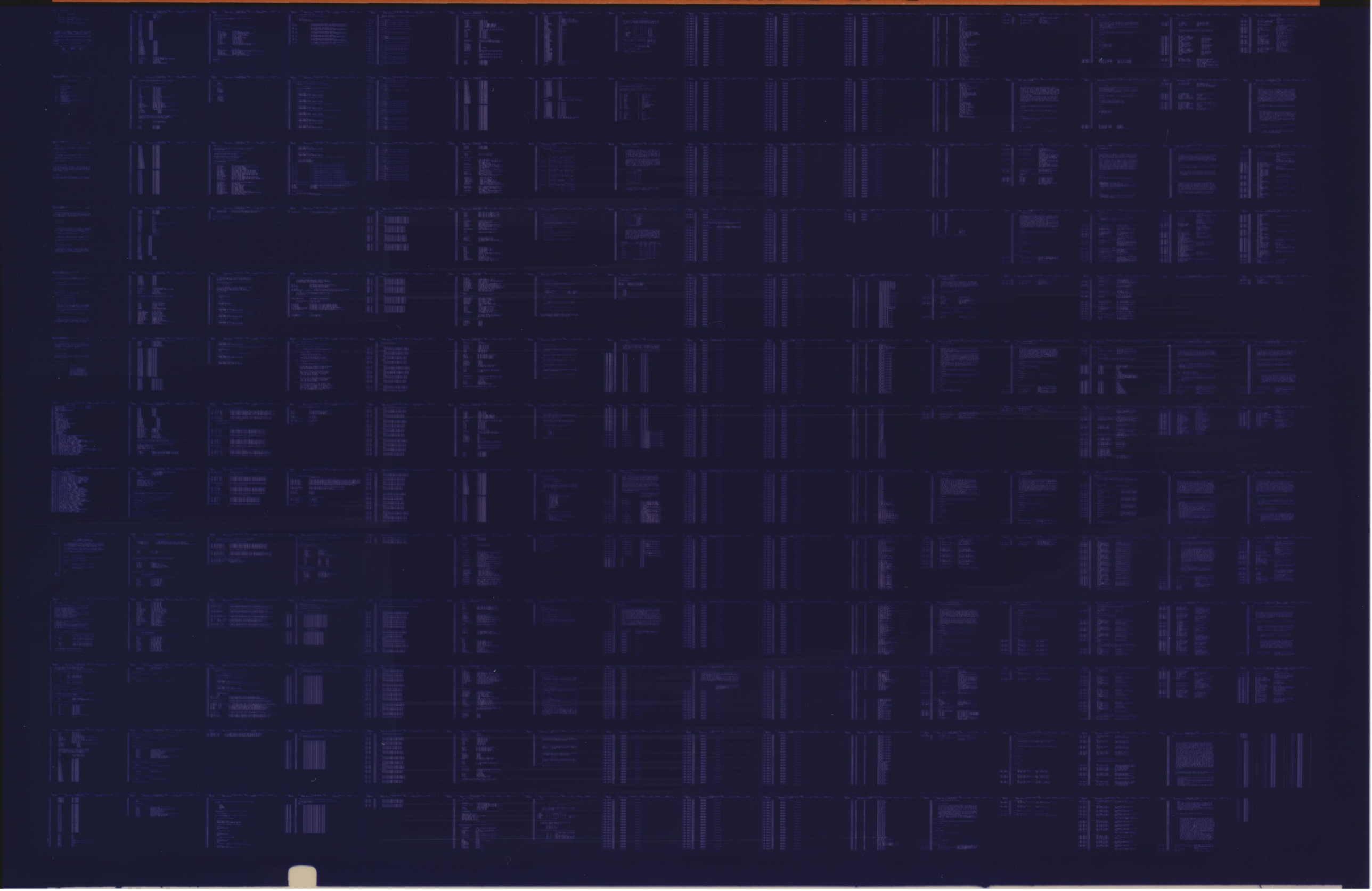
ENKCE-1.1 VAX 11/730 FPA MICRO

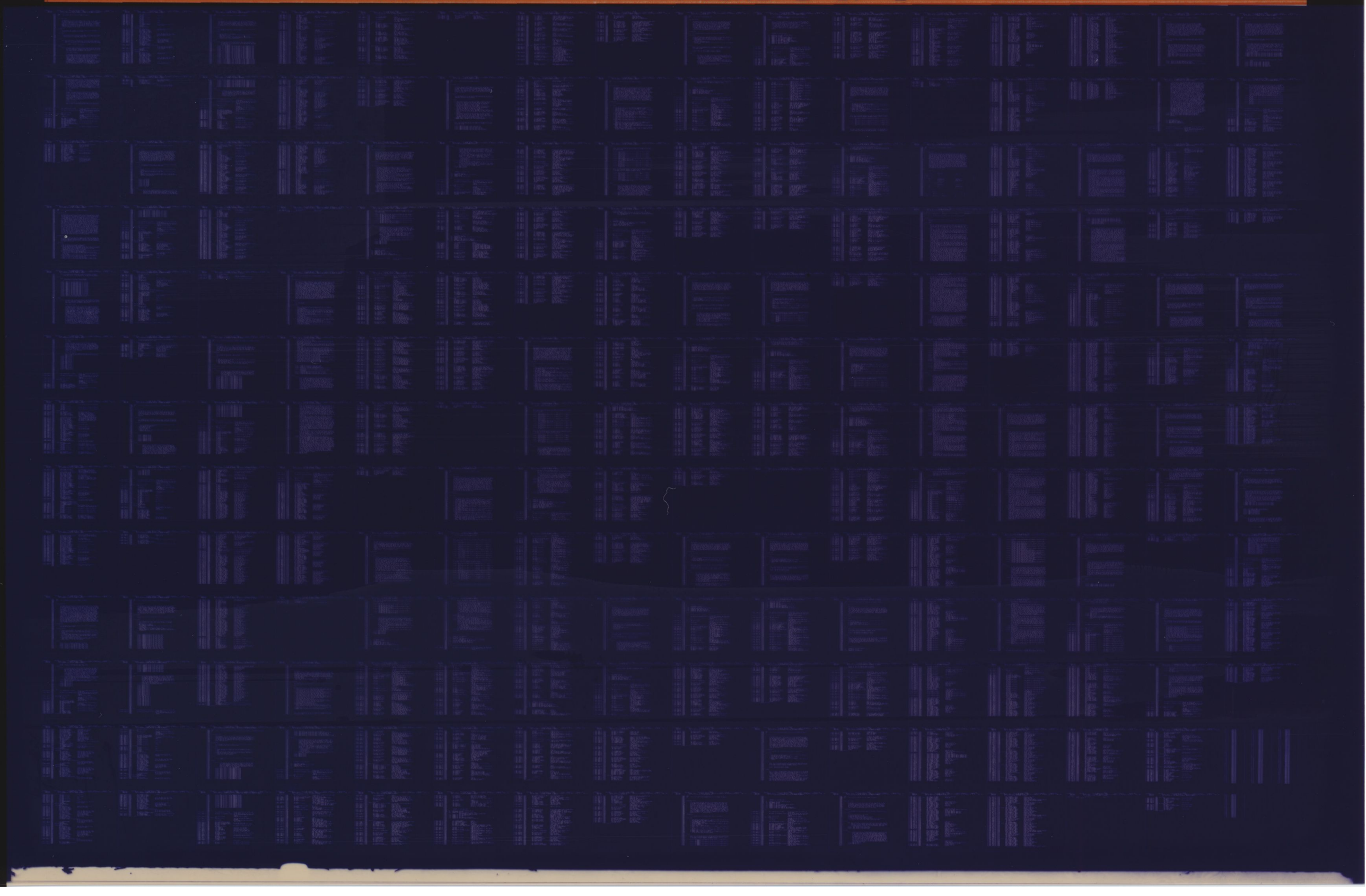
EP-ENKCE-DL-1.1 FICHE 1 OF 3

SEP 1982
COPYRIGHT® 1982
MADE IN USA



ENKCE-1.1 VAX 11/730 FPA MICRO EP-ENKCE-DL-1.1 FICHE 2 OF 3

SEP 1982
COPYRIGHT© 1982
MADE IN USA



ENKCE-1.1 VAX 11/730 FPA MICRO

EP-ENKCE-DL-1.1 FICHE 3 OF 3

SEP 1982
COPYRIGHT© 1982
MADE IN USA

d i g i t a l

IDENTIFICATION

Product code: ZZ-ENKCE VERSION 1.1

Product name: MICRO-DIAGNOSTIC FOR VAX-11/730 FPA MODULE

Product date: 8-MARCH-1982

Maintainer: BASE SYSTEMS DIAGNOSTIC ENGINEERING

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.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1981 by Digital Equipment Corporation. All Rights Reserved.

The following are trademarks of Digital Equipment Corporation.

DEC	DECsystem-10	DECSYSTEM-20
DECUS	MASSBUS	PDP
UNIBUS	VAX	VMS

digital

Table of Contents

1.0 ABSTRACT 3

2.0 HARDWARE REQUIREMENTS 3

3.0 SOFTWARE REQUIREMENTS 3

4.0 PREREQUISITES 3

5.0 OPERATING INSTRUCTIONS 4
5.1 Event Flags 4
5.2 APT Setup 4

6.0 PROGRAM FUNCTIONAL DESCRIPTION 4
6.1 Program Overview 4
6.2 Program Size 5
6.3 Program Run Times 5
6.4 Fault Detection 5
6.5 Performance During Hardware Failures 5
6.6 Program Applications 6
6.7 Test Descriptions 6

7.0 MAINTENANCE HISTORY 6

1.0 ABSTRACT

This program is a micro-diagnostic designed to test the hardware on the FPA (floating point accelerator) module of the VAX-11/730 processor.

2.0 HARDWARE REQUIREMENTS

This program will run with the minimum hardware configuration of a WCS, CPU, MCT, FPA, and array module.

Maximum main memory allowed is 5 one meg boards, for a total of 5 megabytes of main memory storage.

Other options may be installed without affecting the diagnostic's performance.

3.0 SOFTWARE REQUIREMENTS

This diagnostic must be run under the VAX-11/730 micro monitor (ENKAA) in a standalone environment. The micro-diagnostic is written into WCS RAM, wiping out any system micro-code that may have been resident. It will also use some of main memory during testing. The micro-diagnostic monitor takes up the area where the console software is normally present.

4.0 PREREQUISITES

The WCS, DAP, and MCT modules are assumed to have been tested and known to be good before this diagnostic is run. The programs ENKBA through ENKBE, ENKCA through ENKCD will perform this testing.

5.0 OPERATING INSTRUCTIONS

This program may be executed by typing DI SE ENKCE after the micro-monitor has been loaded and the MIC> prompt has been printed at the terminal. Both the micro-monitor and this program are loaded from a TU58 cassette or downline loaded through the APT-RD port. See the OVERVIEW MANUAL for more information.

5.1 Event Flags

Not applicable

5.2 APT Setup

The micro-monitor knows when APT is present by the position of the keyswitch and the state of a line connected to the APT-RD port. It sets a flag in the CPU Local Store which can be read by this diagnostic. When APT is present, the diagnostic will:

1. Size for the FPA. If no FPA an error will be reported, and if there is an FPA ENKCE will be run.

Under APT, the diagnostic will halt on error and report error information to APT via the mailbox in 8085 RAM.

The following descibes the APT parameters needed to execute this diagnostic:
TBS

6.0 PROGRAM FUNCTIONAL DESCRIPTION

6.1 Program Overview

This program is designed to detect stuck-at faults and provide a repair tool which helps isolate the failing component. A bottom up diagnostic strategy is utilized which builds on previous tests. The tests should be run in consecutive order, to gain the best isolation of a failing component.

6.2 Program Size

This program runs in WCS RAM memory and is approximately 39K bytes long (utilizing about 12700 micro-words in WCS).

6.3 Program Run Times

This program will take approximately 26 seconds to run, including initialization.

6.4 Fault Detection

The diagnostic will report errors with the following header:

```
SECT  TST ERR   EXP   REC  OTHER  MSK  MODULE
```

1. SECT is the program name (ENKCE)
2. TST is the test number that failed.
3. ERR is the error number that failed.
4. EXP is the expected (correct) data.
5. REC is the received (actual) data.
6. OTHER is any other pertinent data (such as an address). This field is not always used. N/A will be printed under OTHER when not in use. The ERRORS section in the listing will describe the contents of this field when it is used.
7. MASK is the error mask used to check the result. Bits set to a 1 in this mask correspond to bits in the result that are not checked.
8. MODULE is the suspected module that caused the failure.

6.5 Performance During Hardware Failures

A power fail will cause a rebooting of the system. Other failures, such as a timeout or a WCS parity error, will print an error message and return to the MIC> prompt. The operator can then issue other commands, including a continue if desired.

6.6 Program Applications

This program can be used by field service to determine which module should be replaced and to verify that the replacement eliminated the failure. It can also be used as a repair tool to help isolate a failing component by manufacturing, field service or engineering. The program is APT compatable.

Customers may used the program to verify the basic integrity of the FPA.

6.7 Test Descriptions

See the actual test in the listing for detailed descriptions and error analyses. A brief description of the logic being tested by each test can be found in the table of contents of the listing.

7.0 MAINTENANCE HISTORY

DATE	VERSION	DESCRIPTION
8-MAR-82	1.0	Initial release.
7-JUN-82	1.1	Changes to Test 42 to test for error F in integer mode. Also error 10 was eliminated due to redundant coverage and errors C & D syntax was changed for better desciption of the faults. Other changes were made to Tests 42 and 43 where subroutines were added to eliminate redundant code.

