126  042440  RCEAEB: .ASCIZ  "RCVD EA+EB:"
```

```
 11283  045024  025501  041105  000072
 11284  045032  042123  042457  055501  SDEADF: .ASCIZ  "SD/EA[DF]:"
 11285  045040  043104  035135     000
 11286  045045     123  027523  041105  SSEBSF: .ASCIZ  "SS/EB[SF]:"
 11287  045052  051533  056506  000072
 11288  045060  054105  042120  040440  EAC1R3: .ASCIZ  "EXPD AC1/RITE3:"
 11289  045066  030503  051057  052111
 11290  045074  031505  000072
 11291  045100  041522  042126  040440  RAC1R3: .ASCIZ  "RCVD AC1/RITE3:"
 11292  045106  030503  051057  052111
 11293  045114  031505  000072
 11294                                  .EVEN
 11295
 11296
 11297                                  ;THE END
 11298          000001                  .END
```

```
ABASE = 000000              386
ACDW1 = 000000              386
ACDW2 = 000000              386
ACPUOP= 000000              386   401
AC0   =%000000              197#  2384*  2386   2608*  2610*  2612*  2614   2991*  2992   3086*  3089   3125*  3129
                           3146*  3147   3150*  3154   3169*  3238*  3241   3261*  3262   3279*  3283   3300*  3301
                           3304*  3308   3322*  3324   3407*  3414   3535*  3542   3647*  3649*  3767*  3768*  3774
                           3895*  3995*  3996*  4023*  4024*  4051*  4052*  4079*  4080*  4145*  4146*  4169*  4170*
                           4246*  4254   4449*  4450*  4451   4545*  4546*  4547   4657*  4658   4978*  4979*  4980*
                           4984*  4985   5045*  5049   5065*  5066   5069*  5073   5087*  5104*  5107   5178*  5195
                           5190   5249*  5256   5261   5320*  5326   5330   5389*  5396   5401   5460*  5466   5470
                           5548*  5635*  5702*  5704*  5706*  5781*  5849*  5850*  5923*  5925   5934*  5935*  6051*
                           6053   6061*  6172*  6179*  6290*  6297   6691*  6692*  6696   6988*  6989*  6993   7111*
                           7112*  7116   7215*  7258*  7563*  7856*  8236*  8244*  8416*  8423*  8551*  8558   8726*
                           8740   8872*  8885   9007*  9021
ACODF   044677             11200  11264#
AC1   =%000001              198#  2416*  2418   2644*  2646*  2648*  2650   3087*  3106*  3126*  3148*  3239*  3259*
                           3280*  3302*  3757*  3767   3988*  3989   3995   4016*  4017   4023   4044*  4045   4051
                           4072*  4073   4079   4140*  4145   4163*  4169   4247*  4253   4688*  4689   5046*  5067*
                           5105*  5123*  5553*  5559   5640*  5645   5703*  5706   5711*  5716   5785*  5787   5856
                           5942*  5947   6066*  6071   6173*  6186   6291*  6296   6686*  6691   6983*  6988   7216*
                           7223   7259*  7266   7568   7861   8237*  8239   8240   8245*  8247   8252   8299*  8300
                           8305   8417*  8419   8420   8424*  8426   8431   8552*  8557   8727*  8741   8873*  8886
                           9008*  9022
AC2   =%000002              199#  2448*  2450   2680*  2682*  2684*  2686   2884*  2886   2888*  2890   3088*  3107*
                           3127*  3149*  3240*  3260*  3281*  3303*  3904*  3910   4141*  4146   4164*  4165   4719*
                           4720   4814*  4815   4816*  4817   5047*  5068*  5106*  5124*  6296*  6297*  6301   7217*
                           7224   7260*  7267   7569   7862   8252*  8253*  8258   8305*  8306*  8310   8431*  8432*
                           8437   8557*  8558*  8564   8728*  8742   9009*  9023
AC3   =%000003              200#  2480*  2482   2716*  2718*  2720*  2722   2848*  2850   2852*  2854   3105*  3108
                           3124*  3125   3129*  3133   3147*  3150   3167*  3168   3170*  3174   3241*  3242   3258*
                           3261   3278*  3279   3283*  3287   3301*  3304   3321*  3322   3324*  3328   3408*  3414
                           3536*  3542*  3758*  3768   3990*  3991   4018*  4019   4046*  4047   4074*  4075   4253*
                           4254*  4750*  4751   4781*  4782   4783*  4784   5044*  5045   5049*  5053   5066*  5069
                           5085*  5086   5088*  5092   5122*  5125   5183*  5185*  5191   5254*  5256*  5262   5325*
                           5326*  5331   5394*  5396*  5402   5465*  5466*  5471   8239*  8253   8294*  8295   8306
                           8419*  8432
AC4   =%000004              201#  2850*  2852   3128*  3168*  3170   3282*  3323*  4165*  4170   4782*  4783   5048*
                           5086*  5088
AC5   =%000005              202#  2886*  2888   4815*  4816
AC6   =%000006              203#  3996   4024   4052   4080
AC6SF   044666             11200  11262#
AC7   =%000007              204#
ADDD01  022030              6710   6913#
ADDW0 = 000000              386
ADDW1 = 000000              386
ADDW10= 000000              386
ADDW11= 000000              386
ADDW12= 000000              386
ADDW13= 000000              386
ADDW14= 000000              386
ADDW15= 000000              386
ADDW2 = 000000              386
ADDW3 = 000000              386
ADDW4 = 000000              386
ADDW5 = 000000              386
```

```
ADDW6 = 000000            386
ADDW7 = 000000            386
ADDW8 = 000000            386
ADDW9 = 000000            386
ADEVCT= 000000            386     392
ADEVM = 000000            386
AENV  = 000000            386     397
AENVM = 000000            386     398
AFATAL= 000000            386     389
ALTN  = 125252            226#    227
ALTP  = 052525            224#    225    7505    7753
ALT4N = 170360            229#
ALT4P = 007417            228#
AMADR1= 000000            386
AMADR2= 000000            386
AMADR3= 000000            386
AMADR4= 000000            386
AMAMS1= 000000            386
AMAMS2= 000000            386
AMAMS3= 000000            386
AMAMS4= 000000            386
AMSGAD= 000000            386     394
AMSGLG= 000000            386     395
AMSGTY= 000000            386     388
AMTYP1= 000000            386
AMTYP2= 000000            386
AMTYP3= 000000            386
AMTYP4= 000000            386
AN    = 125252            227#    667    668    669    678    680    688    4567   4576   4585   4594
AP    = 052525            225#    662    663    664    677    678    686    4570   4579   4588   4597   6049   6109
APASS = 000000            386     391
APRIOR= 000000            386
APTCSU= 000040          10118   10228#
APTENV= 000001            780    9894   10111  10184  10226#
APTSIZ= 000200          10225#
APTSP0= 000100          10113   10186  10227#
ASH64I= 104430            5965    5966   6087   6088   10368#
ASH64M= 104431            7574    7575   7870   7871   10369#
ASWREG= 000000            386     399
ATESTN= 000000            386     390
AUNIT = 000000            386     393
AUSWR = 000000            386     400
AVECT1= 000000            386
AVECT2= 000000            386
A01$    020650          6721#
A02$    020700          6728#
A03$    020730          6735#
A04$    020760          6742#
A05$    021010          6749#
A06$    021040          6756#
A07$    021070          6763#
A10$    021120          6770#
A11$    021150          6777#
A12$    021200          6788#
A13$    021230          6795#
A14$    021260          6802#
```

```
A15$    021310          6809#
A16$    021340          6816#
A17$    021370          6823#
A20$    021420          6830#
A21$    021450          6840#
A22$    021500          6847#
A23$    021530          6854#
A24$    021560          6861#
A25$    021610          6868#
A26$    021640          6878#
A27$    021670          6885#
A30$    021720          6892#
A31$    021750          6899#
A32$    022000          6906#
BGNMES  003242           714#   804
BGNPCT  003732           815#   817
BIT0  = 000001           126#  5960  6084  7972  7973  7988  8018
BIT00 = 000001           116#   126  7776
BIT01 = 000002           115#   125  1774
BIT02 = 000004           114#   124
BIT03 = 000010           113#   123  7809  9490
BIT04 = 000020           112#   122  9489  9916
BIT05 = 000040           111#   121  8812  9488
BIT06 = 000100           110#   120  9487
BIT07 = 000200           109#   119  8816  8973  9228  9486  9519
BIT08 = 000400           106#   118  8835  9485  9792
BIT09 = 001000           107#   117  9800  9904
BIT1  = 000002           125#  7972  7973  7987  8020  8025  8030
BIT10 = 002000           106#  1893  9859
BIT11 = 004000           105#  9807
BIT12 = 010000           104#   830
BIT13 = 020000           103#  9866
BIT14 = 040000           102#  1805  1816  9334  9778  9917
BIT15 = 100000           101#  1805  1816  8717  8719  8865  8988  8990  9255  9753 10422 10429
BIT2  = 000004           124#  7970  7971  7985  7986  8019  8020  8024  8029
BIT3  = 000010           123#  1884  7970  7971  7985  7986  8018  8025
BIT4  = 000020           122#  1884  7968  7969  7984  8017
BIT5  = 000040           121#   898  1884  7968  7969  7983  8029
BIT6  = 000100           120#   827   898  7966  7967  7981  7982  9019  9936
BIT7  = 000200           119#  4278  7966  7967  7981  7982  8024
BIT8  = 000400           118#   898
BIT9  = 001000           117#   938
BPTVEC= 000014           133#
CCONLY= 177760           252#
CLRDW = 104411          1002   1405  1765  2047  2142  2252 10353#
CLRFP = 104413          1659   1817  2141  2251 10355#
CLRUB = 104415          1009   1027  1460 10357#
CMPXXM= 104427         10367#
CMP32M= 104426          3095   3114  3136  3157  3177  3248  3268  3290  3311  3331  4456  5789  6995
                        7118   8566 10366#
CMP64M= 104425          4552   4662  4693  4724  4755  4788  4821  4987  5055  5075  5094  5112  5130
                        5194   5202  5265  5273  5334  5342  5405  5413  5474  5482  5646  5717  5858
                        5950   6074  6187  6304  6698  7229  7237  7269  7278  8261  8313  8440  8745
                        8755   8889  8899  9026  9036  9041 10365#
CND$ES= 104405          7504   7752 10349#
CODEA   025256          7799   8081#
```

```
CODEB   025316          7803    8110#
CODEC   025356          7812    8140#
CODED   025376          7817    8162#
CODEI   006016          1878    1908#
CODEJ   005712          1761    1780    1859#
CPUBRK= 177770           210#    993*   1397*
CPUCCR= 177746           211#
CPUERR= 177766           208#   1275    1301    9228   11116
CR     = 000015            41#    714     719     726   10157   10167
CRLF   = 000200            42#  10128   10167
D      = 000200          6203#   6214    6220
DBLDAT= 104420          8662    8663    8790    8791    8939    8940   10360#
DDISP  = 177570            48#    347     768
DHA     042052           425   10959#
DHAA    043022           479   11048#
DHAB    043040           481   11051#
DHAC    042142           477   10970#
DHAL    043054           525   11053#
DHAM    043054           527   11054#
DHAN    043054           529   11055#
DHAQ    043070           515   11057#
DHAT    043077           531   11059#
DHAU    043077           533   11060#
DHAX    043124           539   11064#
DHAY    043156           541   11070#
DHB     042052           427   10960#
DHBA    043167           543   11072#
DHBB    043167           545   11072#
DHBC    043167           547   11072#
DHBD    043167           509     549   11072#
DHBE    043212           551   11077#
DHBF    043167           553   11072#
DHBG    043167           555   11073#
DHBH    043241           557   11081#
DHBI    043156           559   11069#
DHBJ    043303           561   11087#
DHBL    043313           563   11089#
DHBO    043322           569   11091#
DHC     042124           429     431     433   10967#
DHCA    043167           495     497   11072#
DHCC    043372           499   11098#
DHCH    043442           571   11105#
DHF     042167           451   10974#
DHG     042223           453   10979#
DHH     042257           455   10964#
DHI     042322           457     459   10990#
DHL     042374           461   10997#
DHM     042430           463   11002#
DHN     042464           465   11007#
DHO     042522           467   11012#
DHP     042541           469   11015#
DHQ     042560           471   11019#
DHR     042550           473   11017#
DHS     042616           475   11024#
DHW     042644         11028#
DHX     042660           443     445     447     449   11030#
```

```
DHY     042667          11032#
DHZ     042765            441   11043#
DISPLA  001256            347#    768#    776#   9822#   9858#
DISPRE  000174            278#    776
DIVF01  022336           7007    7051#
DSWR  = 177570             47#    346     767
DTA     043460            425   11112#
DTAA    043616            479   11139#
DTAB    043564            481   11131#
DTAC    043502            477   11117#
DTAL    043534            525   11141#
DTAM    043630            527   11142#
DTAN    043630            529   11143#
DTAQ    043564            515   11132#
DTAT    043646            531   11146#
DTAU    043646            533   11147#
DTAX    043656            539   11149#
DTAY    043624            541   11140#
DTB     043460            427   11113#
DTBA    043710            543   11155#
DTBB    043710            545   11155#
DTBC    043710            547   11155#
DTBD    043710            509     549   11155#
DTBE    043700            551   11153#
DTBF    043710            553   11155#
DTBG    043710            555   11156#
DTBH    043666            557   11151#
DTBI    043624            559   11140#
DTBJ    043624            561   11140#
DTBL    043624            563   11140#
DTBO    043730            569   11159#
DTC     043476            429     431     433   11116#
DTCA    043710            495     497   11155#
DTCC    043716            499   11157#
DTCH    043744            571   11161#
DTF     043512            451   11122#
DTG     043524            453   11124#
DTH     043536            455   11126#
DTI     043552            457     459   11128#
DTL     043512            461   11119#
DTM     043634            463   11144#
DTN     043512            465   11121#
DTO     043562            467   11130#
DTP     043572            469   11135#
DTQ     043512            471   11120#
DTR     043564            473   11133#
DTS     043570            475   11134#
DTX     043600            443     445     447     449   11136#
DTZ     043604            441   11137#
DVAD    044020            483   11174#
DVAE    044032            485     487   11176#
DVAF    044054            511     513   11180#
DVAG    044066            489     491   11182#
DVAH    044110            521     527     529     537   11185#
DVAI    044126            523     525   11188#
DVAJ    044134            515   11189#
```

```
DVAK     044140              435     437   11190#
DVAL     044146              517     519   11191#
DVAM     044214              541   11200#
DVAV     044160              535   11194#
DVBA     044236              543   11204#
DVBB     044236              545   11203#
DVBC     044250              547   11206#
DVBD     044256              509     549   11207#
DVBF     044300              553   11210#
DVBH     044202              557   11197#
DVBI     044214              559   11199#
DVBJ     044054              561   11179#
DVBL     044312              563   11212#
DVBM     044324              565   11214#
DVBN     044332              567   11215#
DVBO     044160              569   11193#
DVCD     044344              501   11217#
DVCE     044344              503   11218#
DVCF     044344              505   11217#
DVCG     044344              507   11217#
DV1      043752              439   11166#
DV2      043764              443     449   11168#
DV3      044002              445     447   11171#
DWCNTR   002636              616#    826#   9116#   9146#   9764#  10396#
DWESCP   002634              615#   9140    9142    9760#  10390#  10393#
DWFLAG   002644              619#   9130    9759#  10391#  10394#
DWICNT   003000              656#    825#    826    1759#   2023#   9146   9765#  10396
DWLOOP   002640              617#   1153#   1635#   1764#   2026#   2117#   2227#   2376#   2408#   2440#   2472#   2600#   2636#
                            2672#   2708#   2840#   2876#   2982#   3399#   3527#   3637#   3752#   3894#   4236#   4441#   4535#
                            4646#   4677#   4708#   4739#   4770#   4803#   5544#   5701#   5933#   6059#   6163#   6282#   6680#
                            6976#   7103#   7249#   7552#   7789#   8243#   8298#   8540#   8670#   8798#   8948#   9120    9147
                            9762#  10397
DWLOPC   002632              614#   9112    9145#   9761#
DWOLOP   002642              618#   9120    9147#   9763#  10397#
DW$CNT=  000006              655#    656     825    9764    9765
DW11LC=  177546              214#    827#   9062#   9846#   9936#
DW11LI   030500              706    9112#
DW11LV=  000100              215#    706
DO1$     022162             7015#
DO2$     022176             7019#
DO3$     022212             7023#
DO4$     022226             7027#
DO5$     022242             7031#
DO6$     022256             7035#
DO7$     022272             7039#
D10$     022306             7043#
D11$     022322             7047#
EACX     044356            11189   11218   11222#
EACO     044370            11166   11180   11224#
EAC1     044402            11171   11176   11185   11212   11226#
EAC1R3   045060            11215   11288#
EAC2     044414            11168   11182   11194   11228#
EAC3     044426            11230#
EAC4     044440            11232#
EAC5     044452            11234#
EADF     044762            11197   11207   11276#
```