;
; Table of Contents

; 49 FIELD DEFINITIONS FOR MAIN MICRO WORD VER 00.08
; 1055 MACRO DEFINITIONS VER 00.02
; 1711 FIELD DEFINTITIONS FOR DATA PATH CONTROL MICRO WORD VER 00.01
; 1764 CONTROL ROM CONTENTS VER 00.01
; 2513 DIAGNOSTIC MACROS AND DEFINITIONS
; 3170 COMMON LS ASSIGNMENTS (TO REGISTERS)
; 3323 Microinstruction Formats
; 3706 Tables
; 3837 2901 ALU Control Description
; 3940 DIAGNOSTIC MACROS
; 3964 TEST POINTERS
; 4064 DIAGNOSTIC POINTERS
; 4153 LS DATA SECTION
; 4544 PROGRAM SPECIFIC DATA SECTION (LS 80-F7)
; 4916 MAIN MEMORY DATA SECTION
; 6659 WCS SUBROUTINES FOR CONSOLE USE
; 6660 GENERATE TRANSFER DATA
; 6705 DEPOSIT MCT CSR ROUTINE
; 6771 EXAMINE MCT CSR ROUTINE
; 6858 DEPOSIT MCT TRANSLATION BUFFER ROUTINE
; 6974 EXAMINE TRANSLATION BUFFER ROUTINE
; 7037 DEPOSIT UNIBUS MAP ROUTINE
; 7135 EXAMINE UNIBUS MAP ROUTINE
; 7198 DEPOSIT MAIN MEMORY ROUTINE
; 7256 EXAMINE MAIN MEMORY ROUTINE
; 7318 EXAMINE IDC ROUTINES
; 7384 DEPOSIT IDC ROUTINES
; 7456 SAVE WR ROUTINE
; 7507 RESTORE WR ROUTINE
; 7558 WCS SUBROUTINES FOR CPU USE
; 7559 ERROR ROUTINE
; 7726 START TRANSFER ROUTINE
; 8239 TEST 1 VERIFICATION OF FPA(M8389 FPA MODULE OR CPU MODULE M8390)
; 8350 TEST 2 VERIFICATION OF Y BUS TEST(M8383 FPA MODULE)
; 8434 TEST 3 NUA LINES TEST (M8389 FPA MODULE)
; 8630 TEST 4 PARITY LOGIC TEST (M8389 FPA MODULE)
; 8780 TEST 5 CONTROL STORE TEST (M8389 FPA MODULE)
; 8952 TEST 6 CLOCK SYNC TEST(M8389 FPA OR M8390 CPU MODULE)
; 9029 TEST 7 CPU FORCE TEST DURING FPA FAST CLOCK(M8389 FPA MODULE)
; 9133 TEST 8 CPU DATA AVAIL AND ZERO BRANCH TEST(M8389 FPA OR M8390 CPU MODULE)
; 9225 TEST 9 ZEROS FROM THE 2901 TEST(M8389 FPA MODULE)
; 9356 TEST A WORKING REGISTER TEST (M8389 FPA MODULE)
; 9667 TEST B EXPONENT DATA PATH TEST (M8389 FPA MODULE)
; 9951 TEST C EVEN EXTENSION WORKING REGISTER TEST(M8389 FPA MODULE)
; 10138 TEST D ODD EXTENSION WORKING REGISTER TEST (M8389 FPA MODULE)
; 10260 TEST E REMAINING EXPONENT REGISTER TEST(M8389 FPA MODULE)
; 10450 TEST F ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)
; 10676 TEST 10 EXTENSION DATA PATH BIT ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)
; 10997 TEST 11 UPPER BIT EXPONENT DATA PATH BIT ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)
; 11275 TEST 12 HUGE AND GRAND LOAD TEST(M8389 FPA MODULE)
; 11555 TEST 13 AND FUNCTION, A.Q SRC TEST(M8389 FPA MODULE)
; 11805 TEST 14 R NOT AND S FUNCTION, A.B SRC TEST(M8389 FPA MODULE)
; 12087 TEST 15 R XOR S FUNCTION TEST(M8389 FPA MODULE)
; 12368 TEST 16 XNOR FUNCTION TEST(M8389 FPA MODULE)
; 12647 TEST 17 R PLUS S FUNCTION TEST(M8389 FPA MODULE)

;
; Table of Contents

; 13229 TEST 18 S MINUS R FUNCTION TEST(M8389 FPA MODULE)
; 13699 TEST 19 R MINUS S FUNCTION TEST(M8389 FPA MODULE)
; 14115 TEST 1A 2F -> B DESTINATION TEST
; 14311 TEST 1B F/2 -> B DESTINATION TEST
; 14506 TEST 1C 2F -> B, 2Q -> Q DESTINATION TEST
; 14688 TEST 1D F/2 -> B, Q/2 -> Q DESTINATION TEST
; 14870 TEST 1E OR FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)
; 15111 TEST 1F ADD FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)
; 15277 TEST 20 Q+1 FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)
; 15428 TEST 21 Q-1 FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)
; 15586 TEST 22 Q-A FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)
; 15763 TEST 23 A-Q FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)
; 15956 TEST 24 CLR FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)
; 16088 TEST 25 96 BIT FLOATING DATA TEST(M8389 FPA MODULE)
; 16205 TEST 26 SHIFT FIELD TEST(M8389 FPA MODULE)
; 17102 TEST 27 SIGN CONTROL TEST(M8389 FPA MODULE)
; 17563 TEST 28 CC CONDITION TEST(M8389 FPA MODULE)
; 17890 TEST 29 ENABLE LITERAL TEST(M8389 FPA MODULE)
; 18112 TEST 2A EXP COUT AND GRAND BRANCH TEST(M8389 FPA MODULE)
; 18308 TEST 2B SIGN OUT AND HUGE BRANCH TEST(M8389 FPA MODULE)
; 18411 TEST 2C CPU DATA AVAIL AND SINGLE BRANCH TEST(M8389 FPA MODULE)
; 18527 TEST 2D OPTION SYNC TEST(M8389 FPA MODULE, M8390 CPU MODULE)
; 18708 TEST 2E CPU DATA AVAIL AND ADD + SUB BRANCH TEST(M8389 FPA MODULE)
; 18886 TEST 2F FRAC COUT AND EXT FUNC BRANCH TEST(M8389 FPA MODULE)
; 19038 TEST 30 OP1 SIGN AND EMOD BRANCH TEST(M8389 FPA MODULE)
; 19221 TEST 31 FRAC55 F3 and SINGLE BRANCH TEST(M8389 FPA MODULE)
; 19327 TEST 32 OP2 SIGN AND ADD + SUB BRANCH TEST(M8389 FPA MODULE)
; 19499 TEST 33 EXP COUT AND EXP15 F3 BRANCH TEST(M8389 FPA MODULE)
; 19614 TEST 34 SIGN OUT AND OP2=0 BRANCH TEST(M8389 FPA MODULE)
; 19792 TEST 35 FRAC55 F3 BRANCH TEST(M8389 FPA MODULE)
; 19899 TEST 36 F47, F3 and EXT00 Q0 BRANCH TEST(M8389 FPA MODULE)
; 20008 TEST 37 FRAC<47:16>=0 AND ZERO BRANCH TEST(M8389 FPA MODULE)
; 20234 TEST 38 FRAC<55:00>=0 AND CPU RCV DATA BRANCH TEST(M8389 FPA MODULE)
; 20509 TEST 39 NULL BRANCH AND CPU RCV DATA BRANCH TEST(M8389 FPA MODULE)
; 20620 TEST 3A FRAC<55:7>=0 BRANCH TEST(M8389 FPA MODULE)
; 20834 TEST 3B EXPONENT=0 AND EXP15 F3 BRANCH TEST(M8389 FPA MODULE)
; 20941 TEST 3C OP1=0 AND OP2=0 BRANCH TEST(M8389 FPA MODULE)
; 21096 TEST 3D CALL SUBROUTINE BRANCH TEST(M8389 FPA MODULE)
; 21407 TEST 3E RETURN (OP1+OP2)/=0 AND RETURN EXP15 F3 TEST
; 21563 TEST 3F SUMPATH BRANCH TEST(M8389 FPA MODULE)
; 21960 TEST 40 EXTENDED BRANCH TEST(M8389 FPA MODULE)
; 22415 TEST 41 MUL I1 AND FRAC55 Q3 BRANCH TEST(M8389 FPA MODULE)
; 22931 TEST 42 FRAC<55:32>=0 AND DIV I3 BRANCH TEST(M8389 FPA MODULE)
; 23455 TEST 43 MUL AND DIV LOGIC TEST(M8389 FPA MODULE)
; 25346 TEST 44 INSTRUCTION ROM TEST(M8389 FPA MODULE)
; 25668 TEST 45 CLOCK SYNC TEST VIA TOGGLE CLOCK(M8389 FPA MODULE)

```
                ;1      .TITLE  "ENKCE Micro-diagnostic for FPA module  Rev 01.10"
                ;2      ;
                ;3      ;
                ;4      ;                       COPYRIGHT (c) 1982 BY
                ;5      ;                DIGITAL EQUIPMENT CORPORATION, MAYNARD,
                ;6      ;                 MASSACHUSETTS.  ALL RIGHTS RESERVED.
                ;7      ;
                ;8      ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY  BE  USED  AND  COPIED
                ;9      ; ONLY  IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE INCLUSION
                ;10     ; OF THE ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE OR ANY OTHER COPIES THEREOF
                ;11     ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER  PERSON.  NO
                ;12     ; TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY TRANSFERRED.
                ;13     ;
                ;14     ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE, AND
                ;15     ; SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
                ;16     ;
                ;17     ; DIGITAL  ASSUMES  NO  RESPONSIBILITY  FOR  THE  USE OR RELIABILITY OF ITS
                ;18     ; SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.
                ;19     ;
                ;20
                ;21     ;++
                ;22     ; FACILITY:     VAX-11/730 MICRO-DIAGNOSTIC.
                ;23     ;
                ;24     ; ABSTRACT:     This micro-diagnostic tests the hardware of the FPA module
                ;25     ;               in the VAX-11/730 processor.
                ;26     ;
                ;27     ; ENVIRONMENT:  ENKAA  Micro-diagnostic Monitor
                ;28     ;
                ;29     ; AUTHOR:       Cynthia Oka     29-MAY-81
                ;30     ;
                ;31     ; MODIFIED BY:  David Mayo      8-MAR-82        VERSION 01.00
                ;32     ;               Initial release.
                ;33     ;
                ;34     ;               Ernie Preisig   4-JUN-82        VERSION 01.10
                ;35     ;               Fixed bug in test 42, shortened tests by
                ;36     ;               adding subroutines
                ;37     ;--
                ;38     .nolist
                ;42     .list
                ;43     .NOCREF
;44     .NOBIN
;45     .SEQUENTIAL
;46     .RTOL
;47     .HEXADECIMAL
```

```
;48      .PAGE
;49      .TOC    "FIELD DEFINITIONS FOR MAIN MICRO WORD                  VER 00.08"
;50      .SET/UPC=<000>
;51
;52      ; THE FOLLOWING EXPRESSIONS ARE VALIDITY CHECKS FOR FIELD COMBINATIONS
;53
;54      .SET/A.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0]>
;55      .SET/B.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
;56      .SET/D.VAL=<.CASE[<OPC2/>]OF[0,0,0,1,0,0,0,0,1,1,1,1,1,1,1,1]>
;57      .SET/XD.VAL=<.EQL[<OPC1/>,<OPC1/MOVE>]>
;58      .SET/CC.VAL=<.CASE[<OPC2/>]OF[0,0,0,0,1,1,1,1,1,1,1,1,1,1,1,1]>
;59      .SET/DT.VAL=<.EQL[<OPC2/>,<OPC2/MEM.REQ>]>
;60      .SET/SCTL.VAL=<.CASE[<OPC2/>]OF[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1]>
;61      .SET/JCTL.VAL=<.CASE[<OPC2/>]OF[1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0]>
;62      .SET/JUMP.VAL=<.EQL[<OPC2/>,<OPC2/JUMP>]>
;63      .SET/DECODE.VAL=<.EQL[<OPC2/>,<OPC2/DECODE>]>
;64      .SET/MISC.VAL=<.EQL[<OPC2/>,<OPC2/MISC>]>
;65      .SET/DP.VAL=<.EQL[<OPC/>,<OPC/BASIC>]>
;66      ;
;67      ;       THE FOLLOWING VALIDITY CHECKS ARE USE FOR THE PAGE BOUNDARY PROBLEM.
;68      ;
;69      .SET/PAGE.PROB=<.EQL[<.AND[<7FF>,<.>]>,<7FE>]>
;70      .SET/SKIP.LEGAL2=<.OR[<.EQL[<SCTL/>,<SCTL/RET.NOERR>]>,<.EQL[<SCTL/>,
;71      <SCTL/POP.USTACK>]>]>
;72      .SET/SKIP.LEGAL=<.OR[<.EQL[<SCTL/>,<SCTL/NO.SKIP>]>,<.EQL[<SCTL/>,
;73      <SCTL/RETURN>]>,<.EQL[<SCTL/>,<SCTL/RETURN+1>]>,<SKIP.LEGAL2>]>
;74      .SET/SKIP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<SKIP.LEGAL>]>
;75      .SET/JUMP.CHECK=<.OR[<.EQL[<JCTL/>,<JCTL/JSR>]>,<.EQL[<JCTL/>,
;76      <JCTL/JSR.VALID>]>]>
;77      .SET/JUMP.PAGE.VAL=<.CASE[<PAGE.PROB>]OF[1,<.XOR[1,<JUMP.CHECK>]>]>
;78
;79
;80      ; MICRO INSTRUCTION OPCODE DEFINITIONS
;81      ;
;82      OPC/=<22>,.DEFAULT=<OPC/BASIC>  ; MICRO OPCODE FIELD FOR SINGLE BIT OPCODE.
;83
;84              BASIC=1                 ; OPCODE FOR 'BASIC' MICRO INSTRUCTION
;85
;86      OPC1/=<22:20>                   ; MICRO OPCODE FIELD FOR THREE BIT OPCODES.
;87
;88              EXTENDED=2              ; OPCODE FOR 'EXTENDED' MICRO INSTRUCTION
;89              MOVE=3                  ; OPCODE FOR 'MOVE' MICRO INSTRUCTION
;90
;91      OPC2/=<22:19>                   ; MICRO OPCODE FIELD FOR FOUR BIT OPCODES
;92
;93              MEM.REQ=3               ; OPCODE FOR 'MEM.REQ' MICRO INSTRUCTION
;94              MISC=2                  ; OPCODE FOR 'MISC' MICRO INSTRUCTION
;95              JUMP=1                  ; OPCODE FOR 'JUMP' MICRO INSTRUCTION
;96              DECODE=0                ; OPCODE FOR 'DECODE' MICRO INSTRUCTION
;97
;98
;99      ; THE FOLLOWING FOUR FIELDS ARE RELATED TO ADDRESSING
;100     ; VARIOUS RAM MEMORIES WITHIN THE DATA PATH.
;101     ;
;102     ; THE A.ADRS  REFERS TO THE 2901 INTERNAL RAM 'A-PORT'
```

```
;103     ; THE B.ADRS  REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
;104     ; THE XB.ADRS REFERS TO THE 2901 INTERNAL RAM 'B-PORT'
;105     ; THE D.ADRS  REFERS TO THE LOWER 128 LOCATIONS OF THE LOCAL STORE RAM
;106     ; THE XD.ADRS REFERS TO   ALL     256 LOCATIONS OF THE LOCAL STORE RAM
;107     ;
;108     ; THE VALIDITY STATEMENTS ALLOW USAGE OF THESE FIELDS IN THE FOLLOWING MANNER
;109     ;
;110     ;       A.ADRS          EXTENDED  MICRO INSTRUCTION
;111     ;
;112     ;       B.ADRS          BASIC     MICRO INSTRUCTION
;113     ;                       EXTENDED  MICRO INSTRUCTION
;114     ;                       MOVE      MICRO INSTRUCTION
;115     ;                       DECODE.IS MICRO INSTRUCTION
;116     ;
;117     ;       XB.ADRS         MOVE XWR  MICRO INSTRUCTION
;118     ;
;119     ;       D.ADRS          BASIC     MICRO INSTRUCTION
;120     ;                       MEM.REQ   MICRO INSTRUCTION
;121     ;
;122     ;       XD.ADRS         MOVE      MICRO INSTRUCTION
;123     ;
;124     A.ADRS/=<10:9>,.VALIDITY=<A.VAL>
;125
;126             MASK=3                    ; REGISTER BACKUP MASK
;127
;128     B.ADRS/=<8:7>,.VALIDITY=<B.VAL>,.DEFAULT=0
;129
;130             MASK=3                    ; REGISTER BACKUP MASK
;131
;132     ; THIS ADDRESS FIELD WILL CONTAIN THE LOW TWO BITS OF THE 2901 B ADDRESS
;133     ; THE THIRD BIT WILL BE FORCED BY HARDWARE.
;134     ;
;135     XB.ADRS/=<8:7>,.VALIDITY=<.EQL[<OPC1/>,<OPC1/MOVE>]>
;136
;137             BPC=0                     ; ADDRESS OF BEGINNING PC     WR[4]
;138             VA=1                      ; ADDRESS OF MEMORY REFERENCE WR[5]
;139             VAR=1                     ; ADDRESS OF LAST MEMORY REFERENCE. (WR[5] IN 2901).
;140
;141     ;D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
;142
;143
;144     ;       SAVED.2ND.REQUEST=0       ; MM SAVED STATE FOR REQUEST.
;145     ;       T1=1                      ; TEMP LOCATION 1
;146     ;       T2=2                      ; TEMP LOCATION 2
;147     ;       T3=3                      ; TEMP LOCATION 3
;148     ;       T4=4                      ; TEMP LOCATION 4
;149     ;       T5=5                      ; TEMP LOCATION 5
;150     ;       T6=6                      ; TEMP LOCATION 6
;151     ;       T7=7                      ; TEMP LOCATION 7
;152     ;       T8=8                      ; TEMP LOCATION 8
;153     ;       T9=9                      ; TEMP LOCATION 9
;154     ;       T10=0A                    ; TEMP LOCATION 10
;155     ;       BACKUP.CM.PC=0A           ;COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
;156     ;
;157     ;       T11=0B                    ; TEMP LOCATION 11
```

```
;158     ;       T12=0C                  ; TEMP LOCATION 12
;159     ;       T13=0D                  ; TEMP LOCATION 13
;160     ;       T14=0E                  ; TEMP LOCATION 14
;161     ;       T15=0F                  ; TEMP LOCATION 15
;162     ;       PC=10                   ; MACRO PROGRAM COUNTER
;163     ;       WR.BACKUP=11            ; BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[]
;164     ;       SAVED.VAR=12            ; MM SAVED STATE FOR VAR
;165     ;       SAVED.DATA=13           ; MM SAVED STATE FOR DATA
;166     ;       SAVED.PTE.ADDRESS=14    ; MM SAVED STATE FOR PAGE TABLE ENTRIES ADDRESS
;167     ;       SAVED.PTE=15            ; MM SAVED STATE FOR PAGE TABLE ENTRIES
;168     ;       T16=16                  ; TEMP LOCATION 16
;169     ;       T17=17                  ; TEMP LOCATION 17
;170     ;       IE.TEMP4=18             ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
;171     ;       #FFFFF000=19                    ; CONSTANT
;172     ;       #FFFFFF00=1A                    ; CONSTANT
;173     ;       #FFFFFFFC=1B                    ; CONSTANT
;174     ;       #FFFFFFFF=1C                    ; CONSTANT
;175     ;
;176     ;       THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL.  WHENEVER THE
;177     ;       HARDWARE VERSION OF THE PSL IS CHANGED (PSL.HW), THEN THIS
;178     ;       CHANGE IS ALSO REPORTED HERE BY THE MICROCODE.  ALL BITS OF
;179     ;       THE PSL ARE REALLY LIKE THOSE IN THIS WORD EXCEPT FOR THE
;180     ;       CONDITION CODES WHICH MUST BE OBTAINED FROM PSL.CC
;181     ;
;182     ;       PSL=1D                  ; SOFT COPY OF VAX PSL
;183     ;       VAR=1E                  ; VIRTUAL ADDRESS REGISTER
;184     ;       VAR2=1F                 ; VIRTUAL ADDRESS REGISTER
;185     ;
;186     ;
;187     ;       #1=20                   ; MASK 00000001 (HEX)
;188     ;       #2=21                   ; MASK 00000002
;189     ;       #4=22                   ; MASK 00000004
;190     ;       #8=23                   ; MASK 00000008
;191     ;       #10=24                  ; MASK 00000010
;192     ;       #20=25                  ; MASK 00000020
;193     ;       #40=26                  ; MASK 00000040
;194     ;       #80=27                  ; MASK 00000080
;195     ;       #100=28                 ; MASK 00000100
;196     ;       #200=29                 ; MASK 00000200
;197     ;       #400=2A                 ; MASK 00000400
;198     ;       #800=2B                 ; MASK 00000800
;199     ;       #1000=2C                ; MASK 00001000
;200     ;       #2000=2D                ; MASK 00002000
;201     ;       #4000=2E                ; MASK 00004000
;202     ;       #8000=2F                ; MASK 00008000
;203     ;       #10000=30               ; MASK 00010000
;204     ;       #20000=31               ; MASK 00020000
;205     ;       #40000=32               ; MASK 00040000
;206     ;       #80000=33               ; MASK 00080000
;207     ;       #100000=34              ; MASK 00100000
;208     ;       #200000=35              ; MASK 00200000
;209     ;       #400000=36              ; MASK 00400000
;210     ;       #800000=37              ; MASK 00800000
;211     ;       #1000000=38             ; MASK 01000000
;212     ;       #2000000=39             ; MASK 02000000
```

```
;213     ;       #4000000=3A             ; MASK 04000000
;214     ;       #8000000=3B             ; MASK 08000000
;215     ;       #10000000=3C            ; MASK 10000000
;216     ;       #20000000=3D            ; MASK 20000000
;217     ;       #40000000=3E            ; MASK 40000000
;218     ;       #80000000=3F            ; MASK 80000000
;219     ;
;220     ;       BIT0=20                 ; MASK 00000001
;221     ;       BIT1=21                 ; MASK 00000002
;222     ;       BIT2=22                 ; MASK 00000004
;223     ;       BIT3=23                 ; MASK 00000008
;224     ;       BIT4=24                 ; MASK 00000010
;225     ;       BIT5=25                 ; MASK 00000020
;226     ;       BIT6=26                 ; MASK 00000040
;227     ;       BIT7=27                 ; MASK 00000080
;228     ;       BIT8=28                 ; MASK 00000100