```
EADJ  = 104424          8710   8827   8982  10364#
EADJT   031530          9470   9485#
EASF    044742         11204  11206  11272#
EBDF    044752         11204  11274#
EBSF    044772         11197  11207  11278#
EMA     035656           425  10558#
EMAA    037116           479  10683#
EMAB    037141           481  10687#
EMAC    037211           477  10694#
EMAD    037254           483  10700#
EMAE    037330           485  10708#
EMAF    037356           487  10712#
EMAG    037413           489  10717#
EMAH    037460           491  10724#
EMAI    037525           493  10731#
EMAJ    037570           521  10737#
EMAK    037615           523  10741#
EMAL    037646           525  10746#
EMAM    037701           527  10751#
EMAN    037740           529  10757#
EMAO    040004           513  10763#
EMAP    040036           511  10768#
EMAQ    040070           515  10773#
EMAR    040107           517  10776#
EMAS    040153           519  10783#
EMAT    040220           531  10790#
EMAU    040255           533  10795#
EMAV    040312           535  10800#
EMAW    040345           537  10805#
EMAX    040401           539  10810#
EMAY    040436           541  10815#
EMB     035656           427  10559#
EMBA    040475           543  10821#
EMBB    040533           545  10827#
EMBC    040570           547  10832#
EMBD    040620           549  10836#
EMBE    040724           551  10848#
EMBF    040765           553  10854#
EMBG    041032           555  10861#
EMBH    041101           557  10868#
EMBI    041155           559  10876#
EMBJ    041224           561  10883#
EMBK    040661           509  10842#
EMBL    041270           563  10889#
EMBM    041333           565  10895#
EMBN    041401           567  10902#
EMBO    041451           569  10909#
EMC     035710           429  10564#
EMCA    041504           495  10914#
EMCB    041547           497  10920#
EMCC    041612           499  10926#
EMCD    041650           501  10931#
EMCE    041701           503  10936#
EMCF    041732           505  10941#
EMCG    041763           507  10946#
EMCH    042014           571  10951#
```

EMD	035734	433	10568#	
EME	035761	431	10572#	
EME1	036007	439	10576#	
EME2	036037	441	10581#	
EME3	036067	443	10586#	
EME4	036116	445	10590#	
EME5	036140	447	10593#	
EME6	036166	449	10597#	
EMF	036215	451	10601#	
EMG	036252	453	10606#	
EMH	036307	455	10611#	
EMI	036333	457	10615#	
EMJ	036407	459	10623#	
EMK1	036462	435	10631#	
EMK2	036511	437	10635#	
EML	036540	461	10639#	
EMM	036611	463	10646#	
EMN	036647	465	10652#	
EMO	036677	467	10657#	
EMP	036720	469	10660#	
EMQ	036762	471	10666#	
EMR	037023	473	10672#	
EMS	037070	475	10679#	
EMTVEC=	000030	136#	749*	750*
EMV001	001406	425#		
EMV002	001416	427#		
EMV003	001426	429#		
EMV004	001436	431#		
EMV005	001446	433#		
EMV006	001456	435#		
EMV007	001466	437#		
EMV010	001476	439#		
EMV011	001506	441#		
EMV012	001516	443#		
EMV013	001526	445#		
EMV014	001536	447#		
EMV015	001546	449#		
EMV016	001556	451#		
EMV017	001566	453#		
EMV020	001576	455#		
EMV021	001606	457#		
EMV022	001616	459#		
EMV023	001626	461#		
EMV024	001636	463#		
EMV025	001646	465#		
EMV026	001656	467#		
EMV027	001666	469#		
EMV030	001676	471#		
EMV031	001706	473#		
EMV032	001716	475#		
EMV033	001726	477#		
EMV034	001736	479#		
EMV035	001746	481#		
EMV036	001756	483#		
EMV037	001766	485#		
EMV040	001776	487#		

```
EMV041  002006            489#
EMV042  002016            491#
EMV043  002026            493#
EMV044  002036            495#
EMV045  002046            497#
EMV046  002056            499#
EMV047  002066            501#
EMV050  002076            503#
EMV051  002106            505#
EMV052  002116            507#
EMV053  002126            509#
EMV054  002136            511#
EMV055  002146            513#
EMV056  002156            515#
EMV057  002166            517#
EMV060  002176            519#
EMV061  002206            521#
EMV062  002216            523#
EMV063  002226            525#
EMV064  002236            527#
EMV065  002246            529#
EMV066  002256            531#
EMV067  002266            533#
EMV070  002276            535#
EMV071  002306            537#
EMV072  002316            539#
EMV073  002326            541#
EMV074  002336            543#
EMV075  002346            545#
EMV076  002356            547#
EMV077  002366            549#
EMV100  002376            551#
EMV101  002406            553#
EMV102  002416            555#
EMV103  002426            557#
EMV104  002436            559#
EMV105  002446            561#
EMV106  002456            563#
EMV107  002466            565#
EMV110  002476            567#
EMV111  002506            569#
EMV112  002516            571#
EMV113  002526            573#
EMV114  002536            575#
EMV115  002546            577#
EMV116  002556            579#
EMV117  002566            581#
EMV120  002576            583#
ENPDCT  002776            650#   816
EOFLO = 000010           8598#  8608   8611
EREG0   002746            635#  9847*  11122  11133  11135  11139  11147  11149  11151  11153  11156  11157  11159
                         11161
EREG1   002750            636#  9848*  11122  11124  11130  11134  11141  11149  11151  11157  11159  11161
EREG2   002752            637#  9849*  11122  11124  11135  11149  11151  11153  11157  11159
EREG3   002754            638#  9850*  11122  11125  11128  11151  11153  11156  11157  11159
EREG4   002756            639#  9851*  11124  11128  11143  11159
```

```
EREG5   002760            640#  9852*  11128
EREG6   002762            641#  9853*   9854*
EREG7   002764            642#  9855*
ERRPNT= 104406             989   1158   1266   1292   1402   1647   1802   2038   2122   2232   2381   2413   2445
                          2477   2605   2641   2677   2713   2845   2881   2988   3084   3103   3122   3144   3165
                          3236   3256   3276   3298   3319   3410   3538   3644   3760   3901   3993   4021   4049
                          4077   4143   4167   4250   4446   4542   4654   4685   4716   4747   4778   4811   4975
                          5042   5063   5083   5102   5120   5180   5251   5322   5391   5462   5550   5637   5708
                          5778   5846   5939   6063   6176   6293   6688   6985   7108   7212   7255   7559   7851
                          6249   8302   8428   8554   8722   8868   9003  10350#
ERRVEC= 000004             129#   703    765    766*   777*  1253   1254   1259*  1264*  1290*  1315*  1316*  9783
                          9784*  9786*  9789*
EUFLO = 000012            8599#  8618   8621
EXEAEB  045002           11207  11280#
EXPRES  044710           11200  11266#
EXT001  025416            7901   8171#
EXT002  023474            7288   7433#
F     = 000000            6202#  6211   6217
FALU01  020532            6315   6611#
FEA     002616             602#  9192*  11113  11137
FEC     002614             601#  8736*   8831*  9017*  9163*  9195*  11113  11137
FIXFRA= 104423            5948   7226   7237   7571   7572   7864   7865  10363#
FPALTN  003024             667#  5183   5254   5325   5394   5465   6172   7217
FPALTP  003004             662#  4978   5178   5194   5249   5265   5320   5334   5389   5405   5460   5474   5553
                          5640   5711   5942   6066   6173   7216
FPANIM  003032             669#
FPANMS  003144             688#
FPANOO  003030             668#
FPAPAN  003074             678#  5202   5342   5413
FPAPIM  003012             664#
FPAPMS  003134             686#
FPAPOO  003010             663#  5273   5462
FPEM4Z  003154             690#  5646   5717
FPESCP  002620             604#  9213   9215   9756*  10451*  10454*
FPINST  002774             648#  9127*  11144
FPLENF  002624             608#  3394*  3522*  3632*  3746*  3982*  4229*  4276*  4349   5541*  9693*  9695*  9697*
                          9713  10072
FPMOAN  003022             666#
FPMOAP  003002             661#  3988   4044   4046   4074
FPMOIM  003124             684#
FPONES  003050             673#
FPPEND  030776            9214   9216#
FPPILT  030616             702   9155#
FPPNFP  030664            9160   9178#
FPPONE  003046             672#
FPPRTI  030760            9166   9176   9201   9212#
FPPVEC= 000244             193#   702
FPS     002612             600#  8735*  8750   8880*  8894   9016*  9031   9162*  9188*  11113  11137
FPSEFZ  003164             692#  3086   3095   3105   3114   3124   3136   3146   3157   3167   3177   3238   3248
                          3258   3268   3278   3290   3300   3311   3321   3331  '1218
FPSO  = 000000             248#
FPS1  = 147777             247#
FPTPOK  002622             606#  9165   9200   9212*  9757*  10452*  10455*
FPZEAN  003104             680#
FPZEAP  003070             677#  5044   5055   5065   5075   5085   5094   5104   5112   5122   5130
FPZERO  003060             675#  2021   3087   3088   3106   3107   3239   3240   3259   3260   3990   4016   4018
```

```
                        4072    4449    4545    4987    5105    5106    5123    5124    7229    7237
FPZOIM  003114           682#
FPOO11  003036           670#
FP1100  OG3054           674#
FSPDOI  015254          4831    4927#
FIN   = 140200           240#
FIP   = 040200           239#
GETBRK  005724          1784    1876#
GNS   = ****** U         277   10344   10345   10346   10347   10349   10350   10351   10352   10353   10354   10355   10356
                       10357   10358   10359   10360   10361   10362   10363   10364   10365   10366   10367   10368   10369
                       10370
GTEAEB  013334          4237    4295#
HACO    044572         11168   11171   11176   11185   11191   11194   11250#
HAC1    044604         11182   11194   11252#
HAC2    044616         11254#
HAC3    044630         11182   11256#
HAC4    044642         11258#
HAC5    044654         11260#
HT    = 000011            39#     711   10126   10167
IOTVEC= 000020           134#    747*    748*
LB    = 000377           242#    7594    9185
LF    = 000012            40#     714     719     726   10161   10167
LGN   = 177777           234#
LGP   = 077777           233#
LOGCAD= 000106           171#
LOGCAT= 000107           172#
LOGCUA= 000103           168#    1029
LOGFLG= 000104           169#    1034
LOGJAM= 000100           165#
LOGPBA= 000102           167#
LOGSVC= 000101           166#
LOGWHM= 000105           170#
LOOPNT= 104407         10351#
LPTITE  002645           621#     891    1169    1654    1813    2044    2138    2248    2388    2420    2452    2484    2616
                        2652    2688    2724    2856    2892    2993    3090    3109    3130    3151    3171    3243    3263
                        3284    3305    3325    3415    3543    3650    3770    3907    3998    4026    4054    4082    4147
                        4171    4255    4452    4548    4659    4690    4721    4752    4785    4818    4981    5050    5070
                        5089    5108    5126    5186    5257    5327    5397    5467    5556    5642    5713    5782    5852
                        5944    6068    6181    6298    6693    6990    7113    7220    7263    7565    7858    8255    8307
                        8434    8559    8732    8877    9013   9748*   9909*    9934
MAPADR  024752          7790    7916#
MAS   = 170007           2.9#    3905    5712    5786    5855    5943    9012
MED   = 076600           153#     888     890     897     901     937     939     996    1007    1029    1034    1060    1126
                        1132    1140    1167    1422    1431    1514    1557    1645    1668    1661    1819    1821    2048
                        2123    2135    2147    2239    2245    2257    3418    3546    3653    3773    4001    4029    4057
                        4085    4150    4174    4258    9065    9182    9184    9191    9193    9254    9256    9438    9441
                        9443    9444    9447    9449    9451    9752    9754    9911    9914    9920   10421   10428   10430
                       10532
MEMERR= 177744           209#   11116
MFACO   002646           625#   2368*   2377*   2378    2384    2393   2409*    2410    2416    2425   2441*    2442    2448
                        2457   2473*    2474    2480    2489   2592*   2601*    2602    2608    2621   2637*    2638    2644
                        2657   2673*    2674    2680    2693   2709*    2710    2716    2729   2832*   2841*    2842    2848
                        2861   2877*    2878    2884    2897   3089*   3093*   3094*    3095   3108*   3112*   3113*    3114
                       3133*   3134*   3135*    3136   3154*   3155*   3156*    3157    3174*   3175*   3176*    3177   3242*
                       3246*   3247*    3248   3262*   3255*    3267*    3268    3287*   3288*   3289*    3290   3308*   3309*
                       3310*    3311    3328*   3329*   3330*    3331   3400*   3401*    3407   3528*   3529*    3535   3638*
```

```
                    3639*  3647   3747*  3753*  3757   3989*  4017*  4045*  4073*  4246   4305*  4306*  4310*
                    4442   4450   4456   4536   4546   4552   4647   4657   4662   4678   4688   4693   4709
                    4719   4724   4740   4750   4755   4771   4781   4788   4804   4814   4821   4985*  4987
                    5190*  5194   5261*  5265   5330*  5334   5401*  5405   5470*  5474   5539*  5545*  5548
                    5645*  5646   5716*  5717   5773*  5774*  5781   5841*  5842*  5843*  5844*  5858   5925*
                    5934   5960   5965   6054*  6055*  6060*  6061   6084   6086*  6087   6166   6187   6283
                    6290   6674   6682   6970   6979*  6980*  7105*  7106*  7118   7251   7258   7505*  7548*
                    7556*  7557*  7562   7753*  7830*  7831*  7836*  7855   8230   8236   8289   8294   8410
                    8416   8535*  8546*  8551   8662   8666*  8676   8679   8681   8726   8727   8745   8760*
                    8790   8794*  8799   8825   8872   8889   8904*  8939   8943*  8955   8967   8986   8994
                    8995   9007   9008   9026   9046*  9047   11147  11166  11168  11171  11176  11180  11185
                   11188  11189  11191  11194  11197  11200  11204  11206  11207  11210  11212  11214  11215
                   11218
MFAC1    002656      626#  2386*  2391*  2393   2418*  2423*  2425   2450*  2455*  2457   2482*  2487*  2489
                    2614*  2619*  2621   2650*  2655*  2657   2686*  2691*  2693   2722*  2727*  2729   2854*
                    2859*  2861   2890*  2895*  2897   3402*  3403*  3408   3530*  3531*  3536   3748*  3755*
                    3758   3991*  4019*  4047*  4075*  4247   4302*  4313*  4314*  4451*  4456   4547*  4552
                    4658*  4662   4689*  4693   4720*  4724   4751*  4755   4784*  4788   4817*  4821   5053*
                    5054*  5055   5073*  5074*  5075   5092*  5093*  5094   5107*  5111*  5112   5125*  5129*
                    5130   5191*  5202   5262*  5273   5331*  5342   5402*  5413   5471*  5482   5540*  5547*
                    5561   5775*  5776*  5789   5856*  5858   5926*  5927*  5928*  5936*  5937*  5950   5962*
                    5963*  5964*  5966   6053*  6074   6088   6186*  6187   6291   6686   6977*  6978*  6983
                    7116*  7118   7223*  7226   7229   7269   7571   7574   7582   7585   7596   7607   7864
                    7867   7870   7874   8244   8277*  8273*  8279*  8280*  8283   8299   8326*  8423   8455*
                    8456*  8457*  8458*  8536*  8547*  8552   8663   8715   8728   8791   8858   8859   8863
                    8873   8940   8949   8950   8951   8952   9009   11147  11174  11176  11180  11185  11190
                   11191  11194  11197  11200  11204  11207  11210  11212  11215
MFAC2    002666      627#  3749*  3756*  3789   5559*  5561   5787*  5789   5947*  5948   5950   6071*  6072*
                    6074   6304   6698   6981*  6982*  6995   7224*  7227   7237   7266*  7269   7278   7572
                    7575   7592   7596   7619   7865   7867   7871   7876   8261   8271*  8272*  8273*  8274*
                    8275*  8313   8323*  8440   8450*  8451*  8452*  8453*  8537*  8548*  8566   8739   8745
                    8884   8889   9020   9026   11166  11174  11176  11182  11185  11194  11200  11207
MFAC3    002676      628#  3774*  3775*  3776*  3789   6301*  6304   6696*  6698   6993*  6995   7267*  7278
                    8240*  8295*  8420*  8564*  8566   8899   9036   11171  11176  11182  11194  11200  11207
MFAC4    002706      629#  8247*  8300*  8426*  8755   9041   11168  11182
MFAC5    002716      630#  8258*  8261   8310*  8313   8437*  8440   8825*  8829*  8830*  8852   8865*  8899
                    8997   9036   11171  11182
MFAC6    002726      631#  8671   8687*  8688*  8689*  8690*  8691*  8692*  8704*  8705*  8706*  8709   8712*
                    8713*  8717*  8719*  8755   8949*  8950*  8951*  8952*  8953*  8964*  9041   11168
MFAC7    002736      632#
MNS    = 170004      220#  5554   5641   5705   6067   6180   8238   8246   8418   8425   8876
MPP    = 170005      218#  7219   7262   7564   7857   8731
MO     = 100000      232#
M1     = 177777      231#   670    673    674   4564   4573   4582   4591
M2     = 177776      230#
NA     = 000000      249#
NEWPAS   003722      811#  9105
NOFPIE   002623      607#  2308*  9159   9767
NONE   = 000377     8600#  8604   8614   8624
NWPAS1   003335      726#   833
OFFCL1   025156     7798   8003#
OFFCL2   025164     7811   7816   8006#
OFFTBA   025210     8003   8017#
OLDPC    002766      645#  9125*  9156*  9225*  9270*  9288*  11113  11117  11144
OLDPS    002770      646#  9126*  9157*  9226*  9271*  9289*  11113  11117  11144
OLDSP    002772      647#  9124*  9155*  9224*  9269*  9287*  11113  11117  11144
```