;229     ;       BIT9=29                 ; MASK 00000200
;230     ;       BIT10=2A                ; MASK 00000400
;231     ;       BIT11=2B                ; MASK 00000800
;232     ;       BIT12=2C                ; MASK 00001000
;233     ;       BIT13=2D                ; MASK 00002000
;234     ;       BIT14=2E                ; MASK 00004000
;235     ;       BIT15=2F                ; MASK 00008000
;236     ;       BIT16=30                ; MASK 00010000
;237     ;       BIT17=31                ; MASK 00020000
;238     ;       BIT18=32                ; MASK 00040000
;239     ;       BIT19=33                ; MASK 00080000
;240     ;       BIT20=34                ; MASK 00100000
;241     ;       BIT21=35                ; MASK 00200000
;242     ;       BIT22=36                ; MASK 00400000
;243     ;       BIT23=37                ; MASK 00800000
;244     ;       BIT24=38                ; MASK 01000000
;245     ;       BIT25=39                ; MASK 02000000
;246     ;       BIT26=3A                ; MASK 04000000
;247     ;       BIT27=3B                ; MASK 08000000
;248     ;       BIT28=3C                ; MASK 10000000
;249     ;       BIT29=3D                ; MASK 20000000
;250     ;       BIT30=3E                ; MASK 40000000
;251     ;       BIT31=3F                ; MASK 80000000
;252     ;
;253     ;       GPR0=40                 ; GPR 0
;254     ;       GPR1=41                 ; GPR 1
;255     ;       GPR2=42                 ; GPR 2
;256     ;       GPR3=43                 ; GPR 3
;257     ;       GPR4=44                 ; GPR 4
;258     ;       GPR5=45                 ; GPR 5
;259     ;       GPR6=46                 ; GPR 6
;260     ;       CM.SP=46                ; IN COMPATIBILITY THIS IS THE STACK POINTER.
;261     ;       GPR7=47                 ; GPR 7
;262     ;       CM.PC=47                ; IN COMPATIBILITY MODE THIS IS THE PC.
;263     ;       GPR8=48                 ; GPR 8
;264     ;       GPR9=49                 ; GPR 9
;265     ;       GPR10=4A                ; GPR 10
;266     ;       GPR11=4B                ; GPR 11
;267     ;       GPR12=4C                ; GPR 12
```

```
;268     ;       AP=4C                   ; ARGUMENT POINTER
;269     ;       GPR13=4D                ; GPR 13
;270     ;       FP=4D                   ; FRAME POINTER
;271     ;       SP=4E                   ; GPR 14
;272     ;       TO=4F                   ; TEMP LOCATION 0
;273     ;
;274     ;       ZERO=50                 ; ZERO CONSTANT
;275     ;       #3=51           ; CONSTANT
;276     ;       #5=52           ; CONSTANT
;277     ;       #6=53           ; CONSTANT
;278     ;       #7=54           ; CONSTANT
;279     ;       #9=55           ; CONSTANT
;280     ;       #B=56           ; CONSTANT
;281     ;       #C=57           ; CONSTANT
;282     ;       #D=58           ; CONSTANT
;283     ;       #E=59           ; CONSTANT
;284     ;       #F=5A           ; CONSTANT
;285     ;       #13=5B          ; CONSTANT
;286     ;       #1F=5C          ; CONSTANT
;287     ;       #34=5D          ; CONSTANT
;288     ;       #3B=5E          ; CONSTANT
;289     ;       #3F=5F          ; CONSTANT
;290     ;       #4B=60          ; CONSTANT
;291     ;       #66=61          ; CONSTANT
;292     ;       #A0=62          ; CONSTANT
;293     ;       #F0=63          ; CONSTANT
;294     ;       #FF=64          ; CONSTANT
;295     ;       #1FF=65         ; CONSTANT
;296     ;       #FFF=66         ; CONSTANT
;297     ;       #2001=67                ; CONSTANT
;298     ;       #3000=68                ; CONSTANT
;299     ;       #7F80=69                ; CONSTANT
;300     ;       #C000=6A                ; CONSTANT
;301     ;       #FFE0=6B                ; CONSTANT
;302     ;       #FFFF=6C                ; CONSTANT
;303     ;       #1F0000=6D              ; CONSTANT
;304     ;       #200001=6E              ; CONSTANT
;305     ;       #F27000=6F              ; CONSTANT
;306     ;       #3000000=70             ; CONSTANT
;307     ;       #41F0000=71             ; CONSTANT
;308     ;       #7000000=72             ; CONSTANT
;309     ;       #3FFFFE00=73            ; CONSTANT
;310     ;       #7F800000=74            ; CONSTANT
;311     ;       #7FFFFE00=75            ; CONSTANT
;312     ;       #7FFFFF0E=76            ; CONSTANT
;313     ;       #BF800001=77            ; CONSTANT
;314     ;
;315     ;       GPR.OS=78               ; HARDWARE INDEX TO GPRS
;316     ;       BIT.OS(3-0)=79          ; 16 LOCATION HARDWARE INDEX TO BIT MASKS
;317     ;       BIT.OS(4-0)=7B          ; 32 LOCATION HARDWARE INDEX TO BIT MASKS
;318     ;       OS=7C                   ; OS REGISTER
;319     ;       DEC.CON=7D              ; DECIMAL CARRIES
;320     ;       SIZE=7E                 ; SIZE REGISTER
;321     ;       MDT= 7E                 ; MEMORY REFERENCE DATA SIZE
;322     ;       INTERRUPT.VEC=7E        ; HARDWARE INTERRUPT VECTOR
```

```
;323     ;
;324     ;       THIS WORD IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE READ
;325     ;       OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;326     ;
;327     ;       PSL.CC=7F               ; PSL CONDITION CODES
;328     ;
;329     ;XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
;330     ;
;331     ;       SAVED.2ND.REQUEST=0     ; MM SAVED STATE FOR REQUEST.
;332     ;       T1=1                    ; TEMP LOCATION 1
;333     ;       T2=2                    ; TEMP LOCATION 2
;334     ;       T3=3                    ; TEMP LOCATION 3
;335     ;       T4=4                    ; TEMP LOCATION 4
;336     ;       T5=5                    ; TEMP LOCATION 5
;337     ;       T6=6                    ; TEMP LOCATION 6
;338     ;       T7=7                    ; TEMP LOCATION 7
;339     ;       T8=8                    ; TEMP LOCATION 8
;340     ;       T9=9                    ; TEMP LOCATION 9
;341     ;       T10=0A                  ; TEMP LOCATION 10
;342     ;       BACKUP.CM.PC=0A         ;COMPATIBILITY MODE KEEPS ORIGINAL PC HERE.
;343     ;       T11=0B                  ; TEMP LOCATION 11
;344     ;       T12=0C                  ; TEMP LOCATION 12
;345     ;       T13=0D                  ; TEMP LOCATION 13
;346     ;       T14=0E                  ; TEMP LOCATION 14
;347     ;       T15=0F                  ; TEMP LOCATION 15
;348     ;       PC=10                   ; MACRO INSTRUCTION PC
;349     ;       WR.BACKUP=11            ; BACKUP WORKING REGISTER FOR MOV MEM.DATA TO WR[].
;350     ;       SAVED.VAR=12            ; MM SAVED VAR LOCATION
;351     ;       SAVED.DATA=13           ; MM SAVED DATA LOCATION
;352     ;       SAVED.PTE.ADDRESS=14    ; MM SAVED PAGE TABLE ENTRY ADDRESS
;353     ;       SAVED.PTE=15            ; MM SAVED PAGE TABLE ENTRY
;354     ;       T16=16                  ; TEMP LOCATION 16
;355     ;       T17=17                  ; TEMP LOCATION 17
;356     ;       IE.TEMP4=18             ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION FOUR.
;357     ;       #FFFFF000=19                    ; CONSTANT
;358     ;       #FFFFFF00=1A                    ; CONSTANT
;359     ;       #FFFFFFFC=1B                    ; CONSTANT
;360     ;       #FFFFFFFF=1C                    ; CONSTANT
;361     ;
;362     ;       THIS LOCATION CONTAINS THE SOFT COPY OF THE PSL.  WHENEVER THE HARDWARE
;363     ;       VERSION OF THE PSL IS CHANGED (PSL.HW) THEN THIS CHANGE IS
;364     ;       REPORTED HERE.  ALL BITS OF THE PSL ARE REALLY LIKE THOSE HERE
;365     ;       EXCEPT FOR THE CONDITION CODES WHICH MUST BE OBTAINED
;366     ;       FROM PSL.CC.
;367     ;
;368     ;       PSL=1D                  ; SOFT COPY OF VAX PSL
;369     ;       VAR=1E                  ; VIRTUAL ADDRESS REGISTER
;370     ;       VAR2=1F                 ; VIRTUAL ADDRESS REGISTER
;371     ;
;372     ;
;373     ;       #1=20                   ; MASK 00000001 (HEX)
;374     ;       #2=21                   ; MASK 00000002
;375     ;       #4=22                   ; MASK 00000004
;376     ;       #8=23                   ; MASK 00000008
;377     ;       #10=24                  ; MASK 00000010
```

```
;378     ;       #20=25                  ; MASK 00000020
;379     ;       #40=26                  ; MASK 00000040
;380     ;       #80=27                  ; MASK 00000080
;381     ;       #100=28                 ; MASK 00000100
;382     ;       #200=29                 ; MASK 00000200
;383     ;       #400=2A                 ; MASK 00000400
;384     ;       #800=2B                 ; MASK 00000800
;385     ;       #1000=2C                ; MASK 00001000
;386     ;       #2000=2D                ; MASK 00002000
;387     ;       #4000=2E                ; MASK 00004000
;388     ;       #8000=2F                ; MASK 00008000
;389     ;       #10000=30               ; MASK 00010000
;390     ;       #20000=31               ; MASK 00020000
;391     ;       #40000=32               ; MASK 00040000
;392     ;       #80000=33               ; MASK 00080000
;393     ;       #100000=34              ; MASK 00100000
;394     ;       #200000=35              ; MASK 00200000
;395     ;       #400000=36              ; MASK 00400000
;396     ;       #800000=37              ; MASK 00800000
;397     ;       #1000000=38             ; MASK 01000000
;398     ;       #2000000=39             ; MASK 02000000
;399     ;       #4000000=3A             ; MASK 04000000
;400     ;       #8000000=3B             ; MASK 08000000
;401     ;       #10000000=3C            ; MASK 10000000
;402     ;       #20000000=3D            ; MASK 20000000
;403     ;       #40000000=3E            ; MASK 40000000
;404     ;       #80000000=3F            ; MASK 80000000
;405     ;
;406     ;       BIT0=20                 ; MASK 00000001
;407     ;       BIT1=21                 ; MASK 00000002
;408     ;       BIT2=22                 ; MASK 00000004
;409     ;       BIT3=23                 ; MASK 00000008
;410     ;       BIT4=24                 ; MASK 00000010
;411     ;       BIT5=25                 ; MASK 00000020
;412     ;       BIT6=26                 ; MASK 00000040
;413     ;       BIT7=27                 ; MASK 00000080
;414     ;       BIT8=28                 ; MASK 00000100
;415     ;       BIT9=29                 ; MASK 00000200
;416     ;       BIT10=2A                ; MASK 00000400
;417     ;       BIT11=2B                ; MASK 00000800
;418     ;       BIT12=2C                ; MASK 00001000
;419     ;       BIT13=2D                ; MASK 00002000
;420     ;       BIT14=2E                ; MASK 00004000
;421     ;       BIT15=2F                ; MASK 00008000
;422     ;       BIT16=30                ; MASK 00010000
;423     ;       BIT17=31                ; MASK 00020000
;424     ;       BIT18=32                ; MASK 00040000
;425     ;       BIT19=33                ; MASK 00080000
;426     ;       BIT20=34                ; MASK 00100000
;427     ;       BIT21=35                ; MASK 00200000
;428     ;       BIT22=36                ; MASK 00400000
;429     ;       BIT23=37                ; MASK 00800000
;430     ;       BIT24=38                ; MASK 01000000
;431     ;       BIT25=39                ; MASK 02000000
;432     ;       BIT26=3A                ; MASK 04000000
```

```
;433     ;       BIT27=3B                ; MASK 08000000
;434     ;       BIT28=3C                ; MASK 10000000
;435     ;       BIT29=3D                ; MASK 20000000
;436     ;       BIT30=3E                ; MASK 40000000
;437     ;       BIT31=3F                ; MASK 80000000
;438     ;
;439     ;       GPR0=40                 ; GPR 0
;440     ;       GPR1=41                 ; GPR 1
;441     ;       GPR2=42                 ; GPR 2
;442     ;       GPR3=43                 ; GPR 3
;443     ;       GPR4=44                 ; GPR 4
;444     ;       GPR5=45                 ; GPR 5
;445     ;       GPR6=46                 ; GPR 6
;446     ;       CM.SP=46                ; IN COMPATIBILITY MODE THIS IS THE STACK POINTER.
;447     ;       GPR7=47                 ; GPR 7
;448     ;       CM.PC=47                ; IN COMPATIBILITY MODE THIS IS THE PC.
;449     ;       GPR8=48                 ; GPR 8
;450     ;       GPR9=49                 ; GPR 9
;451     ;       GPR10=4A                ; GPR 10
;452     ;       GPR11=4B                ; GPR 11
;453     ;       GPR12=4C                ; GPR 12
;454     ;       AP=4C                   ; ARGUMENT POINTER
;455     ;       GPR13=4D                ; GPR 13
;456     ;       FP=4D                   ; FRAME POINTER
;457     ;       SP=4E                   ; GPR 14
;458     ;       T0=4F                   ; TEMP LOCATION 0
;459     ;
;460     ;       ZERO=50                 ; ZERO CONSTANT
;461     ;       #3=51           ; CONSTANT
;462     ;       #5=52           ; CONSTANT
;463     ;       #6=53           ; CONSTANT
;464     ;       #7=54           ; CONSTANT
;465     ;       #9=55           ; CONSTANT
;466     ;       #B=56           ; CONSTANT
;467     ;       #C=57           ; CONSTANT
;468     ;       #D=58           ; CONSTANT
;469     ;       #E=59           ; CONSTANT
;470     ;       #F=5A           ; CONSTANT
;471     ;       #13=5B          ; CONSTANT
;472     ;       #1F=5C          ; CONSTANT
;473     ;       #34=5D          ; CONSTANT
;474     ;       #3B=5E          ; CONSTANT
;475     ;       #3F=5F          ; CONSTANT
;476     ;       #4B=60          ; CONSTANT
;477     ;       #66=61          ; CONSTANT
;478     ;       #A0=62          ; CONSTANT
;479     ;       #F0=63          ; CONSTANT
;480     ;       #FF=64          ; CONSTANT
;481     ;       #1FF=65         ; CONSTANT
;482     ;       #FFF=66         ; CONSTANT
;483     ;       #2001=67                ; CONSTANT
;484     ;       #3000=68                ; CONSTANT
;485     ;       #7F80=69                ; CONSTANT
;486     ;       #C000=6A                ; CONSTANT
;487     ;       #FFE0=6B                ; CONSTANT
```

```
;488     ;       #FFFF=6C                ; CONSTANT
;489     ;       #1F0000=6D              ; CONSTANT
;490     ;       #200001=6E              ; CONSTANT
;491     ;       #F27000=6F              ; CONSTANT
;492     ;       #3000000=70             ; CONSTANT
;493     ;       #41F0000=71             ; CONSTANT
;494     ;       #7000000=72             ; CONSTANT
;495     ;       #3FFFFE00=73            ; CONSTANT
;496     ;       #7F800000=74            ; CONSTANT
;497     ;       #7FFFFE00=75            ; CONSTANT
;498     ;       #7FFFFF0E=76            ; CONSTANT
;499     ;       #BF800001=77            ; CONSTANT
;500     ;
;501     ;       GPR.OS=78               ; HARDWARE INDEX TO GPRS
;502     ;       BIT.OS(3-0)=79          ; 16 LOCATION HARDWARE INDEX TO BIT MASKS
;503     ;       BIT.OS(4-0)=7B          ; 32 LOCATION HARDWARE INDEX TO BIT MASKS
;504     ;       OS=7C                   ; OS REGISTER
;505     ;       DEC.CON=7D              ; DECIMAL CARRIES
;506     ;       SIZE=7E                 ; SIZE REGISTER
;507     ;       MDT= 7E                 ; MEMORY REFERENCE DATA SIZE
;508     ;       INTERRUPT.VEC=7E        ; HARDWARE INTERRUPT VECTOR