```
PC     =%000007          60#  1784*  1900*  4237*  4241*  4300*  4308*  4316*  4353*  5923   6051   7790*  7798*
                       7811*  7816*  7946*  8014*  9090*  9093*  9099*  9104   9327*  9361*  9365*  9406*  9891*
                       9897*  10009* 10116* 10135* 10142* 10149* 10163* 10165* 10203* 10220* 10544
PIRQ   = 177772          46#
PIRQVE= 000240          140#
PR0    = 000000          63#
PR1    = 000040          64#
PR2    = 000100          65#
PR3    = 000140          66#
PR4    = 000200          67#
PR5    = 000240          68#   752   10358  10359  10360  10361  10362  10363  10364  10365  10366  10367  10368
                       10369  10370
PR6    = 000300          69#   706   1805
PR7    = 000340          70#   702   703    704    705    748    750    754    1413   10344  10345  10346  10347
                       10349  10350  10351  10352  10353  10354  10355  10356  10357
PS     = 177776          43#   [illegible]  1413*  1443*  1805*  1807*  1816*
PSW    = 177776          44#
PWRVEC= 000024          135#   753*   754*  10510* 10511* 10520* 10526* 10540* 10541*
P13Z   = 104210          241#
R      = 000000         6205#  6217   6220
RACX     044464        11190  11218  11236#
RACO     044476        11166  11180  11214  11238#
RAC1     044510        11171  11174  11176  11185  11188  11212  11240#
RAC1R3   045100        11215  11291#
RAC2     044522        11168  11174  11176  11182  11194  11242#
RAC3     044534        11191  11244#
RAC4     044546        11246#
RAC5     044560        11248#
RAND2    031252         9355   9365#
RANFPS= 104422         8664   8792   8941   10362#
RCEAEB   045016        11207  11282#
RCSPOO= 000100          188#  1168
RCVRES   044725        11200  11269#
RESTRT   003664          789# 10545
RESVEC= 000010          130#   704
RFEA   = 000076          180#  1821   9191
RFEC   = 000036          183#  1557   1661   1819   2048   2147   2257   3418   3546   3653   3773     01   4029
                        4057   4085   4150   4174   4258   9184   9193
RFLAG  = 000144          177#   901   1007   1140   9182   9254   9449   9752   9911   10421  10428
RFPA   = 000066          174#
RNGCOD   013440         4241   4341#
RSERVC= 000141          186#  1431
RWHAMI= 000022          161#   897    937
R0     =%000000          51#   789*   790*   791    795*   796    798    799    814*   815*   816    887*   889*
                         898*   899    902*   916    938*   995*  1008*  1011   1030*  1031*  1032   1035*  1038
                        1059*  1125*  1129*  1141   1161*  1162*  1163   1174   1271*  1275*  1278   1297*  1301*
                        1304   1386*  1419*  1446   1504*  1558   1644*  1657*  1662   1790*  1793*  1794   1795
                        1799*  1800*  1806   1820   2049   2128*  2134*  2149   2152   2238*  2244*  2259   2262
                        2373*  2405*  2437*  2469*  2597*  2633*  2669*  2705*  2837*  2873*  2983*  2984   2986*
                        2991   2997*  2999   3419   3547   3654   3778   3897*  3898*  3899*  3912   4002   4030
                        4058   4086   4151   4175   4259   4643*  4674*  4705*  4736*  4767*  4800*  6674*  6692
                        6970*  6989   7250*  7253*  7516*  7517*  7518*  7519   7522*  7523*  7524*  7525   7527*
                        7528*  7529*  7530   7532*  7533*  7534*  7535   7543*  7546   7577   7633*  7762*  7763*
                        7764*  7765*  7766*  7767   7769*  7770*  7771*  7772*  7773   7799   7803   7812   7817
                        7833*  7834*  7835*  7836   7842*  7845*  7855*  7856   7861*  7862*  7876*  7877*  7878
                        7934*  7938   8006*  8011*  8013*  8229*  8232   8288*  8291   8409*  8412   8542*  8550
```

```
                    8671*  8673*  8676*  8677*  8684*  8694   8703*  8707*  8709*  8711*  8713   8739*  8740*
                    8741*  8742*  8800*  8821*  8826   8828*  8829   8832*  8833*  8835   8837*  8844   8849*
                    8853*  8854   8884*  8885*  8886*  8958*  8959*  8962*  8963*  8964   8967*  8968   8969
                    8970   8971   8981*  8983*  8984   8997*  8998*  8999*  9000*  9001*  9020*  9021*  9022*
                    9023*  9064*  9096*  9099   9178   9183   9185*  9187   9192   9194*  9195   9198*  9253
                    9255*  9257*  9326   9328*  9329*  9330*  9332*  9334*  9335   9336*  9385   9388*  9391*
                    9395*  9398*  9401   9405*  9431   9434   9437*  9440*  9446*  9450*  9453*  9474   9476*
                    9512   9515*  9516   9518*  9519*  9521*  9522   9524*  9549   9554*  9555   9558   9563*
                    9564   9566   9568*  9530*  9584*  9594*  9605   9606*  9610*  9617   9618*  9625   9633*
                    9651   9654*  9657*  9659   9664   9668   9670   9674*  9699   9702*  9703   9704*  9708
                    9710   9716   9718   9723*  9753*  9847   9910   9912   9913*  9915*  9916*  9917*  9918
                    9919*  9921*  9953   9955*  9956*  9962*  9963*  9964*  9965*  9966*  9967   9973   9984
                    9993*  9995   10000  10008* 10041  10042* 10045  10058  10064  10069  10075  10078  10082*
                   10109  10110* 10115  10120  10123* 10180  10188* 10192  10193  10195* 10196* 10197  10219*
                   10324  10325* 10326  10327* 10328* 10329* 10330  10331  10332* 10420  10422* 10427  10429*
                   10431* 10512  10531* 10539*
R1      =%000001     52#    796*   798*   799*   894*   916    985*   995    1011   1038   1130*  1135*  1154*
                    1161   1258*  1278   1304   1641*  1662   1791*  1792*  1820*  1823   1825   2032*  2992*
                    2996*  2999   3896*  4233*  4271*  4272   4295   4297   4303   4311   4342   5697*  5725*
                    5929*  5930*  5935   6164*  6174   6675*  6708*  6971*  7005*  7104*  7112   7549*  7553
                    7578   7630*  7839   7916*  7917*  7918*  7919   7922*  7933*  7938*  7943   8009   8230*
                    8232*  8289*  8291*  8410*  8412*  8544*  8569   8672*  8674*  8678*  8692   8699*  8801*
                    8820*  8838*  8848*  8855   8955*  8956*  8957*  8966*  8978*  9179   9183*  9186*  9187*
                    9188   9197*  9386   9389*  9392*  9396*  9397*  9399*  9400*  9402   9404*  9513   9514*
                    9515   9522*  9523*  9550   9559   9569*  9584   9585*  9593*  9604   9605*  9611*  9619*
                    9626   9632*  9652   9653*  9654   9655*  9658*  9659*  9660*  9661*  9662*  9663*  9664*
                    9665*  9666*  9667*  9668*  9669*  9670*  9673*  9700   9703*  9708   9710   9716   9718
                    9722*  9848   9954   9984*  9986   9988   9995*  9997   10000*  10002  10007*  10181  10218*
                   10513  10538*
R2      =%000002     53#    895*   905    987*   993    1049   1156*  1174   1263*  1289*  1387*  1637*  1639
                    1761*  1776   1777   1778   1780*  2028*  2030   3910*  3911*  3912   4230*  4248   4278*
                    5698*  5704   5724*  5921*  5936   5963   6056*  6060   6086   6682*  6683*  7546*  7547*
                    7548   7554*  7555*  7556   7606*  7607*  7618*  7619*  7803*  7805   7812*  7814   7817*
                    7819   7827*  7828*  7829*  7830   7839*  7843*  7847*  7848*  7849*  7881   8007*  8011
                    8231*  8233*  8290*  8292*  8411*  8413*  8562*  8569   8679*  8680*  8690   8698*  8804*
                    8807*  8819*  8839*  8847*  8858*  8860   8968*  8972*  8973*  8974*  8981   8984*  8988*
                    8990*  8998   9387   9390*  9393*  9403*  9551   9560   9570*  9585   9586*  9592*  9603
                    9604*  9612*  9620*  9627   9631*  9849   10514  10537*
R3      =%000003     54#    899*   905    1032*  1049   1269*  1272*  1295*  1298*  1383*  1384   1398*  1505*
                    1507*  1629*  1640*  1644   1876*  1877*  1878*  1880*  1881*  1887*  1889*  1891   1895*
                    1897*  2031*  2032   2034*  2118*  2144   2226*  2254   3404*  3532*  3640*  3764*  3891*
                    3895   3896   3921*  3986*  4014*  4042*  4070*  4138*  4161*  4341*  4342   4344   4347
                    4348*  4351*  4352*  4442*  4443*  4444*  4536*  4537*  4538*  4539*  4540*  5922*  5937
                    5964   6284*  6288*  6681*  6684*  7553*  7557   7577*  7578*  7582   7592   7799*  7801
                    7807*  7808*  7809*  7822*  7823*  7824*  7825*  7826*  7831   7874*  7875*  7878*  7881
                    8004*  8006   8545*  8572   8681*  8687   8697*  8805*  8808*  8818*  8840*  8846*  8859*
                    8861   8969*  8975*  8999   9468   9472*  9474   9478*  9552   9561   9566*  9567   9568
                    9569   9570   9571*  9586   9587*  9591*  9603*  9613*  9621*  9628   9630*  9769*  9850
                    9922   9923*  9925*  10267  10276* 10282* 10283* 10286* 10291* 10292* 10293  10302* 10515
                   10536*
R4      =%000004     55#    997*   1001*  1003   1760*  1770   1774   1791   1847*  1848   1876   2020*  2028
                    2031   3081*  3093   3112   3134   3155   3175   3233*  3246   3266   3288   3309   3329
                    3396*  3429*  3524*  3557*  3634*  3663*  4437*  4463*  4531*  4559*  4647*  4648*  4650*
                    4651*  4652*  4678*  4679*  4681*  4682*  4683*  4709*  4710*  4712*  4713*  4714*  4740*
                    4741*  4743*  4744*  4745*  4771*  4772*  4774*  4775*  4776*  4804*  4805*  4807*  4808*
                    4809*  5039*  5054   5074   5093   5111   5129   5919*  5929   6048*  6049   6166*  6167*
```

```
                    6168*  6169*  6170*  6283*  6285*  6287*  6676*  6709*  6972*  7006*  7099*  7126*  7251*
                    7252*  7581*  7584*  7594*  7602   7616   7759*  7762   7769   7775   7782   7784   7896*
                    7916   8563*  8572   8799*  8800   8801   8807   8808   8810*  8823*  8843*  8850*  8852*
                    8854*  8855*  8860*  8861*  8970*  8976*  8994*  9000   9353   9354*  9357   9358*  9362*
                    9363*  9366*  9367*  9369*  9467   9470*  9471   9472   9476   9479*  9553   9555*  9562
                    9564*  9572   9578   9582*  9589*  9595*  9599   9601*  9608*  9614*  9624*  9625*  9626*
                    9627*  9628*  9629*  9851   10268  10270*  10271*  10272*  10273  10274*  10288  10290*  10298*
                   10301*  10516  10535*
R5      =%000005      56#   982*   985    987   1150*  1154   1156   1255*  1276   1302   1314   1508*  1555
                    1631*  1637   1640   1641   1770*  1771*  1772*  1773*  1776*  1777*  1781   1884   1893
                    2021*  2034   2114*  2118   2224*  2228   2372*  2377   2404*  2409   2436*  2441   2468*
                    2473   2596*  2601   2632*  2637   2668*  2673   2704*  2709   2836*  2841   2872*  2877
                    2979*  2983   3395*  3400   3402   3404   3523*  3528   3530   3532   3633*  3638   3640
                    3745*  3753   3755   3756   4438*  4443   4444   4532*  4537   4538   4539   4540   4642*
                    4648   4650   4651   4652   4673*  4679   4681   4682   4683   4704*  4710   4712   4713
                    4714   4735*  4741   4743   4744   4745   4766*  4772   4774   4775   4776   4799*  4805
                    4807   4808   4809   5538*  5545   5547   5916*  5919   5921   5922   5927   5928   6045*
                    6048   6055   6056   6160*  6164   6167   6168   6169   6170   6279*  6285   6287   6677*
                    6683   6973*  6977   6978   6979   6980   6981   6982   7100*  7104   7105   7106   7246*
                    7252   7285   7562*  7563   7568*  7569*  7585*  7586   7587*  7588   7638*  7639*  7840*
                    7846*  7926*  7930*  7934   7936   8003*  8007   8009   8534*  8542   8544   8545   8546
                    8547   8548   8971*  8977*  8995*  9001   9852   10003*  10042  10083*  10269  10275*  10277*
                   10279*  10280*  10281*  10282  10300*  10517  10534*
R6      =%000006      57#   741*   742*   743   9853
R7      =%000007      58#
S       = 000200    3433*  3442   3444   3445   3446   3447   3448   3449   3450   3451   3453   3455   3456
                    3457   3458   3459   3460   3461   3462   3561*  3568   3570   3572   3574   3667*  3675
                    3677   3678   3679   3681   3683   3684   3685   3687   3689   3690   3692
SDEADF  045032     11210  11284#
SETDW = 104410      992   1381   1627   1632   1797   2036   2131   2241   10352#
SETFP = 104412     1650   1808   2132   2242  10354#
SETUB = 104414      994   1408  10356#
SGLDAT= 104421    10361#
SMN   = 100200      236#
SMP   = 000200      235#
SP      =%000006     59#   745*   765*   773*   777    811*   820*   821*   834*   835*  1255   1276*  1302*
                   1314*  1508   1555*  1799   1806*  4347*  4352   7841*  7844*  7847   7849   8826*  8832
                    9112   9124   9125   9126   9127   9142*  9145   9155   9156   9157   9178*  9179*  9197
                    9198   9215*  9224   9225   9226   9253*  9257   9261*  9269   9270   9271   9287   9288
                    9289   9326*  9331*  9332*  9333   9336   9353*  9354   9357*  9358   9369   9370*  9385*
                    9386*  9387*  9403   9404   9405   9431*  9432*  9434*  9435*  9450   9453   9467*  9468*
                    9478   9479   9512*  9513*  9514   9523   9524   9525*  9549*  9550*  9551*  9552*  9553*
                    9554   9558*  9559*  9560*  9561*  9562*  9563   9567*  9624   9629   9630   9631   9632
                    9633   9634*  9651*  9652*  9653   9673   9674   9675*  9699*  9700*  9702   9706*  9721*
                    9722   9723   9724*  9775   9783*  9786   9788   9789   9818   9819   9823*  9855   9863
                    9868   9871   9874*  9877*  9878   9879*  9882   9906*  9910*  9913*  9919   9921   9922*
                    9925   9928*  9953*  9954*  9959*  9978*  9981*  9986*  10007  10008  10041*  10045*  10053*
                   10054*  10055*  10058*  10059*  10060*  10061*  10064*  10065*  10069*  10075*  10078*  10082  10109*
                   10110  10120*  10122  10123  10124*  10126  10128  10130  10136  10138*  10140*  10148*  10152
                   10156  10157  10161  10180*  10181*  10188  10189*  10200  10201*  10202*  10212  10213*  10218
                   10219  10259*  10260  10261  10262*  10267*  10268*  10269*  10275  10300  10301  10302  10303*
                   10304*  10323  10324*  10325  10330*  10331*  10332  10393  10395*  10420*  10424  10426*  10427*
                   10431  10454  10456*  10476  10477*  10488  10499  10512*  10513*  10514*  10515*  10516*  10517*
                   10518*  10519  10527*  10531  10533  10534  10535  10536  10537  10538  10539  10544*
SSEBSF  045045     11210  11286#
STACK = 001200       34#   323    745    811
```

```
START   003400            275     281     738#
STKLMT= 177774             45#
STPDCT  002610            598#     814
SUB64M= 104432           7867   10370#
SWR     001254            346#     743     767*    769     775*    782*    791     830    9082    9778    9792    9800    9807
                         9859    9866    9901    9904    9907   10046   10474   10518   10533*
SWREG   000176            279#     775
SW0    = 000001            98#
SW00   = 000001            88#      98
SW01   = 000002            87#      97
SW02   = 000004            86#      96
SW03   = 000010            85#      95
SW04   = 000020            84#      94
SW05   = 000040            83#      93    9907
SW06   = 000100            82#      92
SW07   = 000200            81#      91   10046
SW08   = 000400            80#      90
SW09   = 001000            79#      89
SW1    = 000002            97#
SW10   = 002000            78#    9082
SW11   = 004000            77#
SW12   = 010000            76#
SW13   = 020000            75#
SW14   = 040000            74#
SW15   = 100000            73#
SW2    = 000004            96#
SW3    = 000010            95#
SW4    = 000020            94#
SW5    = 000040            93#
SW6    = 000100            92#   10474
SW7    = 000200            91#
SW8    = 000400            90#
SW9    = 001000            89#
S0     = 000000          6715#    6721    6728    6732    6735    6739    6742    6749    6753    6758    6760    6763    6772
                         6774     6777    6795    6797    6799    6809    6811    6813    6840    6844    6849    6854    6858
                         6861     6865    6868    6872    6878    6880    6882    6892    6894    6896    6899    6901    6903
                         6906     6908    6910    7012#   7015    7016    7017    7021    7023    7028    7031    7032    7033
                         7035     7040    7045    7047    7048    7049    8595#   8604    8608    8611    8614    8618    8621
                         8624
S1     = 177400          6715#    6723    6725    6730    6737    6744    6746    6751    6756    6765    6767    6770    6779
                         6781     6788    6790    6792    6802    6804    6806    6816    6818    6820    6823    6825    6827
                         6830     6832    6834    6842    6847    6851    6856    6863    6870    6885    6887    6889    7012#
                         7019     7020    7024    7025    7027    7029    7036    7037    7039    7041    7043    7044    8596#
                         8604     8608    8614    8618    8621    8624
T      = 000040          6206#    6211    6214
TBITVE= 000014             131#
TKVEC  = 000060            138#
TPVEC  = 000064            139#
TRAPVE= 000034             137#     751*    752*
TRP004  031000             703    9224#
TRP010  031104             704    9269#
TRP114  031126             705    9287#
TRTVEC= 000014             132#
TST1    004026             831     884#
TST10   005206            1624#
TST11   005432            1638    1756#
```

TST12	006216	1852	2019#	
TST13	006400	2029	2112#	
TST14	006576	2169	2222#	
TST15	006750	2279	2303#	
TST16	006772	2365#		
TST17	007374	2475	2589#	
TST2	004124	913	917	933#
TST20	010016	2711	2829#	
TST21	010252	2879	2976#	
TST22	010362	2985	3078#	
TST23	010716	3230#		
TST24	011256	3391#		
TST25	011556	3431	3519#	
TST26	011732	3559	3629#	
TST27	012122	3665	3742#	
TST3	004152	980#		
TST30	012422	3804	3888#	
TST31	012510	3980#		
TST32	013040	4134#		
TST33	013174	4226#		
TST34	013604	4283	4434#	
TST35	013730	4464	4528#	
TST36	014170	4560	4637#	
TST37	015256	4928	4971#	
TST4	004350	986	1123#	
TST40	015324	4988	5036#	
TST41	015634	5175#		
TST42	015724	5203	5246#	
TST43	016014	5274	5317#	
TST44	016100	5343	5386#	
TST45	016170	5414	5457#	
TST46	016254	5483	5535#	
TST47	016420	5546	5632#	
TST5	004564	1155	1251#	
TST50	016464	5647	5694#	
TST51	016560	5770#		
TST52	016650	5790	5838#	
TST53	016736	5859	5912#	
TST54	017242	5920	6041#	
TST55	017472	6050	6158#	
TST56	017652	6165	6276#	
TST57	020534	6612	6671#	
TST6	004752	1379#		
TST60	022032	6914	6967#	
TST61	022340	7052	7096#	
TST62	022520	7127	7205#	
TST63	023474	7502#		
TST64	024226	7662	7750#	
TST65	025416	8225#		
TST66	026004	8332	8406#	
TST67	026212	8463	8532#	
TST7	005130	1500#		
TST70	026476	8543	8658#	
TST71	027064	8787#		
TST72	027506	8935#		
TST73	030132	9058#		

```
TYPE   = 104401          804   833   838  9084  9861  9873  9876  9881  9892  9969  9971  9972  9975
                        9977  9980  9983  9990  9992  9998 10006 10044 10057 10063 10068 10074 10077
                       10081 10131 10294 10344* 10542
TYPMAC  033674         10003 10041*
TYPOC  = 104402          9875  9880  9960  9979  9982  9987 10049 10070 10076 10079 10345*
TYPON  = 104404         10347*
TYPOS  = 104403           836 10056 10062 10066 10346*
UB     = 177400           243*  1141  7584  8677  9186  9194
UBCNTR  002630            612*  1407* 1450  9240* 11139
UBESCP  002626            611*  9259  9261  9751* 10418* 10424*
UBRKOK  031050           9242  9253*
UBRK04  031032           9229  9240*
UBTPOK  002625            610*  9241  9750* 10419* 10425*
VECDON  003716            797   801*
VECINT  003704            796*  800
VECTAB  003174            701*  795
WCSP00= 000300            189*  1131  1133
WFEA   = 000276           181*  9443
WFEC   = 000236           184*  9441
WFLAG  = 000344           178*   996  1060  1126  1422  1514  1645  1658  2129  2239  9256  9438  9451
                         9754  9914  9920 10430
WFPA   = 000266           175*  9444
WINIT  = 000352           155*   890  2135  2245  9065  9447
WNUA   = 000350           159*
WSR    = 000346           157*
WWHAMI= 000222            162*   888   939
X      = 000200          6715* 6721  6723  6725  6728  6730  6732  6735  6737  6739  6742  6744  6746
                         6749  6751  6753  6756  6758  6760  6763  6765  6767  6770  6772  6774  6777
                         6779  6781  6788  6790  6792  6795  6797  6799  6802  6804  6806  6809  6811
                         6813  6816  6818  6820  6823  6825  6827  6830  6832  6834  6840  6842  6844
                         6847  6849  6851  6854  6856  6858  6861  6863  6865  6868  6870  6872  6878
                         6880  6882  6885  6887  6889  6892  6894  6896  6899  6901  6903  6906  6908
                         6910  7012* 7015  7016  7017  7019  7020  7021  7023  7024  7025  7027  7028
                         7029  7031  7032  7033  7035  7036  7037  7039  7040  7041  7043  7044  7045
                         7047  7048  7049  8594* 8595  8596  8604  8608  8611  8614  8618  8621  8624
ZAPHFP= 104416           1317  1382  1406  1411  1461  1502  1567  1628  1636  1766  1787  1851  2027
                         2119  2168  2229  2278  2307  3405  3430  3533  3558  3641  3664  3763  3803
                         3985  4013  4041  4069  4095  4137  4160  4243  4282  8541 10358*
ZAPWFP= 104417           1256  1642  2125  2235 10359*
ZX1MN  = 100177           238*
ZX1MP  = 000177           237*
$APTHD  001000            303   309*
$ASTAT= ****** U        10206 10221
$AS64   031700           9556  9566*
$AS64I  031660           9558* 10368
$AS64M  031636           9549* 10369
$ATYC   034350          10177 10179*
$ATY1   034324          10175*
$ATY3   034332          10116 10176*
$ATY4   034342           9897 10178*
$AUTOB  001250            342*
$BDADR  001236            337*
$BDDAT  001242            339*
$BELL   001346            378*  9084  9861  9939
$CHARC  034320          10133* 10143* 10150 10159* 10164*
$CKSWR= ****** U        10348
```

```
$CLRDW  035234          10353  10390#
$CLRFP  035372          10355  10451#
$CLRUB  035304          10357  10418#
$CMPWD  032204           9693# 10367
$CMPX   032230           9694   9696  9699#
$CMP2W  032214           9695# 10366
$CMP4W  032222           9697# 10365
$CMTAG  001210            324#   740   741   749   755   758   759
$CM1   = 000020            360#   361#  362#  363#  364#  365#  366#  367#  368#  369#  370#  371#  372#
                           373#   374#  375#  376#
$CM2   = 000040            360#   361#  362#  363#  364#  365#  366#  367#  368#  369#  370#  371#  372#
                           373#   374#  375#  376#
$CM3   = 000020            358#   360
$CNDES  035426          10349  10474#
$CNT   = 000100           1902#  1909   1910#  1911#  1912#  1913#  1914#  1915#  1916#  1917#  1918#  1919#  1920#
                          1921#  1922#  1923#  1924#  1925#  1926#  1927#  1928#  1929#  1930#  1931#  1932#  1933#
                          1934#  1935#  1936#  1937#  1938#  1939#  1940#  1941#  1942#  1943#  1944#  1945#  1946#
                          1947#  1948#  1949#  1950#  1951#  1952#  1953#  1954#  1955#  1956#  1957#  1958#  1959#
                          1960#  1961#  1962#  1963#  1964#  1965#  1966#  1967#  1968#  1969#  1970#  1971#  1972#
                          1973#
$CPUOP  001404            401#
$CRLF   001353            380#   838   9881  9892  9939  9971  9977  9992  10006  10081  10132  10167
$D     = 000001              1   828   7752  7863  9067  9068  9112  10371  10480
$DEVCT  001366            392#
$DOAGN  030474           9092   9097  9103#
$DT     003236            712# 10068  10074  10077
$DUBL   031230           9357# 10360
$DW     035266          10392  10396#
$EADJ   031476           9467# 10364
$ENDAD  030464            289   9099#  9930
$ENDCT  030450            755   9094#
$ENV    001376            397#   780   9894  10111  10184  10208
$ENVM   001377            398# 10113  10118  10186
$EOP    030400           9067   9079#
$EOPCT  030442            755*  9091#  9095
$ERFLG  001214            328#   935*  2305*  9060*  9796  9798  9804*  9826  9856*  9939
$ERMAX  001230            334#   760*  9798  9821*  9826
$ERPNT  035452          10350  10488#
$ERROR  032724            749   9777  9845#
$ERRPC  001232            335#  9863*  9864*  9865  9871*  9872*  9874  9878*  9879  9939  9959  10011
$ERRTB  001406            412#  9966
$ERTTL  001224            332#  9862*  9939
$ESCAP  001344            377#   759*  9820*  9926  9928  9939  10476*
$ETABL  001376            396#
$ETEND  001406            315   408#
$FATAL  001360            389# 10212*
$FFLG   034570          10175* 10178*  10206  10215*  10223#
$FILLC  001276            356# 10136  10167
$FILLS  001275            355# 10167
$FIXFR  031564           9512# 10363
$FP     035424          10453  10457#
$FPBRK  001262            349#  9769  9923
$FPS    002610            599#  1794*  8730  8750  8802  8812  8816  8856  8875  8894  8992  9011  9031
                         9335*  9693  11128  11136  11137
$GDADR  001234            336#
$GDDAT  001240            338#
```

```
$GET42  030454          9096#
$GTSWR= ****** U       10348
$HD    = 000000          12
$HIBTS  001000          310#
$HINUM  031374          756*   9362    9367    9389    9397    9402*   9407#
$HT     003234          711#  10044
$ICNT   001216          329#   9811*   9812    9814*   9825
$ILLUP  035640         10510   10526   10547#
$INTAG  001251          343#
$ITEMB  001226          333#   9865*   9896    9939    9956
$LF     001354          381#   9939   10167
$LFLG   034567         10216*  10222#
$LONUM  031376          757*   9331    9363    9366    9388    9395    9401*   9408#
$LPADR  001220          330#    761*   9802*   9818*   9823    9825   10499*
$LPERR  001222          331#    762*   7645*   7653*   9802    9819*   9825    9906   10488*
$LPPNT  035460         10351   10499#
$LPTST  001260          348#   9794
$M     = 000001            1    208    9269
$MAIL   001356          311     315     387#    779    9817    9894   10111
$MBADR  001002          311#
$MFLG   034566         10176*  10182   10217*  10221#
$MSGAD  001372          394#  10192*  10195
$MSGLG  001374          395#  10197*
$MSGTY  001356          388#  10190   10198*  10210   10214*
$MXCNT  032722         9815    9825#
$NULL   001274          354#  10138   10167
$NWTST= 000001          839#    841     930#    946#    948    1080#   1082   1217#   1219   1321#   1323   1465#   1467
                       1570#   1572    1705#   1707   1976#   1978    2085#  2087    2195#  2197    2300#  2314#   2316
                       2532#   2534    2772#   2774   2941#   2943    3035#  3037    3187#  3189    3341#  3343    3466#
                       3468    3578#   3580    3696#  3698    3842#   3844   3926#   3928   4099#   4101   4185#   4187
                       4389#   4391    4482#   4484   4602#   4604    4934#  4936    4996#  4998    5140#  5142    5211#
                       5213    5282#   5284    5351#  5353    5422#   5424   5491#   5493   5596#   5598   5655#   5657
                       5729#   5731    5799#   5801   5867#   5869    5997#  5999    6113#  6115    6227#  6229    6617#
                       6619    6918#   6920    7056#  7058    7155#   7157   7437#   7439   7669#   7671   8175#   8177
                       8354#   8356    8475#   8477   8631#   8633    8764#  8766    8908#  8910    9055#
$OCNT   035014         10266*  10295*  10308#
$OMODE  035016         10261*  10265*  10270   10273*  10284*  10310#
$OVER   032706          9779    9795    9803    9813    9822#
$PASS   001364          391#    779*    834    9088*   9089*   9106    9328    9809   9826
$PASTM  001006          313#
$POWER  035646         10543   10550#
$PWRAD  035634         10545#
$PWRDN  035466           753   10510#  10540
$PWRMG  035630         10543#
$PWRUP  035540         10520   10526#
$QUES   001352          379#   9939   10167
$RAND   031276          9327    9361    9365    9384#
$RDCHR= ****** U       10348
$RDDEC= ****** U       10348
$RDLIN= ****** U       10348
$RDOCT= ****** U       10348
$REGAD  001300          358#
$REG0   001302          360#   1253*   1316    1425*   7535*  11126
$REG1   001304          361#   1254*   1315    7511*   7516    7532    7661*
$REG10  001322          368#   7507*   7629*   7756*   7892*
$REG11  001324          369#   7541*   7586*   7773*   7870    7871   11126
```