;509     ;
;510     ;       THIS LOCATION IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE
;511     ;       READ OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;512     ;
;513     ;       PSL.CC=7F               ; PSL CONDITION CODES
;514
;515     ; THESE ADDRESSES ARE ONLY ACCESSIBLE WITH THE MOV MICRO INSTRUCTION.
;516
;517     ;       KSP=80                  ; KERNEL STACK POINTER
;518     ;       ESP=81                  ; EXECUTIVE STACK POINTER
;519     ;       SSP=82                  ; SUPERVISOR STACK POINTER
;520     ;       USP=83                  ; USER STACK POINTER
;521     ;       ISP=84                  ; INTERRUPT STACK POINTER
;522     ;       ASTLVL=85               ; AST LEVEL
;523     ;       PCBB=86                 ; PROCESS CONTROL BLOCK BASE
;524     ;       SCBB=87                 ; SYSTEM CONTROL BLOCK BASE
;525     ;       SISR=88                 ; SOFTWARE INTERRUPT SUMMARY REGISTER
;526     ;
;527     ;       CONSOLE.COMMAND=89      ; ONE BYTE COMMAND.
;528     ;       CONSOLE.MODIFIER=8A     ; ONE BYTE MODIFIER.
;529     ;       CONSOLE.ADDRESS=8B      ; FOUR BYTES OF ADDRESS.
;530     ;       CONSOLE.DATA=8C         ; FOUR BYTES OF DATA.
;531     ;       CONSOLE.STATUS=8D       ; ONE BYTE OF STATUS.
;532     ;       PSEUDO.NICR=8E          ; COPY OF NICR THAT 8085 HAS.
;533     ;       PSEUDO.ICR=8F           ; COPY OF ICR THAT 8085 HAS.
;534     ;       DEBUG.SAVE.WR0=90       ; TEMPORARY SAVE WR0.
;535     ;       DEBUG.SAVE.WR1=91       ; TEMPORARY SAVE WR1.
;536     ;       DEBUG.SAVE.ALU.CC=92    ; TEMPORARY SAVE CONDITION CODES.
;537     ;       MEMORY.SIZE=93          ; MEMORY SIZE OF THE MACHINE.
;538     ;       CRD.LOG=94              ; INFORMATION ABOUT LAST SOFT MEMORY ERROR.
;539     ;       PACKET.A=95             ; TOP OF PACKET.
;540     ;       PACKET.B=96             ; BOTTOM OF PACKET.
;541     ;       DEBUG.SAVE.ADDRESS=97   ; ADDRESS FOR EXAMINE/DEPOSIT.
;542     ;
```

```
;543     ;       SAVED.MDT=99            ; SAVED ADDRESS OF MDT FOR MMIE OPTIMIZATION.
;544     ;       P0BR=9A                 ; P0 BASE REGISTER
;545     ;       P0LR=9B                 ; P0 LENGTH REGISTER
;546     ;       P1BR=9C                 ; P1 BASE REGISTER
;547     ;       P1LR=9D                 ; P1 LENGTH REGISTER
;548     ;       SBR=9E                  ; SYSTEM BASE REGISTER
;549     ;       SLR=9F                  ; SYSTEM LENGTH REGISTER
;550     ;
;551     ;       PORT0=0A0       ; RESERVED FOR PORT.
;552     ;       PORT1=0A1       ; RESERVED FOR PORT.
;553     ;       PORT2=0A2       ; RESERVED FOR PORT.
;554     ;       PORT3=0A3       ; RESERVED FOR PORT.
;555     ;       PORT4=0A4       ; RESERVED FOR PORT.
;556     ;       PORT5=0A5       ; RESERVED FOR PORT.
;557     ;       PORT6=0A6       ; RESERVED FOR PORT.
;558     ;       PORT7=0A7       ; RESERVED FOR PORT.
;559     ;       PORT8=0A8       ; RESERVED FOR PORT.
;560     ;       PORT9=0A9       ; RESERVED FOR PORT.
;561     ;       PORT10=0AA      ; RESERVED FOR PORT.
;562     ;       PORT11=0AB      ; RESERVED FOR PORT.
;563     ;       PORT12=0AC      ; RESERVED FOR PORT.
;564     ;       PORT13=0AD      ; RESERVED FOR PORT.
;565     ;       PORT14=0AE      ; RESERVED FOR PORT.
;566     ;       PORT15=0AF      ; RESERVED FOR PORT.
;567     ;       PORT16=0B0      ; RESERVED FOR PORT.
;568     ;       PORT17=0B1      ; RESERVED FOR PORT.
;569     ;       PORT18=0B2      ; RESERVED FOR PORT.
;570     ;       PORT19=0B3      ; RESERVED FOR PORT.
;571     ;       PORT20=0B4      ; RESERVED FOR PORT.
;572     ;       PORT21=0B5      ; RESERVED FOR PORT.
;573     ;       PORT22=0B6      ; RESERVED FOR PORT.
;574     ;       PORT23=0B7      ; RESERVED FOR PORT.
;575     ;       PORT24=0B8      ; RESERVED FOR PORT.
;576     ;       PORT25=0B9      ; RESERVED FOR PORT.
;577     ;       PORT26=0BA      ; RESERVED FOR PORT.
;578     ;       PORT27=0BB      ; RESERVED FOR PORT.
;579     ;       PORT28=0BC      ; RESERVED FOR PORT.
;580     ;       PORT29=0BD      ; RESERVED FOR PORT.
;581     ;       PORT30=0BE      ; RESERVED FOR PORT.
;582     ;       PORT31=0BF      ; RESERVED FOR PORT.
;583     ;
;584     ;       XGPR0=0C0               ; BACKUP COPY OF GPR 0
;585     ;       XGPR1=0C1               ; BACKUP COPY OF GPR 1
;586     ;       XGPR2=0C2               ; BACKUP COPY OF GPR 2
;587     ;       XGPR3=0C3               ; BACKUP COPY OF GPR 3
;588     ;       XGPR4=0C4               ; BACKUP COPY OF GPR 4
;589     ;       XGPR5=0C5               ; BACKUP COPY OF GPR 5
;590     ;       XGPR6=0C6               ; BACKUP COPY OF GPR 6
;591     ;       XGPR7=0C7               ; BACKUP COPY OF GPR 7
;592     ;       XGPR8=0C8               ; BACKUP COPY OF GPR 8
;593     ;       XGPR9=0C9               ; BACKUP COPY OF GPR 9
;594     ;       XGPR10=0CA              ; BACKUP COPY OF GPR 10
;595     ;       XGPR11=0CB              ; BACKUP COPY OF GPR 11
;596     ;       XGPR12=0CC              ; BACKUP COPY OF GPR 12
;597     ;       XGPR13=0CD              ; BACKUP COPY OF GPR 13
```

```
;598     ;       XSP=0CE                 ; BACKUP COPY OF GPR 14
;599     ;
;600     ;       #14=0D0                 ; CONSTANT
;601     ;       #18=0D1                 ; CONSTANT
;602     ;       #1C=0D2                 ; CONSTANT
;603     ;       #24=0D3                 ; CONSTANT
;604     ;       #28=0D4                 ; CONSTANT
;605     ;       #2C=0D5                 ; CONSTANT
;606     ;       #2D=0D6                 ; CONSTANT
;607     ;       #30=0D7                 ; CONSTANT
;608     ;       #F8=0D8                 ; CONSTANT
;609     ;       #FC=0D9                 ; CONSTANT
;610     ;       #2D20=0DA                       ; CONSTANT
;611     ;       #140000=0DB                     ; CONSTANT
;612     ;       #150000=0DC                     ; CONSTANT
;613     ;       #160000=0DD                     ; CONSTANT
;614     ;       #170000=0DE                     ; CONSTANT
;615     ;       #180000=0DF                     ; CONSTANT
;616     ;       #1E0000=0E0                     ; CONSTANT
;617     ;       #40A10=0E1                      ; CONSTANT
;618     ;       #C0000E0=0E2                    ; CONSTANT
;619     ;       #0FFF0000=0E3                   ; CONSTANT
;620     ;       #B020FF00=0E4                   ; CONSTANT
;621     ;       #FFFFFF10=0E5                   ; CONSTANT
;622     ;       DEBUG.SAVE.T1=0E6       ; TEMPORARY SAVE T1.
;623     ;       DEBUG.SAVE.OS=0E7       ; TEMPORARY SAVE OS.
;624     ;       DEBUG.SAVE.SIZE=0E8     ; TEMPORARY SAVE SIZE VALUE.
;625     ;       SAVED.WR0=0E9           ; MM SAVED WR0
;626     ;       SAVED.WR1=0EA           ; MM SAVED WR1
;627     ;       SAVED.2ND.VAR=0EB       ; MM SAVED VAR 2
;628     ;       SAVED.ALU.CC=0EC                ; MM SAVED ALU CONDITION CODES
;629     ;       SAVED.SIZE=0ED          ; MM SAVED LS[SIZE]
;630     ;       SAVED.OS=0EE            ; MM SAVED LS[OS]
;631     ;       SAVED.REQUEST=0EF       ; MM SAVED REQUEST DATA
;632     ;       SAVED.CRD.LOG=0F0       ; MM SAVED LOCATION
;633     ;       OS.BAK=0F1              ; USED BY WARM FLOATING POINT.
;634     ;
;635     ;       THIS LOCATION IS FOR USE BY THE MFPR AND MTPR TO THE CONSOLE AND
;636     ;       TU58 CSR'S.  ALL OF THE INTERRUPT ENABLE BITS ARE HEREIN.
;637     ;
;638     ;       CONSOLE/PROGRAM I/O MODE FLAG - BIT<31> 0 - CONSOLE 1 - PROGRAM I/O MODE
;639
;640     ;       CRD RETRY IN PROGRESS - BIT<5>  0 - NOTHING DOING 1 - IN PROGRESS
;641     ;       MACHINE CHECK IN PROGRESS - BIT<4>
;642     ;       TU58 TRANSMIT CSR IE - BIT<3>
;643     ;       TU58 RECEIVER CSR IE - BIT<2>
;644     ;       CONSOLE TRANSMIT CSR IE - BIT<1>
;645     ;       CONSOLE TRANSMIT CSR IE - BIT<0>
;646     ;
;647     ;       CONSOLE.CSR.IES=0F2     ; CONSOLE CSR IE'S AND CONSOLE/PROGRAM I/O MODE FLAG.
;648     ;       IE.TEMP1=0F3            ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION ONE
;649     ;       IE.TEMP2=0F4            ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION TWO
;650     ;       IE.TEMP3=0F5            ; INTERRUPTS AND EXCEPTIONS TEMPORARY LOCATION THREE
;651     ;
;652     ;       XGPR.OS=0F8             ; HARDWARE INDEX TO XGPRS
```

```
;653     ;       PORT(3-0)=0F9           ; 16 LOCATION HARDWARE INDEX TO PORT REGISTERS.
;654     ;       PORT(4-0)=0FB           ; 32 LOCATION HARDWARE INDEX TO PORT REGISTERS.
;655     ;       CCR=0FC                 ; CONSOLE READ REGISTER
;656     ;       TIMER=0FD               ; INTERVAL TIMER VALUE.
;657     ;       ALU.CC=0FE              ; ALU CONDITION CODES
;658     ;
;659     ;       THIS LOCATION IS A WRITE ONLY REGISTER.  THE FOLLOWING BITS IN THE
;660     ;       PSL ARE AFFECTED:
;661     ;
;662     ;       COMPATIBILITY MODE - BIT<31>
;663     ;       CURRENT MODE - BITS<25:24>
;664     ;       INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
;665     ;       TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
;666     ;       NEGATIVE CONDITION CODE - BIT<3>
;667     ;       ZERO CONDITION CODE - BIT<2>
;668     ;       OVERFLOW CONDITION CODE - BIT<1>
;669     ;       CARRY CONDITION CODE - BIT<0>
;670     ;
;671     ;       PSL.HW=0FF              ; HARDWARE PSL BITS
;672
;673
;674     ; DATA PATH CONTROL
;675     ;
;676     ; THIS FIELD IS BUILT FROM AN ADDRESS IN THE CCODE MEMORY.  THE CCODE MEMORY
;677     ; CONTAINS DATA PATH MICRO INSTRUCTIONS TO PRODUCE THE DESIRED OPERATION
;678     ; FOR THE UCODE MACROS.
;679
;680     ; DATA PATH CONTROL FIELD FOR THE 'BASIC' MICRO INSTRUCTION
;681     ;
;682     DP/=<21:16>,.VALIDITY=<DP.VAL>,.DEFAULT=<.SHIFT[.AND[<SDP/Q.CMP.LS>,0FF],-2]>
;683     ;
;684     ; THIS FIELD IS ONLY USED WITH THE INSTRUCTIONS WHICH CAN HAVE
;685     ; XCHG ATTACHED.
;686     ;
;687     DP.SMALL/=<20:16>,.VALIDITY=<DP.VAL>
;688     ;
;689     ; EXCHANGE ORIGINAL WR TO LS
;690     ;
;691     XCHG.BIT/=<21>,.DEFAULT=<0>
;692
;693             YES=1
;694             NO=0
;695
;696     ; DATA PATH CONTROL FIELD FOR  THE 'EXTENDED' MICRO INSTRUCTION
;697     ;
;698     XDP/=<19:14>,.VALIDITY=<A.VAL>
;699
;700     ; DATA PATH CONTROL FOR THE 'MOVE' MICRO INSTRUCTION
;701     ;
;702     MDP/=<19:17>,.VALIDITY=<XD.VAL>
;703
```

```
;704     .PAGE
;705     ; CONDITION CODE FIELD / DATA PATH CONTEXT
;706     ;
;707     CC/=<6:5>,.VALIDITY=<CC.VAL>,.DEFAULT=<CC/DT(LONG)&HOLD.ALU.CC>
;708
;709             DT(LONG)&HOLD.ALU.CC=0          ; USE LONG CONTEXT(32 BITS) AND HOLD ALU CONDITION CODES
;710             DT(LONG)&SET.ALU.CC=1           ; USE LONG CONTEXT(32 BITS) AND  SET ALU CONDITION CODES
;711             DT(SIZE)&SET.ALU.CC=2           ; USE CONTEXT IN THE SIZE REGISTER AND SET ALU CONDITION CODES
;712             DT(SIZE)&MAP.ALU.CC.TO.PSL=3    ; TRANSFER ALU CONDITION CODES TO PSL CONDITION CODES HARDWARE DEPENDENT
;713
;714
;715     ; MEMORY DATA TYPE FIELD
;716     ;
;717     MDT/=<6:5>,.VALIDITY=<DT.VAL>
;718
;719             BYTE=0                  ; SIZE = BYTE
;720             WORD=1                  ; SIZE = WORD
;721             SIZE=2                  ; SIZE = CONTENTS OF SIZE REGISTER
;722             LONG=3                  ; SIZE = LONG
;723
;724
;725     ; SHIFT INPUT FIELD
;726     ;
;727     ; THIS CONTROL FIELD DETERMINES THE SHIFT INPUTS FOR THE FOLLOWING OPERATIONS.
;728     ; THE DIRECTION OF THE SHIFT IS DETERMINED BY THE 2901 DESTINATION CODE.
;729     ;
;730     SI/=<13:11>,.VALIDITY=<A.VAL>
;731
;732             ROT.WR=0                ; ROTATE WR
;733             ROT.WR.Q=1              ; ROTATE WR AND Q
;734             ASH.WR=2                ; ARITHMETIC SHIFT OF WR
;735             ASH.WR.Q=3              ; ARITHMETIC SHIFT OF WR AND Q
;736             MUL.SHF=4               ; SIGNED MULTIPLY SHIFT OPERATION
;737             UMUL.SHF=5              ; UNSIGNED MULTIPLY SHIFT OPERATION
;738             SHF.WR.Q=6              ; LOGICAL SHIFT WR AND Q
;739
;740
;741     ; SKIP MICRO CONTROL FIELD
;742     ;
;743     ; THIS FIELD IS VALID FOR ALL INSTRUCTIONS EXCEPT
;744     ;
;745     ;               JUMP   MICRO INSTRUCTION
;746     ;               DECODE MICRO INSTRUCTION
;747     ;
;748     SCTL/=<4:0>,.VALIDITY=<.AND[<SCTL.VAL>,<SKIP.PAGE.VAL>]>,.DEFAULT=<SCTL/NO.SKIP>
;749
;750             N.CLR=0                 ; ALU N-BIT EQUAL ZERO
;751             NEQ=1                   ; ALU Z-BIT EQUAL ZERO
;752             BIT.SET=1               ; ALU Z-BIT EQUAL ZERO
;753             V.CLR=2                 ; ALU V-BIT EQUAL ZERO
;754             C.SET=3                 ; ALU C-BIT EQUAL ONE
;755             GEQU=3                  ; ALU C-BIT EQUAL ONE
;756             NOT.PSL.C=4             ; PSL C EQUAL ZERO
;757             BR.FALSE=5              ; BRANCH INSTRUCTION FALSE
;758             R.DST=6                 ; REGISTER MODE FLAG
```

```
;759             NO.INTERRUPT=7          ; NO INTERRUPT PENDING
;760             N.SET=8                 ; ALU N-BIT EQUAL ONE
;761             EQL=9                   ; ALU Z-BIT EQUAL ONE
;762             BITS.CLR=9              ; ALU Z-BIT EQUAL ONE
;763             V.SET=0A                ; ALU V-BIT EQUAL ONE
;764             C.CLR=0B                ; ALU C-BIT EQUAL ZERO
;765             LSSU=0B                 ; ALU C-BIT EQUAL ZERO
;766             STATE.ZERO.SET=0C       ; STATE REGISTER BIT ZERO TRUE
;767             STATE.ONE.SET=0D        ; STATE REGISTER BIT ONE TRUE
;768             STATE.ZERO.CLR=0E       ; STATE REGISTER BIT ZERO FALSE
;769             STATE.ONE.CLR=0F        ; STATE REGISTER BIT ONE FALSE
;770             RET.NOERR=10            ; RETURN+1 IF NO MEMORY ERROR
;771             JMP.Z.CLR=11            ; JUMP TO (STACK) IF ALU Z = 0
;772             POP.USTACK=12           ; DISCARD TOP STACK ENTRY
;773             JMP.C.SET=13            ; JUMP TO (STACK) IF ALU C=1
;774             RETURN=14               ; RETURN TO (USTACK)
;775             NO.SKIP=15              ; EXECUTE NEXT INSTRUCTION
;776             RETURN+1=16             ; RETURN TO (USTACK)+1
;777             LOOP.IF.NO.I.AND.SYNC=17; JUMP TO (STACK) IF NO INTERRUPT AND NO ACCELERATOR SYNC.
;778             CONSOLE.ATTN=18         ; CONSOLE ATTENTION
;779             CONSOLE.ACK=19          ; CONSOLE ACKNOWLEDGE
;780             NO.FAST.INTERRUPT=1A    ; NO FAST INTERRUPTS PENDING
;781             BACKUP.MASK.VALID=1B    ; REGISTER BACKUP MASK VALID.
;782             LSS=1C                  ; SIGNED LESS THAN.
;783             MEM.REF.OK=1D           ; NO MEMORY ERROR TRUE
;784             SKIP=1E                 ; EXECUTE ADDRESS PLUS ONE
;785             SYNC=1F                 ; ACCELERATOR SYNCHRONIZATION
;786
;787
;788     ; JUMP MICRO CONTROL FIELD
;789     ;
;790     ; THIS FIELD IS VALID FOR THE FOLLOWING INSTRUCTIONS
;791     ;
;792     ;               JUMP   MICRO INSTRUCTION
;793     ;               DECODE MICRO INSTRUCTION
;794     ;
;795     JCTL/=<3:0>,.VALIDITY=<.AND[<JCTL.VAL>,<JUMP.PAGE.VAL>]>,.DEFAULT=<JCTL/JUMP.VALID>
;796
;797             N.CLR=0                 ; ALU N-BIT EQUAL ZERO
;798             NEQ=1                   ; ALU Z-BIT EQUAL ZERO
;799             BIT.SET=1               ; ALU Z-BIT EQUAL ZERO
;800             V.CLR=2                 ; ALU V-BIT EQUAL ZERO
;801             C.SET=3                 ; ALU C-BIT EQUAL ONE
;802             GEQU=3                  ; ALU C-BIT EQUAL ONE
;803             JUMP=4                  ; JUMP UNCONDITIONALLY
;804             BR.FALSE=5              ; BRANCH INSTRUCTION FALSE
;805             R.DST=6                 ; REGISTER MODE FLAG
;806             NO.INTERRUPT=7          ; NO INTERRUPT PENDING
;807             N.SET=8                 ; ALU N-BIT EQUAL ONE
;808             EQL=9                   ; ALU Z-BIT EQUAL ONE
;809             BITS.CLR=9              ; ALU Z-BIT EQUAL ONE
;810             V.SET=0A                ; ALU V-BIT EQUAL ONE
;811             C.CLR=0B                ; ALU C-BIT EQUAL ZERO
;312             LSSU=0B                 ; ALU C-BIT EQUAL ZERO
;813             JSR=0C                  ; UNCONDITIONAL JUMP AND PUSH USTACK
```

```
;814             JSR.VALID=0D            ; JUMP AND PUSH USTACK IF IB VALID=1
;815             NO.JUMP.TST=0E          ; DO NOT JUMP AND SKIP IF IB VALID=0
;816             JUMP.VALID=0F           ; JUMP IF IB VALID=1
;817
;818
;819     ; JUMP ADDRESS FIELD
;820     ;
;821     JUMP.ADRS/=<17:4>,.ADDRESS,.VALIDITY=<JUMP.VAL>
;822
;823
;824     ; OS ENABLE
;825     ;
;826     OS/=<18>,.VALIDITY=<JUMP.VAL>
;827
;828             NOP=0
;829             WITH.OS=1               ; CONCATENATE OS REGISTER TO JUMP ADDRESS
;830
;831
```

```
;832     .PAGE
;833     ; BACKUP PC
;834     ;
;835     BPC/=<18>,.VALIDITY=<DECODE.VAL>,.DEFAULT=<.CASE[<.EQL[<OPC2/>,<OPC2/DECODE>]>]OF[0,1]>
;836
;837             NOP=1
;838             SAVE.PC=0               ; WRITE ALU DATA INTO WR[B]   (PC+1)
;839
;840
;841     ; DECODE ADDRESS FIELD
;842     ;
;843     DEC.ADRS/=<17:12>,.VALIDITY=<DECODE.VAL>
;844
;845
;846     ; IR FUNCTION
;847     ;
;848     IFUNC/=<11:9>,.VALIDITY=<DECODE.VAL>
;849
;850             CM.EXEC=0               ; COMPATIBILITY MODE EXECUTION DISPATCH WHEN UPPER BYTE = ZERO
;851             SPEC=0                  ; SPECIFIER DISPATCH
;852             CM.IRD=1                ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
;853             FLOAT=1                 ; SPECIFIER FLOAT TYPE DISPATCH
;854             VAX.EXEC=2              ; VAX EXECUTION DISPATCH
;855             ASRC=2                  ; ADDRESS SPECIFIER DISPATCH
;856             VAX.IRD=3               ; VAX OPCODE DISPATCH
;857             VSRC=4                  ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
;858             ESRC=5                  ; SPECIFIER DISPATCH IN INDEX MODE
;859             CM.DST=6                ; COMPATIBILITY MODE DESTINATION DISPATCH
;860             CM.SINGLE=7             ; COMPATIBILITY MODE SOP DISPATCH
;861
;862
;863     ; INSTRUCTION BUFFER REQUEST
;864     ;
;865     IB.REQ/=<8>,.VALIDITY=<DECODE.VAL>,.DEFAULT=0
;866
;867             NOP=0
;868             IB.REQ=1                ; ENABLE HARDWARE DEPENDENT MEMORY REQUEST
;869     ;
;870     ; COMPATIBILITY MODE OPCODE SELECT
;871     ;
;872     CM.HI/=<7>,.VALIDITY=<DECODE.VAL>,.DEFAULT=0
;873
;874             LOW.BYTE=0              ; DECODE IR BITS<7:0>
;875             HI.BYTE=1               ; DECODE IR BITS <15:6>
;876
;877
;878     ; LOAD OS REGISTER
;879     ;
;880     LD.OS/=<6>,.VALIDITY=<DECODE.VAL>,.DEFAULT=0
;881
;882             NOP=0
;883             LOAD.OS=1               ; LOAD OS REGISTER FROM I-BUFFER
;884
;885
;886     ; REGISTER DESTINATION FLAG ENABLE
```

```
;887     ;
;888     R.DST/=<5>,.VALIDITY=<DECODE.VAL>,.DEFAULT=0
;889
;890             NOP=0
;891             ENAB=1
;892     ;
;893     ; OPCODE SPECIFIER BIT
;894     ;
;895     OPC.SPEC/=<4>,.VALIDITY=<DECODE.VAL>
;896
;897             CM.EXEC=1               ; COMPATIBILITY MODE EXECUTION DISPATCH
;898             SPEC=0                  ; SPECIFIER DISPATCH
;899             CM.IRD=1                ; COMPATIBILITY MODE INSTRUCTION DECODE DISPATCH
;900             FLOAT=0                 ; SPECIFIER FLOAT TYPE DISPATCH
;901             VAX.EXEC=1              ; VAX EXECUTION DISPATCH
;902             ASRC=0                  ; ADDRESS SPECIFIER DISPATCH
;903             VAX.IRD=1               ; VAX OPCODE DISPATCH
;904             VSRC=0                  ; ADDRESS SPECIFIER DISPATCH FOR BIT FIELD INSTRUCTIONS
;905             ESRC=0                  ; SPECIFIER DISPATCH IN INDEX MODE
;906             CM.DST=0                ; COMPATIBILITY MODE DESTINATION DISPATCH
;907             CM.SINGLE=0             ; COMPATIBILITY MODE SOP DISPATCH
```

```
;908     .PAGE
;909     ; MISCELLANEOUS FUNCTIONS
;910     ;
;911     ;
;912     ; THIS FIELD IS A FLAG FOR STEERING THE MICROWORD AS A MISCELLANEOUS
;913     ; INSTRUCTION OR AS A PORT INSTRUCTION.
;914     ;
;915     MISC.PORT/=<18>
;916
;917             MISC=0
;918             PORT=1
;919
;920     MISC.0/=<17>,.VALIDITY=<MISC.VAL>
;921
;922             NOP=0                   ; NO OPERATION IN ALU OR CONDITION CODES.
;923
;924     MISC.1/=<15:12>,.VALIDITY=<MISC.VAL>
;925
;926             NOP=0F                  ; NO OPERATION
;927             SET.CP.ATTN=0E          ; SET CPU ATTENTION FOR CONSOLE
;928             SET.CP.ACK=0D           ; SET CPU ACKNOWLEDGE FOR CONSOLE
;929             CLR.CP.ATTN.AND.ACK=0C  ; CLEAR CPU ATTENTION AND CPU ACKNOWLEDGE
;930             SET.HIGH.PAGE=0B        ; SET BIT FOR HIGH HALF OF WCS.
;931             CLR.HIGH.PAGE=0A        ; CLEAR BIT FOR LOW HALF OF WCS.
;932             SET.PORT.I.O=9          ; SET PORT I/O MODE
;933             SET.ACC.I.O=8           ; SET ACCELERATOR I/O MODE
;934             SET.BACKUP.MASK.VALID=7 ; SET REGISTER BACKUP MASK FLAG
;935             SET.S1=6                ; SET STATE ONE BIT
;936             SET.S0=5                ; SET STATE ZERO BIT
;937             CLR.S1=4                ; CLEAR STATE ONE BIT
;938             CLR.S0=3                ; CLEAR STATE ZERO BIT
;939             SET.S1&CLR.S0=2         ; SET STATE ONE BIT AND CLEAR STATE ZERO BIT
;940             CLR.S1&SET.S0=1         ; CLEAR STATE ONE AND SET STATE ZERO
;941             CLR=0                   ; CLEAR STATE ONE AND STATE ZERO BITS.
;942
;943
;944     MISC.2/=<11:9>,.VALIDITY=<MISC.VAL>
;945
;946             NOP=0                   ; NO OPERATION
;947             MASK.INTERRUPTS=1       ; MASK ALL INTERRUPTS (FOR DECODE NEXT CYCLE).
;948             ASSERT.DA.AND.PASS.WR=2; ASSERT DATA AVAILABLE AND PASS WR[0] TO THE ACCELERATOR OR PORT.
;949             TRAP.ACC=3              ; TRAP ACCELERATOR
;950             READ.ACC.UPC=4          ; READ ACCELERATOR PROGRAM COUNTER
;951             TRANSFER.GRANT=5        ; RECOGNIZE PORT REQUEST.
;952             MASK.HALT.AND.T.BIT=6   ; MASK HALT AND T-BIT TRAPS (FOR DECODE TWO CYCLES LATER)
;953             WRITE.WR.TO.CONSOLE.REGISTER=7  ; SEND BITS <7:0> TO 8085.
;954
;955     MISC.WR/=<8:7>
;956
;957     MISC.WRL/=<8>
;958     MISC.WRR/=<7>
;959
;960     PORT.SELECT/=<17:15>
;961
;962             IDC=7
```

```
;963
;964     PORT.FUNC/=<14:13>
;965
;966             READ=0
;967             WRITE=1
;968             CONTROL=2
;969
;970     R.W.FUNC/=<12:10>
;971
;972             CSR=0
;973             DAR=1
;974             DATA.BYTE=2
;975             DATA.WORD=3
;976             PATTERN=4
;977             POSITION=5
;978
;979     CNTL.FUNC/=<12:10>
;980
;981             CLEAR.FIFO.CNTR=0
;982             RESET.BR=1
;983             CLEAR.IDC=3
;984             SET.AUTO=4
;985             CLEAR.AUTO=5
;986             SEL.FIFO.A=6
;987             SEL.FIFO.B=7
;988
```

```
;989     .PAGE
;990     ; MEMORY REQUEST FIELD
;991     ;
;992     ; THIS FIELD IS USED TO INDICATE THE DESIRED OPERATION TO THE MEMORY CONTROL.
;993     ; NOTE THAT THIS IS A DUMMY FIELD. IT IS ACTUALLY MADE UP OF TWO FIELDS.
;994     ; M.FUNC1 AND M.FUNC2
;995     ;
;996     ; THE FOLLOWING SPECIAL CONSIDERATIONS MUST BE MADE:
;997     ;
;998     ; ROTATE.BYTE.RIGHT - PERFORMS 9-BIT ROTATE AND THEREFORE THE
;999     ; ANSWER MUST BE CLEANED UP WITH A ROL WR[].
;1000    ;
;1001    ; WORD.SWAP - PERFORMS A 15-BIT ROTATE AND THERE FORE
;1002    ; THE ANSWER MUST BE CLEANED UP WITH A ROL WR[].
;1003    ;
;1004    ; OCTA.WRITE.P - MUST BE LONGWORD ALLIGNED.
;1005    ;
;1006    ; OCTA.WR.V.WCHK - MUST BE LONGWORD ALLIGNED.  THIS DATA CAN
;1007    ; CROSS PAGE BOUNDARIES
;1008    ;
;1009    MEM.FUNC/=<7>,.VALIDITY=0
;1010
;1011    ;       PREFETCH=0              ; USED TO TAKE VAR AND ADD ONE BEFORE REFERENCE.
;1012            WRITE.TB=01             ; WRITE TRANSLATION BUFFER
;1013            TEST.V.RCHK=2           ; TEST WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
;1014            READ.P=3                ; READ WITH PHYSICAL ADDRESS
;1015            WRITE.P=4               ; WRITE WITH PHYSICAL ADDRESS
;1016            TEST.V.WCHK=5           ; WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
;1017            ROTATE.BYTE.RIGHT=6     ; ROTATE ADDRESS DATA 9 BITS RIGHT AND RETURN
;1018            WORD.SWAP=7             ; ROTATE ADDRESS DATA 15 BITS LEFT AND RETURN
;1019            READ.V.NOCHK=8          ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
;1020            READ.V.WCHK=9           ; READ WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
;1021            READ.V.RCHK=0A          ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK
;1022            READ.MAINT.VAR.INC=0B   ; TEST VAR INCREMENTING.
;1023            WRITE.V.NOCHK=0C        ; WRITE WITH VIRTUAL ADDRESS AND NO ACCESS CHECK
;1024            WRITE.V.WCHK=0D         ; WRITE WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK
;1025            IB.FILL=0E              ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
;1026            READ.V.RCHK.IFILL=0E    ; READ WITH VIRTUAL ADDRESS AND READ ACCESS CHECK AND FILL IB
;1027            WRITE.UBS.MAP=0F        ; WRITE TO THE UNIBUS MAP.
;1028
;1029            OCTA.WRITE.P=10         ; WRITE OCTA DATA WITH PHYSICAL ADDRESS.
;1030            WRITE.TB.STEP=11        ; WRITE TO TWO TB ENTRIES(WITH CORRECT ADDRESS)
;1031            READ.UBS.MAP=12         ; READ UNIBUS MAP
;1032            READ.TB=13              ; READ TRANSLATION BUFFER
;1033            READ.MAINT.ADRS=14      ; RPUTS VA ON UNIBUS LINES AND READS THIS DATA
;1034            MAINT.ECC.DATA=15       ; TEST ALL ECC FUNCTIONS.
;1035            READ.CSR=16             ; READ CONTROL REGISTER
;1036            READ.CNTRL.REG=16       ; READ CONTROL REGISTER
;1037            WRITE.CNTRL.REG=17      ; WRITE CONTROL REGISTER
;1038            WRITE.CSR=17            ; WRITE CONTROL REGISTER
;1039            OCTA.READ.P=18          ; READ OCTA DATA WITH PHYSICAL ADDRESS.
;1040            READ.V.WCHK.LOCKED=19   ; READ VIRTUAL ADDRESS AND WRITE ACCESS CHECK AND LOCKED
;1041            OCTA.READ.V.RCHK=1A     ; READ OCTA WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
;1042            READ.MAINT.BR.CHECK=1B ; TESTS BRANCHING FUNCTION.
;1043            ISSUE.BG=1C             ; ISSUES BUS GRANT (4+ADDRESS<1:0>)
```

```
;1044            OCTA.WR.V.WCHK=1D       ; WRITE OCTA WITH VIRTUAL ADDRESS AND WRITE ACCESS CHECK.
;1045            QUAD.READ.V.RCHK=1E     ; READ QUAD WITH VIRTUAL ADDRESS AND READ ACCESS CHECK.
;1046            READ.MAINT.UBS=1F       ; PUTS VA ON UNIBUS LINES AND READS THIS AS DATA.
;1047
;1048
;1049    M.FUNC2/=<18:16>,.VALIDITY=<DT.VAL>
;1050
;1051    M.FUNC1/=<8:7>,.VALIDITY=<DT.VAL>
;1052
;1053    PAR/=<23>,.DEFAULT=<.PARITY[<OPC2/>,<OS/>,<JUMP.ADRS/>,<JCTL/>]>
;1054    .UCODE
```

```
;1055    .PAGE   'MACRO DEFINITIONS                                          VER 00.02''
;1056
;1057    ;       THE FORMAT FOR THIS GROUP IS TWO ADDRESS WITH A MODIFY DESTINATION.
;1058    ;       THE FIRST OPERAND IS COMBINED WITH THE SECOND OPERAND AND THE
;1059    ;       RESULT WRITTEN TO THE SECOND OPERAND.  CMP AND BIT INSTRUCTIONS ARE
;1060    ;       READ ONLY.
;1061    ;
;1062    ;       DURING ALL ARITHMETIC AND LOGICAL INSTRUCTIONS THE ALU CC'S
;1063    ;       CAN BE LOADED BY ADDING THE
;1064    ;
;1065    ;               DT(SIZE)&SET.ALU.CC
;1066    ;
;1067    ;       MACRO TO THE MICROWORD.  THIS SPECIFIES THAT ALU CC'S ARE LOADED
;1068    ;       WITH THE 2901 CONDITIONS.  NO EXTERNAL MUNGING IS DONE.  IF THE
;1069    ;       DATA OPERATION IS WITH BYTES THEN THE CC'S ARE SET
;1070    ;       ACCORDING TO BITS 7-0. WORD USES BITS 15-0 AND LONGWORD USES BITS
;1071    ;       31-0.  THE FOLLOWING IS A DESCRIPTION OF THE ALU CC'S VALUE
;1072    ;       FOR EACH FUNCTION:
;1073    ;
;1074    ;       ADD -- BINARY ADDITION OF THE TWO OPERANDS
;1075    ;
;1076    ;       N - SIGN BIT
;1077    ;       Z - SET IF ANSWER IS ZERO.
;1078    ;       V - ARITHMETIC OVERFLOW
;1079    ;       C - CARRY
;1080    ;
;1081    ;       SUB -- BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF THE SECOND OPERAND.
;1082    ;
;1083    ;       N - SIGN BIT
;1084    ;       Z - SET IF ANSWER IS ZERO.
;1085    ;       V - ARITHMETIC OVERFLOW
;1086    ;       C - COMPLEMENT OF THE BORROW.
;1087    ;
;1088    ;       BIS -- LOGICAL OR OF THE TWO OPERANDS.
;1089    ;
;1090    ;       N - SIGN BIT
;1091    ;       Z - SET IF ANSWER IS ZERO
;1092    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1093    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1094    ;
;1095    ;       BIC -- ONE'S COMPLEMENT OF FIRST OPERAND ANDED WITH SECOND OPERAND.
;1096    ;
;1097    ;       N - SIGN BIT
;1098    ;       Z - SET IF ANSWER IS ZERO
;1099    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1100    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1101    ;
;1102    ;       AND -- LOGICAL AND OF THE TWO OPERANDS.
;1103    ;
;1104    ;       N - SIGN BIT
;1105    ;       Z - SET IF ANSWER IS ZERO
;1106    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1107    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1108    ;
;1109    ;       XOR -- LOGICAL EXCLUSIVE OR OF THE TWO OPERANDS.
```

```
;1110    ;
;1111    ;       N - SIGN BIT
;1112    ;       Z - SET IF ANSWER IS ZERO
;1113    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1114    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1115    ;
;1116    ;       CMP -- PERFORM BINARY ADDITION OF FIRST OPERAND AND TWO'S COMPLEMENT OF SECOND OPERAND BUT DO NOT STORE.
;1117    ;
;1118    ;       N - SIGN BIT
;1119    ;       Z - SET IF ANSWER IS ZERO.
;1120    ;       V - ARITHMETIC OVERFLOW
;1121    ;       C - CARRY
;1122    ;
;1123    ;       BIT -- PERFORM LOGICAL AND OF THE TWO OPERANDS BUT DO NOT STORE.
;1124    ;
;1125    ;       N - SIGN BIT
;1126    ;       Z - SET IF ANSWER IS ZERO
;1127    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1128    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1129    ;