```
$REG12  001326            370#   7542*   7588*   7767*   7828   11126
$REG13  001330            371#   7537*   7600    7642*   7775*   7776*   8013
$REG14  001332            372#   7801*   7833    7835
$REG15  001334            373#   7805*   7824
$REG16  001336            374#   7814*   7826
$REG17  001340            375#   7819*   7822
$REG2   001306            362#   1440*   7514*   7522    7534    7657*
$REG3   001310            363#   7519*   7527    7547
$REG4   001312            364#   7525*   7555
$REG5   001314            365#   7530*   7574    7575
$REG6   001316            366#   7539*   7589*   7638    7780*   7893*   7922    7936   11126
$REG7   001320            367#   7540*   7598*   7639    7781*   7786*   7943   11126
$RNFPS  031150           9326#  10362
$RTNAD  030476           9105#
$R2A   = ******  U       10348
$SAVRE= ******  U        10348
$SAVR6  035644           10519*  10527   10528*  10529*  10549#
$SB64M  032112            9651#  10370
$SCOPE  032320             747    9746#
$SETDW  035246           10352   10393#
$SETFP  035404           10354   10454#
$SETUB  035330           10356   10424#
$SETUP= 000077             729#    746     747     749     751     753     755     756     758     759     761    9086    9747
                          9846    9904    9930
$SL     003240             713#  10057   10063
$SNGL   031220            9353#  10361
$STUP  = 177777            729#
$SVLAD  032652            9787    9816#
$SVPC  = 000204            287#    292
$SWR   = 167400              1#      12     376     377     378     758     759     761     762     885     934     981    1124
                          1252    1380    1501    1625    1757    2019    2113    2223    2304    2366    2590    2830    2977
                          3079    3231    3392    3520    3630    3743    3889    3981    4135    4227    4435    4529    4638
                          4972    5037    5176    5247    5318    5387    5458    5536    5633    5695    5771    5839    5913
                          6042    6159    6277    6672    6968    7097    7206    7503    7751    8226    8407    8533    8659
                          8788    8936    9059    9076    9087    9098    9104    9106    9738    9739    9740    9741    9742
                          9773    9790    9792    9793    9796    9797    9798    9805    9806    9807    9819    9822    9825
                          9837    9838    9839    9840    9841    9859    9866    9901    9904    9939   10546
$SWREG  001400             399#    782
$SWRMK= 000000            9742    9794
$TESTN  001362             390#   9817*  10010
$TIMES  001342             376#    758*    934*   1758*   2022*   2304*   4228*   5914*   6043*   7508*   7755*   8660*   8788*
                          8937*   9059*   9087*   9805*   9812    9815*   9825
$TKB    001266             351#
$TKS    001264             350#
$TN    = 000074              1#      12     839     885#    913     917     930     934#    946     981#    986    1080    1124#
                          1155    1217    1252#   1321    1380#   1465    1501#   1570    1625#   1638    1705    1757#   1852
                          1976    2019#   2029    2085    2113#   2169    2195    2223#   2279    2300    2304#   2314    2366#
                          2475    2532    2590#   2711    2772    2830#   2879    2941    2977#   2985    3035    3079#   3187
                          3231#   3341    3392#   3431    3466    3520#   3559    3578    3630#   3665    3696    3743#   3804
                          3842    3889#   3926    3981#   4099    4135#   4185    4227#   4283    4389    4435#   4464    4482
                          4529#   4560    4602    4638#   4928    4934    4972#   4988    4996    5037#   5140    5176#   5203
                          5211    5247#   5274    5282    5318#   5343    5351    5387#   5414    5422    5458#   5483    5491
                          5536#   5546    5596    5633#   5647    5655    5695#   5729    5771#   5790    5799    5839#   5859
                          5867    5913#   5920    5997    6042#   6050    6113    6159#   6165    6227    6277#   6612    6617
                          6672#   6914    6918    6968#   7052    7056    7097#   7127    7155    7206#   7437    7503#   7662
                          7669    7751#   8175    8226#   8332    8354    8407#   8463    8475    8533#   8543    8631    8659#
```

```
                        8764    8788#   8908    8936#   9055    9059#
$TPB    001272           353#  10156#  10167
$TPFLG  001277           357#  10105   10167
$TPS    001270           352#  10154   10167
$TRAP   035020           751   10323#
$TRP  = 000032         10336#  10345#  10346#  10347#  10349#  10350#  10351#  10352#  10353#  10354#  10355#  10356#  10357#
                       10358#  10359#  10360#  10361#  10362#  10363#  10364#  10365#  10366#  10367#  10368#  10369#  10370#
$TRPAD  035060         10330   10331   10343#
$TSTM   001004           312#
$TSTNM  001212           327#    789    9086#   9794    9816#   9817    9822    9826    9858    9939
$TYPBN= ****** U       10348
$TYPDS= ****** U       10348
$TYPE   034042         10105#  10203   10336   10344
$TYPEC  034254         10135   10142   10149   10154#  10155
$TYPER  033422          9891    9952#
$TYPEX  034322         10160   10162   10165#
$TYPOC  034616         10264#  10345
$TYPON  034632         10263   10266#  10347
$TYPOS  034572         10259#  10346
$UB     035362         10423   10430#
$UNIT   001370           393#
$UNITM  001010           314#
$USWR   001402           400#
$XTSTR  032456          9781#
$ZPHFP  031406          9434#  10358
$ZPWFP  031400          9431#  10359
$ZPXFP  031414          9433    9437#
$$GET4= 000000          9098#
$OFILL  035015         10260#  10264#  10274   10309#
$40CAT  000000           272#   9773    9868
.     = 045520           257#    273     277#    287     288#    290#    292#    293#    299     300#    302#    304#    323#
                         382     625#    626#    627#    628#    629#    630#    631#    632#    729#    744     761     762
                         821    1039    1050    1142    3096    3115    3137    3158    3178    3249    3269    3291    3312
                        3332    4003    4031    4059    4087    4152    4176    5056    5076    5095    5113    5131    8746
                        8751    8756    8890    8895    8900    9027    9032    9037    9042    9072#   9106    9825    9826
                        9939   10016#  10167   10224#  10522   10548
.$ASTA= ****** U       10176   10179
.$X   = 001000           299#    304
```

```
CHASE1   2085#  2195
CHASE2   2085#  2136   2246
CHASE3   2085#  2168   2278
CHASE4   2085#  2191   2292
COMENT    839#   841    946#   948   1080#  1082  1217#  1219  1321#  1323  1465#  1467  1570#  1572  1705#
         1707   1976#  1978  2085#  2087  2195#  2197  2314#  2316  2532#  2534  2772#  2774  2941#  2943
         3035#  3037   3187#  3189  3341#  3343  3466#  3468  3578#  3580  3696#  3698  3842#  3844  3926#
         3928   4099#  4101  4185#  4187  4389#  4391  4482#  4484  4602#  4604  4934#  4936  4996#  4998
         5140#  5142   5211#  5213  5282#  5284  5351#  5353  5422#  5424  5491#  5493  5596#  5598  5655#
         5657   5729#  5731  5799#  5801  5867#  5869  5997#  5999  6113#  6115  6227#  6229  6617#  6619
         6918#  6920   7056#  7058  7155#  7157  7437#  7439  7669#  7671  8175#  8177  8354#  8356  8475#
         8477   8631#  8633  8764#  8766  8908#  8910
COMMEN    141#
ENDCOM    141#
EOPMAC   9072#  9080
ERROR      35#   907    918  1013  1020  1040  1051  1144  1177  1280  1306  1390  1453  1561  1664
         1828   1838   2051  2154  2160  2264  2270  2395  2427  2459  2491  2623  2659  2695  2731
         2863   2899   3001  3097  3116  3138  3159  3179  3250  3270  3292  3313  3333  3421  3549
         3656   3780   3791  3914  4004  4032  4060  4088  4153  4177  4261  4458  4554  4664  4695
         4726   4757   4790  4823  4989  5057  5077  5096  5114  5132  5196  5204  5267  5275  5336
         5344   5407   5415  5476  5484  5563  5648  5719  5791  5860  5952  6076  6189  6306  6700
         6997   7120   7231  7239  7271  7280  7609  7621  7646  7884  8263  8315  8442  8575  8747
         8752   8757   8891  8896  8901  9028  9033  9038  9043  9133  9168  9203  9231  9244  9273
         9291
ER.PNT     1#    988   1157  1265  1291  1401  1646  1801  2037  2121  2231  2380  2412  2444  2476
         2604   2640   2676  2712  2844  2880  2987  3094  3103  3122  3144  3165  3236  3256  3276
         3298   3319   3409  3537  3643  3759  3900  3993  4021  4049  4077  4143  4167  4249  4445
         4541   4653   4684  4715  4746  4777  4810  4974  5042  5063  5083  5102  5120  5179  5250
         5321   5390   5461  5549  5636  5707  5777  5845  5938  6062  6175  6292  6687  6984  7107
         7211   7254   7558  7850  8248  8301  8427  8553  8721  8867  9002
ESCAPE    141#
GETPRI    141#
GETSWR    141#
IFORKO   1902#  1909   1910  1911  1912  1913  1917  1921  1925  1929  1933  1937  1941  1945  1949
         1953   1957   1961  1965  1969
LP.TIT     1#    891   1169  1653  1812  2044  2137  2247  2388  2420  2452  2484  2616  2652  2688
         2724   2856   2892  2993  3090  3109  3130  3151  3171  3243  3263  3284  3305  3325  3415
         3543   3650   3770  3907  3998  4026  4054  4082  4147  4171  4255  4452  4548  4659  4690
         4721   4752   4785  4818  4981  5050  5070  5039  5108  5126  5186  5257  5327  5397  5467
         5556   5642   5713  5782  5852  5944  6068  6181  6298  6693  6990  7113  7220  7263  7565
         7858   8255   8307  8434  8559  8732  8877  9013
MULT      141#
NEWTST    141#   839    930   946  1080  1217  1321  1465  1570  1705  1976  2085  2195  2300  2314
         2532   2772   2941  3035  3187  3341  3466  3578  3696  3842  3926  4099  4185  4389  4482
         4602   4934   4996  5140  5211  5282  5351  5422  5491  5596  5655  5729  5799  5867  5997
         6113   6227   6617  6918  7056  7155  7437  7669  8175  8354  8475  8631  8764  8908  9055
POP       141#  9403  10218 10219 10533 10534
PUSH      141#  9384  10179 10181 10202 10512 10518
REPORT    141#
SCOPE      36#   884    933   980  1123  1251  1379  1500  1624  1756  2018  2112  2222  2303  2365
         2589   2829   2976  3078  3230  3391  3519  3629  3742  3888  3980  4134  4226  4434  4528
         4637   4971   5036  5175  5246  5317  5386  5457  5535  5632  5694  5770  5838  5912  6041
         6158   6276   6671  6967  7096  7205  7502  7750  8225  8406  8532  8658  8787  8935  9058
         9080
SETPRI    141#
SETTRA  10336#  10345  10346 10347 10349 10350 10351 10352 10353 10354 10355 10356 10357 10358 10359
```

```
            10360   10361   10362   10363   10364   10365   10366   10367   10368   10369   10370
SETUP         1#    141#    738
SKIP        141#    913     917     986     1155    1636    1852    2029    2169    2279    2475    2711    2879    2985    3431
            3559    3665    3804    4283    4464    4560    4927    4968    5203    5274    5343    5414    5483    5546    5647
            5790    5859    5920    6050    6165    6611    6913    7051    7127    7662    8332    8463    8543
SLASH       141#    785     787     806     808     1063    1185    1516    1547    1675    1854    1867    2058    2500    2740
            2909    3009    3435    3561    3668    3807    4286    4332    4385    4833    4930    5571    5969    6091    6198
            6712    7009    7129    7903    7990    8034    8070    8099    8128    8150    8170    8334    8343    8465    8474
            8590    8630    9747    9749    9755    9758    9766    9772
SPACE       141#
STARS       141#    285     296     298     305     319     382     385     730     734     839     883     926     930     932
            941     946     979     1080    1122    1217    1250    1321    1378    1465    1499    1570    1623    1705    1755
            1976    2017    2085    2111    2195    2221    2296    2300    2302    2311    2314    2364    2532    2588    2772
            2828    2941    2975    3035    3077    3187    3229    3341    3390    3466    3518    3578    3628    3696    3741
            3842    3887    3926    3979    4099    4133    4185    4225    4389    4433    4482    4527    4602    4636    4934
            4970    4996    5035    5140    5174    5211    5245    5282    5316    5351    5385    5422    5456    5491    5534
            5596    5631    5655    5693    5729    5769    5799    5837    5867    5911    5997    6040    6113    6157    6227
            6275    6617    6670    6918    6966    7056    7095    7155    7204    7437    7501    7669    7749    8175    8224
            8354    8405    8475    8531    8631    8657    8764    8786    8908    8934    9052    9055    9057    9069    9074
            9108    9151    9220    9265    9283    9300    9303    9340    9375    9411    9457    9494    9529    9532    9638
            9679    9728    9731    9735    9829    9833    9942    10019   10086   10090   10170   10174   10232   10236   10313
            10317   10373   10400   10434   10459   10480   10491   10503   10504   10508   10524   10554
SWRSU       141#    763#
TYPBIN      141#
TYPDEC      141#
TYPNAM      141#
TYPNUM      141#
TYPOCS      141#
TYPOCT      141#    9874    9879
TYPTXT      141#
UPCODE        1#    10531
X          6323#    6328    6333    6342    6347    6352    6361    6366    6371    6380    6385    6390    6399    6404    6409
            6418    6423    6428    6437    6442    6447    6456    6461    6466    6475    6480    6485    6494    6499    6504
            6513    6518    6523    6532    6537    6542    6551    6556    6561    6570    6575    6580    6589    6594    6599
            7292#   7339    7343    7347    7351    7357    7361    7365    7369    7373    7377    7381    7385    7389    7393
            7397    7401    7405    7409    7413    7417    7421    7425
XX         6323#    6326    6331    6336    6345    6350    6355    6364    6369    6374    6383    6388    6393    6402    6407
            6412    6421    6426    6431    6440    6445    6450    6459    6464    6469    6478    6483    6488    6497    6502
            6507    6516    6521    6526    6535    6540    6545    6554    6559    6564    6573    6578    6583    6592    6597
            6602
$IFKO      1902#    1909    1910    1911    1912    1913    1914    1915    1916    1917    1918    1919    1920    1921    1922
            1923    1924    1925    1926    1927    1928    1929    1930    1931    1932    1933    1934    1935    1936    1937
            1938    1939    1940    1941    1942    1943    1944    1945    1946    1947    1948    1949    1950    1951    1952
            1953    1954    1955    1956    1957    1958    1959    1960    1961    1962    1963    1964    1965    1966    1967
            1968    1969    1970    1971    1972
$$CMRE      317#    360     361     362     363     364     365     366     367     368     369     370     371     372     373
            374     375
$$CMTM      317#
$$ESCA      141#
$$NEWT      141#    839     930     946     1080    1217    1321    1465    1570    1705    1976    2085    2195    2300    2314
            2532    2772    2941    3035    3187    3341    3466    3578    3696    3842    3926    4099    4185    4389    4482
            4602    4934    4996    5140    5211    5282    5351    5422    5491    5596    5655    5729    5799    5867    5997
            6113    6227    6617    6918    7056    7155    7437    7669    8175    8354    8475    8631    8764    8908    9055
$$SET     10336#   10345   10346   10347   10349   10350   10351   10352   10353   10354   10355   10356   10357   10358   10359
            10360   10361   10362   10363   10364   10365   10366   10367   10368   10369   10370
$$SETM      779#
```

```
$$SKIP    141#    913    917    986   1155   1638   1852   2029   2169   2279   2475   2711   2879   2985   3431
         3559    3665   3804   4283   4464   4560   4928   4988   5203   5274   5343   5414   5483   5546   5647
         5790    5859   5920   6050   6165   6612   6914   7052   7127   7662   8332   8463   8543
.EQUAT      1#     31
.HEADE      1#      2
.SETUP      1#    729
.SKIP3      1#   1214   1318   1702   1973   2082   2529   2769   2938   3032   3184   3338   3463   3575   3693
         3839    3923   4096   4182   4386   4479   4599   4931   4993   5137   5208   5279   5348   5419   5488
         5593    5652   5726   5796   5864   5994   6110   6224   6614   6915   7053   7152   7434   7666   8172
         8351
.$ACT1      1#    283
.$APTB      1#    383#
.$APTH      1#    294
.$APTY      1#  10172
.$CMTA      1#    317
.$EOP       1#   9072
.$ERRO      1#   9831
.$POWE      1#  10506
.$RAND      1#   9373
.$SCOP      1#   9733
.$TRAP      1#  10317
.$TYER      1#   9942
.$TYPE      1#  10088
.$TYPO      1#  10234
.$4OCA      1#    255
```