;1130    ;       SWAP -- SWITCH THE TWO VALUES.
;1131    ;
;1132    ;       N - SIGN BIT OF VALUE TO WR.
;1133    ;       Z - SET IF ANSWER IS ZERO OF VALUE TO WR.
;1134    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1135    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
```

```
;1136    .PAGE
;1137    ;
;1138    ;       MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE LS[D]
;1139    ;
;1140    ADD  LS[] TO   WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.PLUS.WR>,07F]>,-2]>''
;1141    SUB  LS[] FROM WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.FROM.WR>,07F]>,-2]>''
;1142    BIS  LS[] TO   WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.OR.WR>,0FF]>,-2]>''
;1143    BIC  LS[] TO   WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.MASK.WR>,0FF]>,-2]>''
;1144    AND  LS[] TO   WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.AND.WR>,07F]>,-2]>''
;1145    XOR  LS[] TO   WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP.SMALL/<.SHIFT[<.AND[<SDP/LS.XOR.WR>,07F]>,-2]>''
;1146
;1147    CMP  LS[] WITH WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/LS.CMP.WR>,0FF]>,-2]>''
;1148    BIT  LS[] WITH WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,0FF]>,-2]>''
;1149    SWAP LS[] WITH WR[]              ''OPC/BASIC,D.ADRS/a1,B.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/SWAP.LS.WR>,0FF]>,-2]>''
;1150    ;
;1151    ;       THE FOLLOWING MACROS IS TO BE USED WITH ADD,SUB,
;1152    ;       AND,XOR LS[] TO WR[]
;1153    ;
;1154    ;       IT WILL TAKE THE ORIGINAL CONTENTS OF WR[]
;1155    ;       AND PUT IT IN LS[].
;1156    ;
;1157    XCHG                             ''XCHG.BIT/YES''
;1158    ;
;1159    ;       MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE Q-REGISTER
;1160    ;
;1161    ADD    Q TO   LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS>,0FF]>,-2]>''
;1162    SUB    Q FROM LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS>,0FF]>,-2]>''
;1163    BIS    Q TO   LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.OR.LS>,0FF]>,-2]>''
;1164    AND    Q TO   LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.AND.LS>,0FF]>,-2]>''
;1165    XOR    Q TO   LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.XOR.LS>,0FF]>,-2]>''
;1166
;1167    CMP    Q WITH LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,0FF]>,-2]>''
;1168    BIT    Q WITH LS[]               ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.BIT.Q>,0FF]>,-2]>''
;1169    ;
;1170    ;       MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE LS[D]
;1171    ;
;1172    ADD  LS[] TO   Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.Q>,0FF]>,-2]>''
;1173    SUB  LS[] FROM Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q>,0FF]>,-2]>''
;1174    BIS  LS[] TO   Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.OR.Q>,0FF]>,-2]>''
;1175    BIC  LS[] TO   Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.MASK.Q>,0FF]>,-2]>''
;1176    AND  LS[] TO   Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.AND.Q>,0FF]>,-2]>''
;1177    XOR  LS[] TO   Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.XOR.Q>,0FF]>,-2]>''
;1178
;1179    CMP  LS[] WITH Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.CMP.Q>,0FF]>,-2]>''
;1180    BIT  LS[] WITH Q                 ''OPC/BASIC,D.ADRS/a1,DP/<.SHIFT[<.AND[<SDP/LS.BIT.Q>,0FF]>,-2]>''
;1181    ;
;1182    ;       MACROS FOR THE FORM OF LS[D]<-- LS[D] OPERATE WR[B]
;1183    ;
;1184    ADD  WR[] TO   LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS>,0FF]>,-2]>''
;1185    SUB  WR[] FROM LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS>,0FF]>,-2]>''
;1186    BIS  WR[] TO   LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS>,0FF]>,-2]>''
;1187    AND  WR[] TO   LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS>,0FF]>,-2]>''
;1188    XOR  WR[] TO   LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS>,0FF]>,-2]>''
;1189
;1190    CMP  WR[] WITH LS[]              ''OPC/BASIC,B.ADRS/a1,D.ADRS/a2,DP/<.SHIFT[<.AND[<SDP/WR.CMP.LS>,0FF]>,-2]>''
```

```
;1191   BIT  WR[] WITH LS[]          "OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.BIT.LS>,0FF]>,-2]>"
;1192   SWAP WR[] WITH LS[]          "SWAP LS[@2] WITH WR[@1]"
;1193   ;
;1194   ;       MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE WR[A]
;1195   ;
;1196   ADD  WR[] TO   WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR>,03F]>"
;1197   SUB  WR[] FROM WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR>,03F]>"
;1198   BIS  WR[] TO   WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR>,03F]>"
;1199   BIC  WR[] TO   WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR>,03F]>"
;1200   AND  WR[] TO   WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR>,03F]>"
;1201   XOR  WR[] TO   WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>"
;1202
;1203   CMP  WR[] WITH WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.CMP.WR>,03F]>"
;1204   BIT  WR[] WITH WR[]          "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.BIT.WR>,03F]>"
;1205   ;
;1206   ;       MACROS FOR THE FORM OF WR[B]<-- WR[B] OPERATE Q-REGISTER
;1207   ;
;1208   ADD  Q TO   WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.PLUS.WR>,03F]>"
;1209   SUB  Q FROM WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR>,03F]>"
;1210   BIS  Q TO   WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.OR.WR>,03F]>"
;1211   AND  Q TO   WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.AND.WR>,03F]>"
;1212   XOR  Q TO   WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.XOR.WR>,03F]>"
;1213
;1214   CMP  Q WITH WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.CMP.WR>,03F]>"
;1215   BIT  Q WITH WR[]             "OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>"
;1216   ;
;1217   ;       MACROS FOR THE FORM OF Q-REGISTER<-- Q-REGISTER OPERATE WR[B]
;1218   ;
;1219   ADD  WR[] TO   Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.Q>,03F]>"
;1220   SUB  WR[] FROM Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.FROM.Q>,03F]>"
;1221   BIS  WR[] TO   Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.OR.Q>,03F]>"
;1222   BIC  WR[] TO   Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.MASK.Q>,03F]>"
;1223   AND  WR[] TO   Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.AND.Q>,03F]>"
;1224   XOR  WR[] TO   Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.Q>,03F]>"
;1225
;1226   CMP  WR[] WITH Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/WR.CMP.Q>,03F]>"
;1227   BIT  WR[] WITH Q              "OPC1/EXTENDED,A.ADRS/@1,XDP/<.AND[<SDP/Q.BIT.WR>,03F]>"
```

```
;1228    .PAGE
;1229    ;
;1230    ;       THE FORMAT OF THIS GROUP IS THREE ADDRESS TYPE. THE FIRST AND SECOND
;1231    ;       OPERANDS ARE READ AND THE RESULT IS WRITTEN INTO THE THIRD OPERAND.
;1232    ;
;1233    ;       MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE WR[A]
;1234    ;
;1235    ADD  WR[] PLUS WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR.Q>,03F]>''
;1236    SUB  WR[] FROM WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.Q>,03F]>''
;1237    BIS  WR[] WITH WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>''
;1238    BIC  WR[] WITH WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.MASK.WR.Q>,03F]>''
;1239    AND  WR[] WITH WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.AND.WR.Q>,03F]>''
;1240    XOR  WR[] WITH WR[] TO Q        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>''
;1241    ;
;1242    ;       MACROS FOR THE FORM OF Q-REGISTER<-- WR[B] OPERATE LS[D]
;1243    ;
;1244    ADD  WR[] PLUS LS[] TO Q        ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS.Q>,0FF]>,-2]>''
;1245    SUB  WR[] FROM LS[] TO Q        ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS.Q>,0FF]>,-2]>''
;1246    BIS  WR[] WITH LS[] TO Q        ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.OR.LS.Q>,0FF]>,-2]>''
;1247    AND  WR[] WITH LS[] TO Q        ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.AND.LS.Q>,0FF]>,-2]>''
;1248    XOR  WR[] WITH LS[] TO Q        ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.XOR.LS.Q>,0FF]>,-2]>''
;1249    ;
;1250    ;       MACROS FOR THE FORM OF Q-REGISTER <-- LS[D] OPERATE WR[B]
;1251    ;
;1252    ADD LS[] PLUS WR[] TO Q ''ADD WR[@2] PLUS LS[@1] TO Q''
;1253    SUB  LS[] FROM WR[] TO Q        ''OPC/BASIC,B.ADRS/@2,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.Q>,0FF]>,-2]>''
;1254    BIS LS[] WITH WR[] TO Q ''BIS WR[@2] WITH LS[@1] TO Q''
;1255    AND LS[] WITH WR[] TO Q         ''AND WR[@2] WITH LS[@1] TO Q''
;1256    XOR LS[] WITH WR[] TO Q         ''XOR WR[@2] WITH LS[@1] TO Q''
;1257
;1258
```

```
;1259    .PAGE
;1260    ; SPECIAL FUNCTION ADD/SUB OPERATIONS
;1261
;1262    ADD (WR[] TO WR[])+1            ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.WR+1>,03F]>''
;1263    ADD (LS[] TO WR[])+1            ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.PLUS.WR+1>,0FF]>,-2]>''
;1264    ADD (WR[] TO LS[])+1            ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.LS+1>,0FF]>,-2]>''
;1265
;1266    ADD OS PLUS WR[] TO LS[]        ''OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/WR.PLUS.OS>,3F]>,-3]>''
;1267
;1268    SUB (WR[] FROM WR[])-1          ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR-1>,03F]>''
;1269    SUB (LS[] FROM WR[])-1          ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR-1>,0FF]>,-2]>''
;1270    SUB (WR[] FROM LS[])-1          ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.FROM.LS-1>,0FF]>,-2]>''
;1271
;1272
;1273    SUB  WRB[] FROM WRA[] TO WRB[]  ''OPC1/EXTENDED,B.ADRS/@1,A.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.WR.WR>,03F]>''
;1274    SUB  LS[]  FROM WR[]  TO LS     ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.FROM.WR.LS>,0FF]>,-2]>''
;1275    SUB  WR[] FROM LS[]  TO WR      ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WRA.FROM.LS.WRB>,0FF]>,-2]>''
;1276
;1277    SUB  WRA[] FROM Q    TO WRB[]   ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.FROM.Q.WR>,03F]>''
;1278    SUB  LS[]  FROM Q    TO LS[]    ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/LS.FROM.Q.LS>,0FF]>,-2]>''
;1279    SUB  Q     FROM WR[] TO Q       ''OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/Q.FROM.WR.Q>,03F]>''
;1280    SUB  Q     FROM LS[] TO Q       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.LS.Q>,0FF]>,-2]>''
;1281
;1282    ADD Q PLUS WR[] TO LS[]         ''OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.WR.TO.LS>,0FF]>,-2]>''
;1283    SUB Q FROM WR[] TO LS[]         ''OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.FROM.WR.TO.LS>,0FF]>,-2]>''
;1284    ADD Q PLUS LS[] TO WR[] ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/Q.PLUS.LS.TO.WR>,0FF]>,-2]>''
;1285
;1286    ADD Q TO WR[] XCHG TO LS[]      ''OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/WR.PLUS.Q.XCHG>,0FF]>,-2]>''
;1287
```

```
;1288    .PAGE
;1289    ; MOVE MACRO INSTRUCTIONS
;1290    ;
;1291    ;       DURING THE MOVE INSTRUCTIONS THE ALU CC'S CAN BE LOADED BY
;1292    ;       ADDING THE
;1293    ;
;1294    ;               DT(SIZE)&SET.ALU.CC
;1295    ;
;1296    ;       MACRO TO THE MICROWORD.  THE FOLLOWING IS A DESCRIPTION OF THE ALU
;1297    ;       CC'S VALUE FOR EACH FUNCTION:
;1298    ;
;1299    ;       MOV -- PUT FIRST OPERAND ON TOP OF THE SECOND OPERAND.
;1300    ;
;1301    ;       N - SIGN BIT
;1302    ;       Z - SET IF ANSWER IS ZERO
;1303    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1304    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1305    ;
;1306    ;       MCOM -- PUT ONE'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
;1307    ;
;1308    ;       N - SIGN BIT
;1309    ;       Z - SET IF ANSWER IS ZERO
;1310    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1311    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1312    ;
;1313    ;       MNEG -- PUT TWO'S COMPLEMENT OF THE FIRST OPERAND ON TOP OF THE SECOND OPERAND.
;1314    ;
;1315    ;       N - SIGN BIT
;1316    ;       Z - SET IF ANSWER IS ZERO.
;1317    ;       V - ARITHMETIC OVERFLOW
;1318    ;       C - CARRY
;1319    MOV Q     TO LS[]                   ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.LS>,0FF]>,-2]>''
;1320    MOV Q     TO WR[]                   ''OPC1/EXTENDED,B.ADRS/@1,XDP/<.AND[<SDP/MOV.Q.WR>,03F]>''
;1321    MOV LS[] TO Q                       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.LS.Q>,0FF]>,-2]>''
;1322    MOV Q TO WR[] XCHG TO LS[]          ''OPC/BASIC,D.ADRS/@2,B.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/MOV.Q.WR&WR.LS>,0FF]>,-2]>''
;1323
;1324    MOV IB.DATA TO OS                   ''OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/IB.REQ,JCTL/NO.JUMP.TST,LD.OS/LOAD.OS''
;1325    MOV CM.IB.DATA TO OS                ''OPC2/DECODE,IFUNC/SPEC,R.DST/NOP,IB.REQ/NOP,JCTL/NO.JUMP.TST,LD.OS/LOAD.OS''
;1326
;1327    MOV MEM.DATA TO WR[]                ''OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/WR.BACKUP''
;1328    MOV MEM.DATA TO WR[] XCHG TO LS[]       ''OPC1/MOVE,B.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.WR>,3F]>,-3]>,XD.ADRS/@2''
;1329    MOV MEM.DATA TO LS[]                ''OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.MEM.LS>,3F]>,-3]>''
;1330    MOV LS[]     TO WR[]                ''OPC1/MOVE,XD.ADRS/@1,B.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.WR>,3F]>,-3]>''
;1331    MOV WR[]     TO LS[]                ''OPC1/MOVE,B.ADRS/@1,XD.ADRS/@2,MDP/<.SHIFT[<.AND[<SDP/MOV.WR.LS>,3F]>,-3]>''
;1332    MOV WR[]     TO WR[]                ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>''
;1333    MOV WR[]     TO Q                   ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.OR.WR.Q>,03F]>''
;1334    MOV XWR[]    TO  Q                  ''OPC1/MOVE,XB.ADRS/@1,XD.ADRS/0,MDP/<.SHIFT[<.AND[<SDP/MOV.XWR.Q>,3F]>,-3]>''
;1335
;1336    MOV FPA TO LS[]                     ''OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>''
;1337    MOV PORT TO LS[]                    ''OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.ACC.LS>,3F]>,-3]>,CC/DT(LONG)&HOLD.ALU.CC''
;1338
;1339    MCOM WR[] TO LS[]                   ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.COM.LS>,0FF]>,-2]>''
;1340    MCOM LS[] TO WR[]                   ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.COM.WR>,0FF]>,-2]>''
;1341    MNEG WR[] TO LS[]                   ''OPC/BASIC,B.ADRS/@1,D.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/WR.NEG.LS>,0FF]>,-2]>''
;1342    MNEG LS[] TO WR[]                   ''OPC/BASIC,D.ADRS/@1,B.ADRS/@2,DP/<.SHIFT[<.AND[<SDP/LS.NEG.WR>,0FF]>,-2]>''
```

```
;1343   MNEG WR[] TO WR[]              ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>''
;1344   MCOM WR[] TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.COM.WR>,03F]>''
;1345   MINC WR[] TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.INC.WR>,03F]>''