```
ABSF    3649
ADC     7844  8688  8689  8691  9396  9399
ADD     2996  2997  3911  4314  7527  7528  7771  7849  8687  8690  8692  8829  9046  9370  9395
        9397  9398  9400  9519  9525  9589  9601  9634  9657  9658  9663  9667  9675  9724  9854
        9966 10124 10189 10201 10213 10262 10272 10395 10426 10456 10477
ADDD    6297  6692
ADDF    3996  4024  4146  4170  4254
ASH     1030  1771  1880  7533  7547  7823  7825  7834  8711  8828  8956  8963  8983  9329
ASHC    7555  7828
ASL     7517  7518  7523  7524  7764  7765  7770  7807  7808  7848  7918  8697  8704  8846  9391
        9591  9963  9964  9965 10059 10328 10329
ASR     1877  1895  7763  7843  7917  8323  8326  8684  8837  8974  9610 10196
ASRB    8271  8277  8450  8455
BCC     1879  8330  8461  8685
BCS     1785  1896  4239  7791
BEQ      797   906   917   986  1004  1012  1039  1050  1142  1175  1279  1305  1451  1559  1638
        1663  1775  1824  1894  2029  2050  2145  2150  2153  2255  2269  2263  2379  2394  2411
        2426  2443  2458  2475  2490  2603  2622  2639  2658  2675  2694  2711  2730  2843  2862
        2879  2898  2985  3000  3096  3115  3137  3158  3178  3249  3269  3291  3312  3332  3420
        3548  3655  3779  3790  3913  4003  4031  4059  4087  4152  4176  4260  4277  4457  4553
        4663  4694  4725  4756  4789  4822  4988  5056  5076  5095  5113  5131  5195  5203  5266
        5274  5335  5343  5406  5414  5475  5483  5562  5647  5718  5790  5859  5951  6050  6075
        6188  6286  6305  6699  6996  7119  7230  7238  7270  7279  7583  7593  7597  7603  7617
        7640  7882  7935  7937  7944  8008  8010  8262  8284  8314  8441  8573  8695  8746  8751
        8756  8817  8836  8890  8895  8900  9027  9032  9037  9042  9097  9141  9160  9214  9260
        9473  9475  9574  9590  9609  9768  9793  9795  9797  9801  9810  9857  9860  9905  9908
        9927  9968  9974  9985  9989  9994  9996 10001 10047 10114 10127 10162 10183 10187 10207
       10209 10289 10475
BGE     1849  3922  4296  7631  7634  7894  9117  9813
BGT     1779  1892  4304  4312  7601  9092  9602 10296
BHI     9121  9776  9799  9869
BIC      898  1031  1772  1816  1881  2391  2423  2455  2487  2619  2655  2691  2727  2859  2895
        3093  3112  3134  3155  3175  3246  3266  3288  3309  3329  3775  5054  5074  5093  5111
        5129  5962  5963  5964  6072  6086  7776  7809  7829  7875  7877  8275  8677  8680  8712
        8717  8830  8833  8853  8943  8953  8957  8972  8988  9089  9185  9186  9194  9330  9332
        9518  9521  9706  9753  9916 10055 10065 10286 10422
BICB    1777  7588
BIS      938  1162  1413  1773  1805  1889  4278  4352  5936  5937  6060  7534  7584  7594  7639
        7824  7826  7835  7878  7938  8011  8013  8713  8719  8854  8865  8964  8973  8984  8990
        9187  9255  9333  9334  9450  9721  9917 10291 10292 10429
BISB    1776  7586  7607  7619  9956
BIT      830  1141  1774  1884  1893  5960  6084  7936  7943  8009  8283  8812  8816  8835  9047
        9082  9228  9474  9516  9778  9792  9800  9807  9859  9866  9904  9907 10046 10474
BITB   10113 10118 10150 10186
BLE     4273  4298  7785  9600
BLOS     817
BLT      792  4343  7658  7662  7783  7897  9579  9583 10141 10297
BMI     1155  3754  4649  4680  4711  4742  4773  4806  5546  5920  6165  7643  7800  7804  7813
        7818  8543  8716  8803  8845  8857  8987  8993  9573
BNE      744   770   781   831   892  1170  1447  1655  1814  1826  1885  2045  2139  2249  2389
        2421  2453  2485  2617  2653  2689  2725  2857  2893  2994  3091  3110  3131  3152  3172
        3244  3264  3285  3306  3326  3416  3544  3651  3771  3908  3999  4027  4055  4083  4148
        4172  4256  4350  4453  4549  4660  4691  4722  4753  4786  4819  4982  5051  5071  5090
        5109  5127  5187  5258  5328  5398  5468  5557  5643  5714  5783  5853  5945  5961  6069
        6085  6182  6299  6694  6991  7114  7221  7264  7566  7859  8256  8308  8435  8560  8567
        8570  8733  8761  8813  8878  8905  9014  9048  9083  9113  9131  9166  9201  9229  9242
```

	9394	9517	9709	9711	9717	9719	9779	9808	9867	9895	9931	9935	9957	10112	10119
	10121	10129	10137	10151	10158	10185	10191	10194	10211	10287	10530				
BPL	7286	7605	8864	9714	9902	10073	10106	10155	10285						
BR	273	772	800	913	1017	1024	1046	1061	1183	1273	1299	1388	1673	1836	1852
	2056	2159	2165	2169	2269	2275	2279	2401	2433	2465	2497	2629	2665	2701	2737
	2869	2905	3006	3431	3559	3665	3800	3804	4279	4283	4345	4464	4560	4670	4701
	4732	4763	4796	4829	4928	5569	5967	6089	6195	6313	6612	6914	7052	7127	7235
	7654	7923	7927	7939	8012	8332	8463	8587	8700	8718	8806	8841	8989	9143	9176
	9355	9433	9520	9556	9581	9588	9596	9607	9615	9694	9696	9781	9787	9790	9803
	9806	9883	9900	9961	9991	10005	10050	10108	10134	10144	10153	10160	10177	10199	10263
	10278	10299	10392	10423	10453	10522	10548								
CFCC	1164	2042													
CLC	1883	1899	4307	4315	7942	8811	8822								
CLR	742	758	759	779	790	815	820	934	935	997	1059	1129	1259	1269	1271
	1295	1297	1407	1504	1505	1657	2304	2305	2368	2373	2592	2597	2832	3094	3113
	3135	3156	3176	3247	3267	3289	3310	3330	3401	3403	3529	3531	3639	3747	3748
	3749	3776	3897	4233	4313	4351	4643	5539	5540	5774	5776	5842	5847	5944	7507
	7539	7540	7541	7554	7581	7606	7618	7756	7781	7827	7841	7933	8004	8035	8536
	8537	8673	8678	8804	8805	8958	9059	9060	9062	9086	9087	9432	9580	9587	9751
	9756	9760	9761	9762	9763	9804	9805	9820	9846	9913	9955	10276	10390	10418	10451
	10528														
CLRB	902	1008	1035	1443	1807	3394	3522	3632	3746	3982	4229	5541	8666	8794	9212
	9695	9748	9750	9757	9759	10060	10133	10159	10215	10216	10217	10391	10419	10452	
CLRD	3904	4980	5046	5047	5048	5067	5068	5087	5635	5702	5703	5849	5926	6054	6179
	7215	7259	7260	8237	8245	8417	8424								
CLRF	1296	3126	3127	3128	3148	3149	3169	3280	3281	3282	3302	3303	3323	5785	7111
CMP	743	769	791	816	905	916	1011	1038	1049	1174	1278	1304	1446	1450	1558
	1662	1823	1825	1848	1891	2049	2144	2149	2152	2254	2259	2262	2378	2393	2410
	2425	2442	2457	2474	2489	2602	2621	2638	2657	2674	2693	2710	2729	2842	2861
	2878	2897	2984	2999	3419	3547	3654	3778	3789	3912	4002	4030	4058	4086	4151
	4175	4259	4272	4295	4297	4303	4342	4344	5561	6049	7782	7784	7881	8569	8572
	8750	8894	9031	9112	9120	9578	9599	9708	9710	9716	9718	9775	9788	9794	9798
	9812	9868	9882	9930	10323										
CMPB	780	7582	7592	7596	9894	10111	10126	10128	10136	10157	10161	10184			
CMPF	3414														
COM	1001	1272	1298												
COMB	4276	7587													
DEC	1847	3921	5724	5930	7630	7633	7642	7657	7661	7893	7896	9090	9116	9595	9962
DECB	10140	10143	10284	10295											
DIV	3898	8959													
DIVF	6989														
EMT	35														
HALT	272	9903	9932	10107	10521	10547									
INC	835	1134	1153	1635	1764	2026	2117	2227	2376	2405	2408	2437	2440	2469	2472
	2600	2633	2636	2669	2672	2705	2708	2840	2873	2876	2982	3399	3527	3637	3752
	3894	3899	4236	4271	4441	4535	4646	4674	4677	4705	4708	4736	4739	4767	4770
	4800	4803	5544	5701	5933	6059	6163	6282	6680	6708	6976	7005	7103	7249	7552
	7589	7598	7629	7789	7892	8243	8298	8540	8670	8798	8948	8962	8966	9088	9240
	9393	9614	9811	9816	9856	9862	10214	10290	10298	10529					
INCB	8760	8904	10163												
IOT	36	277													
JMP	281	4831	6315	6710	7007	7288	7653	7663	7898	7901	7919	9067	9104	9777	
JSR	1784	4237	4241	7790	7798	7811	7816	9099	9327	9361	9365	9891	9897	10003	10116
	10135	10142	10149	10203											
LDCDF	5396														
LDCFD	5466														

LDCIF 7112

LDD 4545 4546 4657 4688 4719 4750 4781 4783 4814 4816 4978 4984 5044 5049 5065
5069 5085 5088 5104 5105 5106 5122 5123 5124 5178 5183 5249 5254 5320 5325
5389 5394 5460 5465 5640 5711 5923 5934 5942 6051 6061 6066 6172 6173 6290
6291 6686 7216 7217 7258 7563 7856 8236 8244 8294 8299 8416 8423 8726 8727
8728 8872 8873 9007 9008 9009

LDEXP 2991 3895 5704 5850 5935

LDF 2384 2416 2448 2480 2608 2612 2644 2648 2680 2684 2716 2720 2848 2852 2884
2888 3086 3087 3088 3105 3106 3107 3124 3129 3146 3150 3167 3170 3238 3239
3240 3258 3259 3260 3278 3300 3321 3407 3408 3535 3536 3647 3757 3758 3988
3990 4016 4018 4044 4046 4072 4074 4140 4141 4163 4164 4246 4247 4449 4450
5185 5548 5553 5781 6983 8551 8552

LDFPS 1138 1257 1384 1416 1435 1509 1551 1643 1795 2035 2126 2236 2367 2591 2831
2978 3080 3232 3393 3413 3521 3541 3631 3648 3744 3766 3890 3992 4020 4048
4076 4142 4166 4248 4436 4530 4639 4973 5038 5177 5248 5319 5388 5459 5537
5634 5696 5772 5840 5915 6044 6171 6174 6278 6673 6969 7098 7210 7506 7754
8226 8407 8550 8725 8730 8738 8871 8875 8883 9006 9011 9019 9442 9918

LDUB 999 1385 1410 1506 1511 1630 1786 2033 2120 2230 3406 3534 3642 3765 3987
4015 4043 4071 4139 4162 4245 9770 9924

MOV 741 745 747 748 749 750 751 752 753 754 755 756 757 760 761
762 765 766 767 768 773 775 776 777 782 789 795 796 798 799
811 814 821 825 826 827 834 887 889 894 895 899 982 985 987
993 995 1032 1125 1130 1131 1150 1154 1156 1161 1163 1253 1254 1255 1258
1263 1264 1275 1276 1289 1290 1301 1302 1314 1315 1316 1383 1397 1398 1419
1507 1508 1555 1629 1631 1637 1639 1640 1641 1644 1758 1759 1760 1761 1770
1780 1781 1790 1791 1794 1799 1806 1820 1876 1897 2020 2021 2022 2023 2028
2030 2031 2032 2034 2114 2118 2128 2134 2224 2228 2238 2244 2372 2377 2404
2409 2436 2441 2468 2473 2596 2601 2632 2637 2668 2673 2704 2709 2836 2837
2841 2872 2877 2979 2983 3081 3233 3395 3396 3400 3402 3404 3523 3524 3528
3530 3532 3633 3634 3638 3640 3745 3753 3755 3756 3764 3891 3896 3986 4014
4042 4070 4138 4161 4228 4230 4302 4305 4310 4341 4347 4348 4437 4438 4442
4443 4464 4531 4532 4536 4537 4538 4539 4540 4642 4647 4648 4650 4651 4652
4673 4678 4679 4681 4682 4683 4704 4709 4710 4712 4713 4714 4735 4740 4741
4743 4744 4745 4766 4771 4772 4774 4775 4776 4799 4804 4805 4807 4808 4809
5039 5538 5545 5547 5697 5698 5773 5775 5841 5914 5916 5919 5921 5922 5927
5928 5929 6043 6045 6048 6055 6056 6160 6164 6166 6167 6168 6169 6170 6279
6283 6284 6285 6287 6674 6675 6676 6677 6681 6682 6683 6970 6971 6972 6973
6977 6978 6979 6980 6981 6982 7099 7100 7104 7105 7106 7246 7250 7251 7252
7505 7508 7511 7514 7516 7519 7522 7525 7530 7532 7535 7537 7542 7543 7546
7548 7549 7553 7556 7557 7562 7577 7638 7645 7653 7753 7755 7759 7762 7767
7769 7773 7775 7780 7786 7801 7805 7814 7819 7822 7830 7831 7833 7836 7839
7840 7842 7847 7855 7874 7876 7916 7922 7926 7930 7934 8003 8006 8007 8229
8230 8231 8232 8288 8289 8290 8291 8409 8410 8411 8412 8534 8542 8544 8545
8546 8547 8548 8660 8671 8672 8676 8679 8681 8703 8709 8739 8788 8799 8870
8901 8807 8808 8810 8825 8826 8832 8843 8852 8855 8858 8859 8860 8861 8884
8937 8949 8950 8951 8952 8955 8967 8968 8969 8970 8971 8981 8994 8995 8997
8998 8999 9000 9001 9020 9064 9093 9096 9124 9125 9126 9127 9142 9145 9146
9147 9155 9156 9157 9178 9179 9183 9188 9192 9195 9197 9198 9215 9224 9225
9226 9253 9257 9261 9269 9270 9271 9287 9288 9289 9326 9331 9335 9336 9353
9354 9357 9358 9362 9363 9366 9367 9369 9385 9386 9387 9388 9389 9390 9401
9402 9403 9404 9405 9431 9434 9435 9437 9440 9446 9453 9467 9468 9470 9472
9476 9478 9479 9512 9513 9514 9515 9522 9523 9524 9549 9550 9551 9552 9553
9554 9555 9556 9559 9560 9561 9562 9563 9564 9566 9567 9568 9569 9570 9571
9584 9585 9586 9603 9604 9605 9624 9625 9626 9627 9628 9629 9630 9631 9632
9633 9651 9652 9653 9654 9655 9673 9674 9699 9700 9702 9703 9704 9722 9723
9764 9765 9769 9783 9784 9786 9789 9802 9814 9815 9817 9818 9819 9821 9822

```
        9823  9847  9848  9849  9850  9851  9852  9853  9855  9858  9863  9871  9874  9878  9879
        9906  9910  9912  9919  9921  9922  9923  9925  9928  9936  9953  9954  9959  9967  9973
        9978  9981  9984  9986  9993  9995  9997 10000 10002 10007 10008 10041 10042 10045 10058
       10064 10069 10075 10078 10082 10109 10110 10115 10123 10138 10180 10181 10188 10192 10197
       10198 10200 10202 10212 10218 10219 10259 10267 10268 10269 10275 10282 10300 10301 10302
       10303 10307 10324 10325 10330 10331 10332 10393 10396 10397 10420 10424 10427 10431 10454
       10476 10488 10499 10510 10511 10512 10513 10514 10515 10516 10517 10518 10519 10520 10526
       10527 10531 10533 10534 10535 10536 10537 10538 10539 10540 10541 10544
MOVB    1878  2308  7585  7799  7803  7812  7817  9328  9693  9697  9865  9896  9909 10120 10148
       10156 10175 10176 10178 10260 10261 10264 10265 10266 10270 10273 10274 10293 10327 10394
       10425 10455
MUL     7578
MULD    8253  8306  8432
MULF    3542  3768  8558
NEG     7529  7772 10271
NOP     1652  1810  2043  9100  9101  9102
RESET   9098
ROL     1887  8698  8699  8705  8706  8818  8819  8820  8821  8847  8848  8849  9392  9592  9593
        9594 10053 10054 10277 10279 10280 10281 10283
ROR     1792  8272  8273  8274  8278  8279  8280  8451  8452  8453  8456  8457  8458  8838  8839
        8840  8975  8976  8977  9611  9612  9613
RORB    1793
RTI      774  9148  9216  9238  9251  9262  9280  9298  9337  9371  9454  9480  9526  9635  9676
        9725  9824  9938 10125 10305 10333 10398 10432 10457 10478 10489 10500 10546
RTS     1900  4300  4308  4316  4353  7946  8014  9406 10009 10083 10165 10220
RTT      822
SBC     9660  9661  9662  9665  9666  9669
SEC     1886  4299  7945  8814
SETD    5189  5260  5400  6185
SETF    4244  5184  5255  5395
SOB     1135  3429  3557  3663  4463  4559  5725  6288  6684  6709  7006  7126  7253  7845  7846
        8233  8292  8413  8674  8707  8823  8850  8978
STCDF   5326
STCFD   5256
STD     4547  4658  4689  4720  4751  4782  4784  4815  4817  4979  4985  5045  5053  5066  5073
        5086  5092  5107  5125  5190  5191  5261  5262  5330  5331  5401  5402  5470  5471  5645
        5706  5716  5856  5925  5947  6053  6071  6186  6296  6301  6691  6696  7223  7224  7266
        7267  7568  7569  7861  7862  8239  8240  8247  8252  8258  8295  8300  8305  8310  8419
        8420  8426  8431  8437  8740  8741  8742  8885  8886  9021  9022  9023
STEXP   2992  3910
STF     2386  2418  2450  2482  2610  2614  2646  2650  2682  2686  2718  2722  2850  2854  2886
        2890  3089  3108  3125  3133  3147  3154  3168  3174  3241  3242  3261  3262  3279  3283
        3287  3301  3304  3308  3322  3324  3328  3767  3774  3989  3991  3995  4017  4019  4023
        4045  4047  4051  4073  4075  4079  4145  4165  4169  4253  4451  5559  5787  6988  6993
        7116  8557  8564
STFPS   1386  8562  8735  8880  9016  9162  9915
STST    1387  1425  1440  8563  8736  8881  9017  9163
SUB     1800  2986  4306  7766  9582  9608  9659  9664  9668  9670  9864  9872  9877 10195
SUBF    4052  4080
SWAB   10061
SXT     2076  9618  9619  9620  9621
TRAP   10336 10345 10346 10347 10349 10350 10351 10352 10353 10354 10355 10356 10357 10358 10359
       10360 10361 10362 10363 10364 10365 10366 10367 10368 10369 10370
TST     1003  4311  7285  7600  7602  8694  8715  8863  8986  9140  9213  9259  9471  9572  9617
        9785  9796  9809  9901  9926  9988 10122 10130 10152 10190 10208 10210 10288 10326
TSTB     891  1169  1654  1778  1813  2044  2138  2248  2388  2420  2452  2484  2616  2652  2688
```

```
            2724    2856    2892    2993    3090    3109    3130    3151    3171    3243    3263    3284    3305    3325    3415
            3543    3650    3770    3907    3998    4026    4054    4082    4147    4171    4255    4349    4452    4548    4659
            4690    4721    4752    4785    4818    4901    5050    5070    5089    5108    5126    5186    5257    5327    5397
            5467    5556    5642    5713    5782    5852    5944    6068    6181    6298    6693    6990    7113    7220    7263
            7565    7616    7858    8255    8307    8434    8559    8732    8802    8844    8856    8877    8992    9013    9130
            9159    9165    9200    9241    9713    9767    9934   10072   10105   10154   10182   10193   10206
TSTF        1270
.ASCII       379     380     714     717     718   10967   11015
.ASCIZ       378     381     711     712     713     719     726    9884    9885   10012   10013   10550   10559   10564   10568
           10572   10576   10581   10586   10590   10593   10597   10601   10606   10611   10615   10623   10631   10635   10639
           10646   10652   10657   10660   10666   10672   10679   10683   10687   10694   10700   10708   10712   10717   10724
           10731   10737   10741   10746   10751   10757   10763   10768   10773   10776   10783   10790   10795   10800   10805
           10810   10815   10821   10827   10832   10836   10842   10848   10854   10861   10868   10876   10883   10889   10895
           10902   10909   10914   10920   10926   10931   10936   10941   10946   10951   10960   10970   10974   10979   10984
           10990   10997   11002   11007   11012   11017   11019   11024   11028   11030   11032   11043   11048   11051   11055
           11057   11060   11064   11070   11073   11077   11081   11087   11089   11091   11098   11105   11222   11224   11226
           11228   11230   11232   11234   11236   11238   11240   11242   11244   11246   11248   11250   11252   11254   11256
           11258   11260   11262   11264   11266   11269   11272   11274   11276   11278   11280   11282   11284   11286   11288
           11291
.BLKW        625     626     627     628     629     630     631     632
.BYTE        342     343     354     355     356     357     397     398     606     607     608     610     619     621     837
            1859    1860    1861    1862    1863    8081    8082    8083    8084    8085    8086    8087    8088    8089    8090
            8091    8092    8093    8094    8095    8096    8110    8111    8112    8113    8114    8115    8116    8117    8118
            8119    8120    8121    8122    8123    8124    8125    8140    8141    8142    8143    8144    8145    8146    8147
            8162    8163    8164    8165    8166    8167    8168    8169    9898    9899   10221   10222   10223   10306   10307
           10308   10309   11298
.ENABL         1     733
.END       11298
.ENDC          1       7      35     127     1[illegible]     212     286     290     292     297     299     306     320     324     327
             358     376     377     378     379     383     386     410     729     733     737     745     746     749     751
             753     755     756     758     759     761     763     784     786     788     807     809     828     840     841
             883     884     885     893     914     918     929     931     932     933     934     944     947     948     979
             980     981     987    1064    1081    1082    1122    1123    1124    1156    1171    1186    1218    1219    1250
            1251    1252    1322    1323    1378    1379    1380    1466    1467    1499    1500    1501    1517    1548    1571
            1572    1623    1624    1625    1639    1656    1676    1706    1707    1755    1756    1757    1815    1853    1855
            1868    1910    1911    1912    1913    1914    1915    1916    1917    1918    1919    1920    1921    1922    1923
            1924    1925    1926    1927    1928    1929    1930    1931    1932    1933    1934    1935    1936    1937    1938
            1939    1940    1941    1942    1943    1944    1945    1946    1947    1948    1949    1950    1951    1952    1953
            1954    1955    1956    1957    1958    1959    1960    1961    1962    1963    1964    1965    1966    1967    1968
            1969    1970    1971    1972    1973    1977    1978    2017    2018    2019    2030    2046    2059    2086    2087
            2111    2112    2113    2125    2127    2133    2140    2150    2153    2170    2175    2196    2197    2221    2222
            2223    2235    2237    2243    2250    2260    2263    2280    2285    2299    2301    2302    2303    2304    2314
            2315    2316    2364    2365    2366    2390    2422    2454    2476    2486    2501    2533    2534    2588    2589
            2590    2618    2654    2690    2712    2726    2741    2773    2774    2828    2829    2830    2858    2880    2894
            2910    2942    2943    2975    2976    2977    2986    2995    3010    3036    3037    3077    3078    3079    3092
            3111    3132    3153    3173    3188    3189    3229    3230    3231    3245    3265    3286    3307    3327    3342
            3343    3390    3391    3392    3417    3432    3436    3467    3468    3518    3519    3520    3545    3560    3562
            3579    3580    3628    3629    3630    3652    3666    3669    3697    3698    3741    3742    3743    3772    3805
            3808    3843    3844    3887    3888    3889    3909    3927    3928    3979    3980    3981    4000    4028    4056
            4084    4100    4101    4133    4134    4135    4149    4173    4186    4187    4225    4226    4227    4257    4284
            4287    4333    4386    4390    4391    4433    4434    4435    4454    4465    4483    4484    4527    4528    4529
            4550    4561    4603    4604    4636    4637    4638    4661    4692    4723    4754    4787    4820    4834    4929
            4931    4935    4936    4970    4971    4972    4983    4989    4997    4998    5035    5036    5037    5052    5072
            5091    5110    5128    5141    5142    5174    5175    5176    5188    5204    5212    5213    5245    5246    5247
            5259    5275    5283    5284    5316    5317    5318    5329    5344    5352    5353    5385    5386    5387    5399
            5415    5423    5424    5456    5457    5458    5469    5484    5492    5493    5534    5535    5536    5547    5558
```

```
                5572    5597    5598    5631    5632    5633    5644    5648    5656    5657    5693    5694    5695    5715    5730
                5731    5769    5770    5771    5784    5791    5800    5801    5837    5838    5839    5854    5860    5868    5869
                5911    5912    5913    5921    5946    5970    5998    5999    6040    6041    6042    6051    6070    6092    6114
                6115    6157    6158    6159    6166    6183    6199    6228    6229    6275    6276    6277    6300    6328    6333
                6338    6341    6347    6352    6357    6360    6366    6371    6376    6379    6385    6390    6395    6398    6404
                6409    6414    6417    6423    6428    6433    6436    6442    6447    6452    6455    6461    6466    6471    6474
                6480    6485    6490    6493    6499    6504    6509    6512    6518    6523    6528    6531    6537    6542    6547
                6550    6556    6561    6566    6569    6575    6580    6585    6588    6594    6599    6604    6607    6613    6618
                6619    6670    6671    6672    6695    6713    6915    6919    6920    6966    6967    6968    6992    7010    7053
                7057    7058    7095    7096    7097    7115    7128    7130    7156    7157    7204    7205    7206    7222    7265
                7438    7439    7501    7502    7503    7567    7663    7670    7671    7749    7750    7751    7752    7860    7863
                7904    7991    8035    8071    8100    8129    8151    8171    8176    8177    8224    8225    8226    8257    8309
                8333    8335    8344    8355    8356    8405    8406    8407    8436    8464    8466    8475    8476    8477    8531
                8532    8533    8544    8561    8591    8631    8632    8633    8657    8658    8659    8734    8765    8766    8786
                8787    8788    8879    8909    8910    8934    8935    8936    9015    9055    9056    9057    9058    9059    9067
                9068    9072    9075    9076    9078    9086    9092    9095    9096    9098    9104    9106    9109    9113    9152
                9221    9266    9281    9284    9301    9304    9341    9376    9412    9458    9495    9530    9533    9639    9680
                9729    9732    9736    9743    9747    9748    9750    9756    9759    9767    9773    9778    9780    9791    9794
                9795    9796    9798    9800    9807    9811    9816    9818    9822    9825    9826    9830    9834    9837    9847
                9863    9891    9892    9893    9901    9930    9934    9939    9943   10020   10087   10091   10120   10171   10175
               10176   10179   10206   10221   10233   10237   10314   10318   10344   10345   10346   10347   10348   10349   10350
               10351   10352   10353   10354   10355   10356   10357   10358   10359   10360   10361   10362   10363   10364   10365
               10366   10367   10368   10369   10370   10371   10374   10401   10435   10460   10480   10481   10492   10504   10505
               10509   10518   10519   10525   10533   10534   10544   10546   10553   10555
.EQUIV            35      36      44      89      90      91      92      93      94      95      96      97      98     117     118
                 119     120     121     122     123     124     125     126
.EVEN             386     593    9889   10016   10224   10552   11111   11165   11294
.IF                 1       3      33      99     127     208     285     288     290     296     298     305     319     323     326
                  358     376     377     378     382     383     385     408     410     729     730     734     740     745     747
                  749     751     753     755     756     758     759     761     779     785     787     806     808     828     839
                  841     883     885     892     913     917     926     930     932     934     941     946     948     979     981
                  986    1063    1080    1082    1122    1124    1155    1170    1185    1217    1219    1250    1252    1321    1323
                 1378    1380    1465    1467    1499    1501    1516    1547    1570    1572    1623    1625    1638    1655    1675
                 1705    1707    1755    1757    1814    1852    1854    1867    1909    1910    1911    1912    1913    1914    1915
                 1916    1917    1918    1919    1920    1921    1922    1923    1924    1925    1926    1927    1928    1929    1930
                 1931    1932    1933    1934    1935    1936    1937    1938    1939    1940    1941    1942    1943    1944    1945
                 1946    1947    1948    1949    1950    1951    1952    1953    1954    1955    1956    1957    1958    1959    1960
                 1961    1962    1963    1964    1965    1966    1967    1968    1969    1970    1971    1972    1973    1976    1978
                 2017    2019    2029    2045    2058    2085    2087    2111    2113    2125    2126    2127    2132    2139    2149
                 2152    2169    2175    2195    2197    2221    2223    2235    2236    2237    2242    2249    2259    2262    2279
                 2285    2296    2300    2302    2304    2311    2314    2316    2364    2366    2389    2421    2453    2475    2485
                 2500    2532    2534    2588    2590    2617    2653    2689    2711    2725    2740    2772    2774    2828    2830
                 2857    2879    2893    2909    2941    2943    2975    2977    2985    2994    3009    3035    3037    3077    3079
                 3091    3110    3131    3152    3172    3187    3189    3229    3231    3244    3264    3285    3306    3326    3341
                 3343    3390    3392    3416    3431    3435    3466    3468    3518    3520    3544    3559    3561    3578    3580
                 3628    3630    3651    3665    3668    3696    3698    3741    3743    3771    3804    3807    3842    3844    3887
                 3889    3908    3926    3928    3979    3981    3999    4027    4055    4083    4099    4101    4133    4135    4148
                 4172    4185    4187    4225    4227    4256    4283    4286    4332    4385    4389    4391    4433    4435    4453
                 4464    4482    4484    4527    4529    4549    4560    4602    4604    4636    4638    4660    4691    4722    4753
                 4786    4819    4833    4928    4930    4934    4936    4970    4972    4982    4988    4996    4998    5035    5037
                 5051    5071    5090    5109    5127    5140    5142    5174    5176    5187    5203    5211    5213    5245    5247
                 5258    5274    5282    5284    5316    5318    5328    5343    5351    5353    5385    5387    5398    5414    5422
                 5424    5456    5458    5468    5483    5491    5493    5534    5536    5546    5557    5571    5596    5598    5631
                 5633    5643    5647    5655    5657    5693    5695    5714    5729    5731    5769    5771    5783    5790    5799
                 5801    5837    5839    5853    5859    5867    5869    5911    5913    5920    5945    5969    5997    5999    6040
                 6042    6050    6069    6091    6113    6115    6157    6159    6165    6182    6198    6227    6229    6275    6277
```

```
                6299    6326    6328    6331    6333    6336    6338    6345    6347    6350    6352    6355    6357    6364    6366
                6369    6371    6374    6376    6383    6385    6388    6390    6393    6395    6402    6404    6407    6409    6412
                6414    6421    6423    6426    6428    6431    6433    6440    6442    6445    6447    6450    6452    6459    6461
                6464    6466    6469    6471    6478    6480    6483    6485    6488    6490    6497    6499    6502    6504    6507
                6509    6516    6518    6521    6523    6526    6528    6535    6537    6540    6542    6545    6547    6554    6556
                6559    6561    6564    6566    6573    6575    6578    6580    6583    6585    6592    6594    6597    6599    6602
                6604    6612    6617    6619    6670    6672    6694    6712    6914    6918    6920    6966    6968    6991    7009
                7052    7056    7058    7095    7097    7114    7127    7129    7155    7157    7204    7206    7221    7264    7437
                7439    7501    7503    7566    7662    7669    7671    7749    7751    7752    7859    7863    7903    7990    8034
                8070    8099    8128    8150    8170    8175    8177    8224    8226    8256    8308    8332    8334    8343    8354
                8356    8405    8407    8435    8463    8465    8474    8475    8477    8531    8533    8543    8560    8590    8630
                8631    8633    865?    8659    8733    8764    8766    8786    8788    8878    8908    8910    8934    8936    9014
                9052    9055    905?    9059    9067    9068    9069    9074    9075    9076    9077    9078    9080    9091    9094
                9096    9098    91?4    9106    9108    9112    9151    9220    9265    9269    9283    9300    9303    9340    9375
                9411    9457    9494    9529    9532    9638    9679    9728    9731    9735    9742    9747    9749    9755    9758
                9766    9772    9773    9790    9792    9793    9794    9796    9?5?    9798    9807    9809    9817    9819    9824
                9825    9826    9829    9833    9836    9847    9859    9866    9868    9891    9892    9894    9901    9904    9930
                9938    9939    9942    10019   10086   10090   10111   10170   10174   10176   10179   10206   10221   10232   10236
                10313   10317   10336   10345   10346   10347   10348   10349   10350   10351   10352   10353   10354   10355   10356
                10357   10358   10359   10360   10361   10362   10363   10364   10365   10366   10367   10368   10369   10370   10371
                10373   10400   10434   10459   10480   10491   10503   10504   10508   10518   10519   10524   10531   10534   10542
                10544   10546   10550   10554
.IFF             35     212     286     290     292     297     299     306     320     ?74     327     358     383     386     730
                 ?54    745     786     788     807     809     840     841     884     6d5     892     914     918     926     931
                 932    933     934     941     947     948     980     981     987     1064    1081    1082    1123    1124    1156
                1170    1186    1218    1219    1251    1252    1322    1323    1379    1380    1466    1467    1500    1501    1517
                1548    1571    1572    1624    1625    1639    1655    1676    1706    1707    1756    1757    1814    1853    1855
                1868    1909    1910    1911    1912    1913    1914    1915    1916    1917    1918    1919    1920    1921    1922
                1923    1924    1925    1926    1927    1928    1929    1930    1931    1932    1933    1934    1935    1936    1937
                1938    1939    1940    1941    1942    1943    1944    1945    1946    1947    1948    1949    1950    1951    1952
                1953    1954    1955    1956    1957    1958    1959    1960    1961    1962    1963    1964    1965    1966    1967
                1968    1969    1970    1971    1972    1973    1977    1978    2018    2019    2030    2045    2059    2086    2087
                2112    2113    2125    2126    2127    2132    2139    2150    2153    2170    2196    2197    2222    2223    2235
                2237    2242    2249    2260    2263    2280    2296    2301    2302    2303    2304    2311    2315    2316    2365
                2366    2390    2422    2454    2476    2486    2501    2533    2534    2589    2590    2618    2654    2690    2712
                2726    2741    2773    2774    2829    2830    2858    2880    2894    2910    2942    2943    2976    2977    2986
                2994    3010    3036    3037    3078    3079    3092    3111    3132    3153    3173    3188    3189    3230    3231
                3245    3265    3286    3307    3327    3342    3343    3391    3392    3416    3432    3436    3467    3468    3519
                3520    3544    3560    3562    3579    3580    3629    3630    3651    3666    3669    3697    3698    3742    3743
                3771    3805    3808    3843    3844    3888    3889    3908    3927    3928    398C    3981    4000    4028    4056
                4084    4100    4101    4134    4135    4149    4173    4186    4187    4226    4227    425C    4284    4287    4333
                4386    4390    4391    4434    4435    4453    4465    4483    4484    4528    4529    4549    4561    4603    4604
                4637    4638    4661    4692    4723    4754    4787    4820    4834    4929    4931    4935    4936    4971    4972
                4982    4989    4997    4998    5036    5037    5052    5072    5091    5110    5128    5141    5142    5175    5176
                5187    5204    5212    5213    5246    5247    5258    5275    5283    5284    5317    5318    5328    5344    5352
                5353    5386    5387    5398    5415    5423    5424    5457    5458    5468    5484    5492    5493    5535    5536
                5547    5557    5572    5597    5598    5632    5633    5643    5648    5656    5657    5694    5695    5714    5730
                5731    5770    5771    5783    5791    5800    5801    5838    5839    5853    5860    5868    5869    5912    5913
                5921    5945    5970    5998    5999    6041    6042    6051    6069    6092    6114    6115    6158    6159    6166
                6182    6199    6228    6229    6276    6277    6299    6328    6333    6336    6347    6352    6355    6366    6371
                6374    6385    6390    6393    6404    6409    6412    6423    6426    6431    6442    6445    6450    6461    6466
                6469    6480    6485    6488    6499    6504    6507    6518    6523    6526    6537    6542    6545    6556    6561
                6564    6575    6580    6583    6594    6599    6602    6613    6618    6619    6671    6672    6694    6713    6915
                6919    6920    6967    6968    6991    7010    7053    7057    7058    7096    7097    7114    7128    7130    7156
                7157    7205    7206    7221    7265    7438    7439    7502    7503    7566    7663    7670    7671    7750    7751
                7859    7904    7991    8035    8071    8100    8129    8151    8171    8176    8177    8225    8226    8256    8309
```

```
                8333    8335    8344    8355    8356    8406    8407    8435    8464    8466    8475    8476    8477    8532    8533
                8544    8560    8591    8631    8632    8633    8658    8659    8733    8765    8766    8787    8788    8878    8909
                8910    8935    8936    9014    9052    9056    9057    9058    9059    9069    9075    9077    9080    9092    9095
                9106    9109    9112    9152    9221    9266    9281    9284    9301    9304    9341    9376    9412    9458    9495
                9530    9533    9639    9680    9729    9732    9736    9748    9750    9756    9759    9767    9773    9791    9794
                9795    9798    9825    9826    9830    9834    9836    9859    9930    9939    9943   10020   10087   10091   10171
               10175   10233   10237   10314   10318   10374   10401   10435   10460   10481   10492   10504   10505   10509   10525
               10544   10555
.IFT            208     828     892    1170    1655    1814    1909    1910    1911    1912    1913    1914    1915    1916    1917
               1918    1919    1920    1921    1922    1923    1924    1925    1926    1927    1928    1929    1930    1931    1932
               1933    1934    1935    1936    1937    1938    1939    1940    1941    1942    1943    1944    1945    1946    1947
               1948    1949    1950    1951    1952    1953    1954    1955    1956    1957    1958    1959    1960    1961    1962
               1963    1964    1965    1966    1967    1968    1969    1970    1971    1972    1973    2045    2125    2126    2139
               2149    2152    2175    2235    2236    2237    2249    2259    2262    2285    2389    2421    2453    2485    2617
               2653    2689    2725    2857    2893    2994    3091    3110    3131    3152    3172    3244    3264    3285    3306
               3326    3416    3544    3651    3771    3908    3999    4027    4055    4083    4148    4172    4256    4453    4549
               4660    4691    4722    4753    4786    4819    4982    5051    5071    5090    5109    5127    5187    5258    5328
               5398    5468    5557    5643    5714    5783    5853    5945    6069    6182    6299    6326    6328    6331    6333
               6336    6338    6345    6347    6350    6352    6355    6357    6364    6366    6369    6371    6374    6376    6383
               6385    6388    6390    6393    6395    6402    6404    6407    6409    6412    6414    6421    6423    6426    6428
               6431    6433    6440    6442    6445    6447    6450    6452    6459    6461    6464    6466    6469    6471    6478
               6480    6483    6485    6488    6490    6497    6499    6502    6504    6507    6509    6516    6518    6521    6523
               6526    6528    6535    6537    6540    6542    6545    6547    6554    6556    6559    6561    6564    6566    6573
               6575    6578    6580    6583    6585    6592    6594    6597    6599    6602    6604    6694    6991    7114    7221
               7264    7566    7859    8256    8308    8435    8560    8733    8878    9014    9067    9068    9112    9269    9806
               9892
.IFTF          9804    9868
.IIF              2       7      12     277     382     386     746     749     755     758     759     761     762     988     991
               1157    1160    1265    1268    1291    1294    1401    1404    1646    1649    1801    1804    2037    2040    2121
               2124    2231    2234    2380    2383    2412    2415    2444    2447    2476    2479    2604    2607    2640    2643
               2676    2679    2712    2715    2844    2847    2880    2883    2987    2990    3084    3086    3103    3105    3122
               3124    3144    3146    3165    3167    3236    3238    3256    3258    3276    3278    3298    3300    3319    3321
               3409    3412    3537    3540    3643    3646    3759    3762    3900    3903    3993    3995    4021    4023    4049
               4051    4077    4079    4143    4145    4167    4169    4249    4252    4445    4448    4541    4544    4653    4656
               4684    4687    4715    4718    4746    4749    4777    4780    4810    4813    4974    4977    5042    5044    5063
               5065    5083    5085    5102    5104    5120    5122    5179    5182    5250    5253    5321    5324    5390    5393
               5461    5464    5549    5552    5636    5639    5707    5710    5777    5780    5845    5848    5938    5941    6062
               6065    6175    6178    6232    6295    6687    6690    6984    6987    7107    7110    7211    7214    7254    7257
               7558    7561    7850    7853    8248    8251    8301    8304    8427    8430    8553    8556    8721    8724    8867
               8870    9002    9005    9086    9087    9106    9738    9739    9740    9741    9742    9747    9805    9806    9822
               9825    9826    9837    9838    9839    9840    9841    9846    9875    9880    9904    9930    9939   10167   10344
              10345   10346   10347   10349   10350   10351   10352   10353   10354   10355   10356   10357   10358   10359   10360
              10361   10362   10363   10364   10365   10366   10367   10368   10369   10370
.IRP            729     839     930     946    1080    1217    1321    1465    1570    1705    1976    2085    2195    2300    2314
               2532    2772    2941    3035    3187    3341    3466    3578    3696    3842    3926    4099    4185    4389    4482
               4602    4934    4996    5140    5211    5282    5351    5422    5491    5596    5655    5729    5799    5867    5997
               6113    6227    6617    6918    7056    7155    7437    7669    8175    8354    8475    8631    8764    8908    9055
               9080    9385    9403   10180   10181   10202   10218   10219   10512   10518   10533   10534
.LIST             1     141     277     358     360     361     362     363     364     365     366     367     368     369     370
                371     372     373     374     375     376     383     386     729     730     731     732     733     734     735
                736     737     763     839     885     926     927     928     929     930     934     941     942     943     944
                946     981    1080    1124    1217    1252    1321    1380    1465    1501    1570    1625    1705    1757    1902
               1910    1911    1912    1913    1914    1915    1916    1917    1918    1919    1920    1921    1922    1923    1924
               1925    1926    1927    1928    1929    1930    1931    1932    1933    1934    1935    1936    1937    1938    1939
               1940    1941    1942    1943    1944    1945    1946    1947    1948    1949    1950    1951    1952    1953    1954
               1955    1956    1957    1958    1959    1960    1961    1962    1963    1964    1965    1966    1967    1968    1969
```

M05


```
                1970    1971    1972    1973    1976    2019    2085    2113    2195    2223    2296    2297    2298    2299    2300
                2304    2311    2312    2313    2314    2366    2532    2590    2772    2830    2941    2977    3035    3079    3187
                3231    3341    3392    3433    3466    3520    3561    3578    3630    3667    3696    3743    3842    3889    3926
                3981    4099    4135    4185    4227    4389    4435    4482    4529    4602    4638    4934    4972    4996    5037
                5140    5176    5211    5247    5282    5318    5351    5387    5422    5458    5491    5536    5596    5633    5655
                5695    5729    5771    5799    5839    5867    5913    5997    6042    6113    6159    6227    6277    6617    6672
                6715    6918    6968    7012    7056    7097    7155    7206    7292    7335    7437    7503    7669    7751    8175
                8226    8354    8407    8475    8533    8631    8659    8764    8788    8908    8936    9052    9053    9054    9055
                9059    9069    9070    9071    9072    9086    9098    9742    9930   10336   10344   10345   10346   10347   10349
               10350   10351   10352   10353   10354   10355   10356   10357   10358   10359   10360   10361   10362   10363   10364
               10365   10366   10367   10368   10369   10370   11298
.MACRO             1     317     779     839     946    1080    1217    1321    1465    1570    1705    1902    1976    2085    2195
                2314    2532    2772    2941    3035    3187    3341    3466    3578    3696    3842    3926    4099    4185    4389
                4482    4602    4934    4996    5140    5211    5282    5351    5422    5491    5596    5655    5729    5799    5867
                5997    6113    6227    6323    6617    6918    7056    7155    7292    7437    7669    8175    8354    8475    8631
                8764    8908    9072   10336
.MCALL             1     141     383     763
.MEXIT           409
.NLIST             1     141     277     358     360     361     362     363     364     365     366     367     368     369     370
                 371     372     373     374     375     376     383     386     729     730     731     732     733     734     735
                 736     737     763     839     885     926     927     928     929     930     934     941     942     943     944
                 946     981    1080    1124    1217    1252    1321    1380    1465    1501    1570    1625    1705    1757    1902
                1910    1911    1912    1913    1914    1915    1916    1917    1918    1919    1920    1921    1922    1923    1924
                1925    1926    1927    1928    1929    1930    1931    1932    1933    1934    1935    1936    1937    1938    1939
                1940    1941    1942    1943    1944    1945    1946    1947    1948    1949    1950    1951    1952    1953    1954
                1955    1956    1957    1958    1959    1960    1961    1962    1963    1964    1965    1966    1967    1968    1969
                1970    1971    1972    1973    1976    2019    2085    2113    2195    2223    2296    2297    2298    2299    2300
                2304    2311    2312    2313    2314    2366    2532    2590    2772    2830    2941    2977    3035    3079    3187
                3231    3341    3392    3433    3466    3520    3561    3578    3630    3667    3696    3743    3842    3889    3926
                3981    4099    4135    4185    4227    4389    4435    4482    4529    4602    4638    4934    4972    4996    5037
                5140    5176    5211    5247    5282    5318    5351    5387    5422    5458    5491    5536    5596    5633    5655
                5695    5729    5771    5799    5839    5867    5913    5997    6042    6113    6159    6227    6277    6617    6672
                6715    6918    6968    7012    7056    7097    7155    7206    7292    7335    7437    7503    7669    7751    8175
                8226    8354    8407    8475    8533    8631    8659    8764    8788    8908    8936    9052    9053    9054    9055
                9059    9069    9070    9071    9072    9086    9098    9742    9930   10336   10344   10345   10346   10347   10349
               10350   10351   10352   10353   10354   10355   10356   10357   10358   10359   10360   10361   10362   10363   10364
               10365   10366   10367   10368   10369   10370   11298
.PAGE            317     410     728     839    1705    2085    2296    2314    3926    4389    5491    6113    6227    6617    6918
                7056    7155    7291    8475    8631    9072    9106    9301    9530    9729   10314   10555
.REPT            277     360     730     734     926     941    1913    1917    1921    1925    1929    1933    1937    1941    1945
                1949    1953    1957    1961    1965    1969    2296    2311    9052    9069    9679   10371   11298
.SBTTL            13      31     255     280     283     294     317     383     410     591     728     739     839     930     946
                1080    1217    1321    1465    1570    1705    1976    2085    2195    2300    2314    2532    2772    2941    3035
                3187    3341    3466    3578    3696    3842    3926    4099    4185    4389    4482    4602    4934    4996    5140
                5211    5282    5351    5422    5491    5596    5655    5729    5799    5867    5997    6113    6227    6617    6918
                7056    7155    7437    7669    6175    8354    8475    8631    8764    8908    9055    9072    9106    9110    9153
                9222    9267    9285    9301    9305    9342    9373    9413    9459    9496    9530    9534    9640    9681    9729
                9733    9831    9944   10021   10088   10172   10234   10314   10336   10375   10402   10436   10461   10482   10493
               10506   10555
.TITLE             2
.WORD            275     276     277     278     279     291     310     311     312     313     314     315     326     327     328
                 329     330     331     332     333     334     335     336     337     338     339     340     341     344     346
                 347     348     349     358     360     361     362     363     364     365     366     367     368     369     370
                 371     372     373     374     375     388     389     390     391     392     393     394     395     399     400
                 401     425     427     429     431     433     435     437     439     441     443     445     447     449     451
                 453     455     457     459     461     463     465     467     469     471     473     475     477     479     481
```

```
  483   485   487   489   491   493   495   497   499   501   503   505   507   509   511
  513   515   517   519   521   523   525   527   529   531   533   535   537   539   541
  543   545   547   549   551   553   555   557   559   561   563   565   567   569   571
  573   575   577   579   581   583   595   599   600   601   602   604   611   612   614
  615   616   617   618   635   636   637   638   639   640   641   642   645   646   647
  648   650   656   661   662   663   664   666   667   668   669   670   672   673   674
  675   677   678   680   682   684   686   688   690   692   702   703   704   705   706
  707  1069  1071  1073  1075  1077  1191  1193  1195  1197  1199  1201  1203  1205  1207
 1209  1211  1213  1681  1683  1685  1687  1689  1691  1693  1695  1697  1699  1701  1909
 1910  1911  1912  1913  1914  1915  1916  1917  1918  1919  1920  1921  1922  1923  1924
 1925  1926  1927  1928  1929  1930  1931  1932  1933  1934  1935  1936  1937  1938  1939
 1940  1941  1942  1943  1944  1945  1946  1947  1948  1949  1950  1951  1952  1953  1954
 1955  1956  1957  1958  1959  1960  1961  1962  1963  1964  1965  1966  1967  1968  1969
 1970  1971  1972  2506  2508  2510  2512  2514  2516  2518  2520  2522  2524  2526  2528
 2746  2748  2750  2752  2754  2756  2758  2760  2762  2764  2766  2768  2915  2917  2919
 2921  2923  2925  2927  2929  2931  2933  2935  2937  3015  3017  3019  3021  3023  3025
 3027  3029  3031  4468  4470  4472  4474  4476  4478  4564  4567  4570  4573  4576  4579
 4582  4585  4588  4591  4594  4597  5924  6052  6326  6331  6336  6345  6350  6355  6364
 6369  6374  6383  6388  6393  6402  6407  6412  6421  6426  6431  6440  6445  6450  6459
 6464  6469  6478  6483  6488  6497  6502  6507  6516  6521  6526  6535  6540  6545  6554
 6559  6564  6573  6578  6583  6592  6597  6602  6608  7135  7137  7139  7141  7143  7145
 7147  7149  7151  7339  7341  7343  7345  7347  7349  7351  7353  7357  7359  7361  7363
 7365  7367  7369  7371  7373  7375  7377  7379  7381  7383  7385  7387  7389  7391  7393
 7395  7397  7399  7401  7403  7405  7407  7409  7411  7413  7415  7417  7419  7421  7423
 7425  7427  7430  7952  7953  7954  7955  7956  7957  7958  7966  7967  7968  7969  7970
 7971  7972  7973  7981  7982  7983  7984  7985  7986  7987  7988  8017  8018  8019  8020
 8021  8024  8025  8026  8029  8030  8031  8336  8338  8340  8345  8347  8349  8467  8469
 8471  9091  9094  9105  9407  9408  9485  9486  9487  9488  9489  9490  9491  9970  9976
 9999 10004 10010 10011 10117 10164 10204 10310 10343 10543 10545 11113 11116 11117 11122
11124 11125 11126 11128 11130 11133 11134 11135 11136 11137 11139 11141 11143 11144 11147
11149 11151 11153 11156 11157 11159 11161 11166 11168 11171 11174 11176 11180 11182 11185
11188 11189 11190 11191 11194 11197 11200 11204 11206 11207 11210 11212 11214 11215 11218

. ABS.  045520     000


 ERRORS DETECTED:  0

 DSKM:DQFPEA,DSKW:DQFPEA=DQFPEA.P11
 RUN-TIME: 36 37 5 SECONDS
 RUN-TIME RATIO: 249/80=3.1
 CORE USED:  25K  (49 PAGES)

 DOCUMENT PAGES:  252
 WRAP-AROUND:  0%

 USER SYMBOLS:  915
 MACRO  NAMES:  66
 UNDF SYMBOLS:  13
 DISK BLOCKS  READ:  1973
 DISK BLKS WRITTEN:  1335
 KILO CORE SECONDS:  2910
```

B06