;1346   MDEC WR[] TO WR[]        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>''
;1347   MNEG Q TO LS[]           ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/Q.NEG.LS>,0FF]>,-2]>''
```

```
;1348    .PAGE
;1349    ;
;1350    ; SINGLE OPERAND OPERATIONS
;1351    ;
;1352    ;       THIS FORM OF INSTRUCTION PERFORMS THE DESIRED OPERATION ON THE OPERAND
;1353    ;       SPECIFIED.
;1354    ;
;1355    ;               CLEAR
;1356    ;               NEGATE
;1357    ;               INCREMENT
;1358    ;               DECREMENT
;1359    ;               COMPLEMENT
;1360    ;               TEST
;1361    ;
;1362    ;       DURING THE SINGLE OPERAND INSTRUCTIONS THE ALU CC'S CAN BE
;1363    ;       LOADING USING THE
;1364    ;
;1365    ;               DT(SIZE)&SET.ALU.CC
;1366    ;
;1367    ;       MACRO IN THE MICROWORD.  THE FOLLOWING IS A DESCRIPTION OF THE
;1368    ;       ALU CC'S FOR EACH FUNCTION:
;1369    ;
;1370    ;       CLR -- STORE A ZERO IN THE OPERAND.
;1371    ;
;1372    ;       N - SIGN BIT
;1373    ;       Z - SET IF ANSWER IS ZERO
;1374    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1375    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1376    ;
;1377    ;       NEG -- PERFORM TWO'S COMPLEMENT OF THE OPERAND.
;1378    ;
;1379    ;       N - SIGN BIT
;1380    ;       Z - SET IF ANSWER IS ZERO.
;1381    ;       V - ARITHMETIC OVERFLOW
;1382    ;       C - CARRY
;1383    ;
;1384    ;       INC -- ADD ONE TO THE OPERAND.
;1385    ;
;1386    ;       N - SIGN BIT
;1387    ;       Z - SET IF ANSWER IS ZERO.
;1388    ;       V - ARITHMETIC OVERFLOW
;1389    ;       C - CARRY
;1390    ;
;1391    ;       DEC -- SUBTRACT ONE FROM THE OPERAND.
;1392    ;
;1393    ;       N - SIGN BIT
;1394    ;       Z - SET IF ANSWER IS ZERO.
;1395    ;       V - ARITHMETIC OVERFLOW
;1396    ;       C - CARRY
;1397    ;
;1398    ;       COM -- PERFORM ONE'S COMPLEMENT OF THE OPERAND (XNOR WITH ZERO).
;1399    ;
;1400    ;       N - SIGN BIT
;1401    ;       Z - SET IF ANSWER IS ZERO.
;1402    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
```

```
;1403    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1404    ;
;1405    ;       TST -- ADD ZERO TO THE OPERAND TO SET THE CONDITION CODES.
;1406    ;
;1407    ;       N - SIGN BIT
;1408    ;       Z - SET IF ANSWER IS ZERO
;1409    ;       V - ARITHMETIC OVERFLOW (IN THIS CASE ALWAYS ZERO)
;1410    ;       C - CARRY (IN THIS CASE ZERO)
;1411    ;
;1412
;1413    CLR Q                           ''OPC1/EXTENDED,A.ADRS/0,B.ADRS/0,XDP/<.AND[<SDP/WR.XOR.WR.Q>,03F]>''
;1414    CLR LS[]                        ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/CLR.LS>,0FF]>,-2]>''
;1415    CLR WR[]                        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.XOR.WR>,03F]>''
;1416
;1417    NEG  Q                          ''OPC1/EXTENDED,XDP/<.AND[<SDP/NEG.Q>,03F]>''
;1418    NEG  LS[]                       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/NEG.LS>,0FF]>,-2]>''
;1419    NEG  WR[]                       ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.NEG.WR>,03F]>''
;1420
;1421    INC  Q                          ''OPC1/EXTENDED,XDP/<.AND[<SDP/INC.Q>,03F]>''
;1422    INC  LS[]                       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/INC.LS>,0FF]>,-2]>''
;1423    INC  WR[]                       ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.INC.WR>,03F]>''
;1424
;1425    DEC  Q                          ''OPC1/EXTENDED,XDP/<.AND[<SDP/DEC.Q>,03F]>''
;1426    DEC  LS[]                       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/DEC.LS>,0FF]>,-2]>''
;1427    DEC  WR[]                       ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.DEC.WR>,03F]>''
;1428
;1429    COM  Q                          ''OPC1/EXTENDED,XDP/<.AND[<SDP/COM.Q>,03F]>''
;1430    COM  LS[]                       ''OPC/BASIC,D.ADRS/@1,DP/<.SHIFT[<.AND[<SDP/COM.LS>,0FF]>,-2]>''
;1431    COM  WR[]                       ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.COM.WR>,03F]>''
;1432
;1433    TST WR[]                        ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,XDP/<.AND[<SDP/WR.PLUS.0.TO.WR>,03F]>''
;1434    TST Q                           ''OPC1/EXTENDED,XDP/<.AND[<SDP/Q.PLUS.0>,03F]>''
;1435
;1436    NOP                             ''OPC/BASIC,D.ADRS/0,B.ADRS/0,DP/<.SHIFT[<.AND[<SDP/Q.CMP.LS>,0FF]>,-2]>''
```

```
;1437    .PAGE
;1438    ;
;1439    ; MACROS FOR SHIFT OPERATION ON WR[B] AND/OR Q-REGISTER
;1440    ;
;1441    ;       DURING THE SHIFT OPERATIONS THE ALU CC'S CAN BE LOADED BY
;1442    ;       USING THE
;1443    ;
;1444    ;               DT(SIZE)&SET.ALU.CC
;1445    ;
;1446    ;       MACRO WITH THE MICROWORD.  THE FOLLOWING IS A DESCRIPTION OF THE
;1447    ;       ALU CC'S WITH EACH FUNCTION.
;1448    ;
;1449    ;       ROR WR[] -- ROTATE 32 BIT VALUE ONE POSITION RIGHT.
;1450    ;
;1451    ;       N - SIGN OF INPUT
;1452    ;       Z - SET IF INPUT IS ZERO
;1453    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1454    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1455    ;
;1456    ;       ROL WR[] -- ROTATE 32 BIT VALUE ONE POSITION LEFT.
;1457    ;
;1458    ;       N - SIGN OF INPUT
;1459    ;       Z - SET IF INPUT IS ZERO
;1460    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1461    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1462    ;
;1463    ;       ASHR WR[] -- SHIFT 32 BIT VALUE RIGHT ONE POSITION AND SIGN EXTEND.
;1464    ;
;1465    ;       N - SIGN OF INPUT
;1466    ;       Z - SET IF INPUT IS ZERO
;1467    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1468    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1469    ;
;1470    ;       ASHL WR[] -- SHIFT 32 BIT VALUE LEFT ONE POSITION AND INSERT A ZERO IN BOTTOM BIT.
;1471    ;
;1472    ;       N - SIGN OF INPUT
;1473    ;       Z - SET IF INPUT IS ZERO
;1474    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1475    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1476    ;
;1477    ;       ASHRC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND SIGN EXTEND.
;1478    ;
;1479    ;       N - SIGN OF INPUT WR[]
;1480    ;       Z - SET IF INPUT WR[] IS ZERO
;1481    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1482    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1483    ;
;1484    ;       RORC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION RIGHT.
;1485    ;
;1486    ;       N - SIGN OF INPUT WR[]
;1487    ;       Z - SET IF INPUT WR[] IS ZERO
;1488    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1489    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1490    ;
;1491    ;       ASHLC WR[].Q -- SHIFT 64 BIT WR.Q ONE POSITION LEFT AND INSERT ZERO.
```

```
;1492    ;
;1493    ;       N - SIGN OF INPUT WR[]
;1494    ;       Z - SET IF INPUT WR[] IS ZERO
;1495    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1496    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1497    ;
;1498    ;       ROLC WR[].Q -- ROTATE 64 BIT WR.Q ONE POSITION LEFT.
;1499    ;
;1500    ;       N - SIGN OF INPUT WR[]
;1501    ;       Z - SET IF INPUT WR[] IS ZERO
;1502    ;       V - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1503    ;       C - RANDOM GARBAGE (DO NOT ASSUME ANY FUNCTION)
;1504    ;
;1505    ;       SHL2 WR[] -- SHIFT 32 BIT VALUE TWO POSITIONS LEFT INSERTING ZEROES IN THE BOTTOM BITS.
;1506    ;
;1507    ;       N - SIGN OF INPUT WR[]+WR[]
;1508    ;       Z - SET IF INPUT WR[]+WR[] IS ZERO
;1509    ;       V - OVERFLOW OF INPUT WR[]+WR[]
;1510    ;       C - CARRY OF INPUT WR[]+WR[]
;1511    ;
;1512    ROR  WR[]                       ''OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHR>,03F]>''
;1513
;1514    ROL  WR[]                       ''OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR,XDP/<.AND[<SDP/ASHL>,03F]>''
;1515
;1516
;1517    ASHR  WR[]                      ''OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHR>,03F]>''
;1518
;1519    ASHL  WR[]                      ''OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR,XDP/<.AND[<SDP/ASHL>,03F]>''
;1520
;1521    ASHRC  WR[].Q                   ''OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>''
;1522
;1523    RORC WR[].Q                     ''OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>''
;1524
;1525    ASHLC  WR[].Q                   ''OPC1/EXTENDED,B.ADRS/@1,SI/ASH.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>''
;1526
;1527    ROLC WR[].Q                     ''OPC1/EXTENDED,B.ADRS/@1,SI/ROT.WR.Q,XDP/<.AND[<SDP/ASHLC>,03F]>''
;1528
;1529    SHL2  WR[]                      ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@1,SI/2,XDP/<.AND[<SDP/SHL2>,03F]>''
;1530
;1531    COND.ADD&SHIFT  WR[] TO WR[].Q  ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>''
;1532
;1533    COND.SUB&SHIFT  WR[] FROM WR[].Q ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/MUL.SHF,XDP/<.AND[<SDP/COND.SUB&SHIFT>,03F]>''
;1534
;1535    UNSIGNED.MUL WR[] TO WR[].Q     ''OPC1/EXTENDED,A.ADRS/@1,B.ADRS/@2,SI/UMUL.SHF,XDP/<.AND[<SDP/COND.ADD&SHIFT>,03F]>''
;1536    SHR WR[]                        ''OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHR>,03F]>''
;1537    SHL WR[]                        ''ASHL WR[@1]''
;1538    SHRC WR[].Q                     ''OPC1/EXTENDED,B.ADRS/@1,SI/SHF.WR.Q,XDP/<.AND[<SDP/ASHRC>,03F]>''
;1539    SHLC WR[].Q                     ''ASHLC WR[].Q''
;1540
;1541
;1542    ; MEMORY REQUEST MACRO
;1543
;1544    ; *      THIS MACRO IS NOT FOR GENERAL USE!!!            *
;1545    ; *      THIS IS ONLY FOR USE IN THE FOLLOWING MACRO     *
;1546    MEM.FUNC[]                      'M.FUNC2/<.SHIFT[<MEM.FUNC/@1>,-2]>,M.FUNC1/<.AND[<MEM.FUNC/@1>,3]>''
```

```
;1547
;1548   MEM.REQ[] ADRS[] DT[]          ''OPC2/MEM.REQ,MEM.FUNC[@1],D.ADRS/@2,MDT/@3''
;1549   WRITE.MEM LS[]                 ''OPC1/MOVE,XD.ADRS/@1,MDP/<.SHIFT[<.AND[<SDP/MOV.LS.MEM>,3F]>,-3]>''
```

```
;1550    .PAGE
;1551    ;
;1552    ; MACROS FOR MISCELLANEOUS OPERATIONS
;1553    ;
;1554    ;       IN THE NEBULA MICROPROGRAMMING LANGUAGE THERE IS A NEED FOR
;1555    ;       VARIOUS UNIQUE FUNCTIONAL OPERATIONS.  THESE ARE FOUND IN THE
;1556    ;       MISC INSTRUCTIONS.  MOST FUNCTIONS ARE EXECUTED BY INSERTING
;1557    ;       THE REQUEST INSIDE THE MISC[] AND MISC2[] MACROS.
;1558    ;
;1559    MISC []                         "OPC2/MISC,MISC.1/@1,MISC.2/NOP,MISC.PORT/MISC"
;1560    MISC2 []                        "OPC2/MISC,MISC.1/NOP,MISC.2/@1,MISC.PORT/MISC"
;1561    MISC [] WR[]                    "MISC [@1],MISC.WRL/0,MISC.WRR/@2"
;1562    MISC2 [] WR[]                   "MISC2 [@1],MISC.WRL/0,MISC.WRR/@2"
;1563
;1564    PORT.CONTROL []                 "OPC2/MISC,MISC.PORT/PORT,CNTL.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/CONTROL"
;1565    PORT.WRITE PORT.ADRS[] WITH WR[]        "OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/WRITE,MISC.WRL/0,MIS
;1566    PORT.READ PORT.ADRS[]   "OPC2/MISC,MISC.PORT/PORT,R.W.FUNC/@1,PORT.SELECT/IDC,PORT.FUNC/READ"
;1567    ;
;1568    ;       A FEW CASES OF THE MISCELLANEOUS FUNCTIONS HAVE BEEN CALLED OUT AS
;1569    ;       UNIQUE FUNCTIONS IN THE MICROPROGRAMMING LANGUAGE SET.  THESE
;1570    ;       ARE LISTED BELOW.
;1571    ;
;1572    ;       THE FOLLOWING TO MACROS SEND DATA TO EXTERNAL ACCELLERATORS.
;1573    ;
;1574    ASSERT.DA.AND.PASS.WR0          "MISC2 [ASSERT.DA.AND.PASS.WR] WR[0]"
;1575    ASSERT.DA.AND.PASS.WR1          "MISC2 [ASSERT.DA.AND.PASS.WR] WR[1]"
;1576    ;
;1577    ;       STATE REGISTER MACROS - THE POSSIBLE CHANGES TO THE STATE
;1578    ;       BITS ARE SPECIFIED AS INDIVIDUAL FUNCTIONS.
;1579    ;
;1580    CLR.STATE.ZERO                  "OPC2/MISC,MISC.1/CLR.S0,MISC.2/NOP,MISC.PORT/MISC"
;1581    CLR.STATE.ONE                   "OPC2/MISC,MISC.1/CLR.S1,MISC.2/NOP,MISC.PORT/MISC"
;1582    SET.STATE.ZERO                  "OPC2/MISC,MISC.1/SET.S0,MISC.2/NOP,MISC.PORT/MISC"
;1583    SET.STATE.ONE                   "OPC2/MISC,MISC.1/SET.S1,MISC.2/NOP,MISC.PORT/MISC"
;1584    SET.STATE.ZERO&CLR.STATE.ONE    "OPC2/MISC,MISC.1/CLR.S1&SET.S0,MISC.2/NOP,MISC.PORT/MISC"
;1585    SET.STATE.ONE&CLR.STATE.ZERO    "OPC2/MISC,MISC.1/SET.S1&CLR.S0,MISC.2/NOP,MISC.PORT/MISC"
;1586    CLR.STATE.ONE&CLR.STATE.ZERO    "OPC2/MISC,MISC.1/CLR,MISC.2/NOP,MISC.PORT/MISC"
;1587    SET.BACKUP.MASK.VALID           "OPC2/MISC,MISC.1/SET.BACKUP.MASK.VALID,MISC.2/NOP,MISC.PORT/MISC"
;1588
;1589
;1590    ; CONTEXT SIZE AND CONDITION CODE MACROS
;1591
;1592    DT(SIZE)&MAP.ALU.CC.TO.PSL      "CC/DT(SIZE)&MAP.ALU.CC.TO.PSL"
;1593    DT(LONG)&SET.ALU.CC             "CC/DT(LONG)&SET.ALU.CC"
;1594    DT(SIZE)&SET.ALU.CC             "CC/DT(SIZE)&SET.ALU.CC"
;1595
```

```
;1596    .PAGE
;1597    ; JUMP AND MICRO SEQUENCER CONTROL MACROS
;1598    ;
;1599    ;       THE FOLLOWING MACROS ARE USED TO CONTROL THE PROGRAMS FLOW OF
;1600    ;       CONTROL.
;1601    ;
;1602    ;       JMP [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY.
;1603    ;
;1604    ;       JMP.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF ADDRESS
;1605    ;               IN THE INSTRUCTION AND THE OS REGISTER UNCONDITIONALLY.
;1606    ;
;1607    ;       JMP.IF[] TO [] -- TRANSFER TO NEW ADDRESS ONLY IF JUMP
;1608    ;               CONDITION IS MET.
;1609    ;
;1610    ;       JSR [] -- TRANSFER TO NEW ADDRESS UNCONDITIONALLY AND PUT
;1611    ;               RETURN ADDRESS ON THE SEQUENCER STACK.
;1612    ;
;1613    ;       JSR.OS [] -- TRANSFER TO RESULTANT ADDRESS OF LOGICAL OR OF
;1614    ;               ADDRESS IN THE INSTRUCTION AND THE OS REGISTER.
;1615    ;               THE RETURN ADDRESS IS ALSO PUSHED ON THE SEQUENCER STACK.
;1616    ;
;1617    ;       RETURN -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER STACK AND
;1618    ;                POP THE STACK.
;1619    ;
;1620    ;       RETURN+1 -- TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
;1621    ;               STACK PLUS ONE AND POP THE STACK.
;1622    ;
;1623    ;       RETURN+1.IF(MEM.REF.OK) -- IF THE MOST RECENT MEMORY REQUEST HAD
;1624    ;               NO ERROR THEN TRANSFER TO ADDRESS ON TOP OF THE SEQUENCER
;1625    ;               STACK PLUS ONE AND POP THE STACK.  IF THERE WAS AN
;1626    ;               ERROR THEN SKIP OVER THE ENEXT INSTRUCTION.
;1627    ;
;1628    ;       LOOP.IF(NEQ) -- IF THE ALU Z-BIT IS ZERO (LAST VALUE CALCULATED
;1629    ;               WHEN THE ALU.CC'S WERE SET DID NOT RESULT AS ZERO)
;1630    ;               THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
;1631    ;               STACK.  THE STACK IS NOT POPPED.  IF THE ALU Z-BIT
;1632    ;               IS ONE THEN CONTINUE WITH THE NEXT INSTRUCTION.
;1633    ;               (UPC+1) AND THE STACK IS POPPED.
;1634    ;
;1635    ;       LOOP.IF(GEQU) -- IF THE ALU C-BIT IS ONE (LAST VALUE CALCULATED WHEN
;1636    ;               THE ALU.CC'S WERE SET WAS GREATER THAN OR EQUAL TO ZERO)
;1637    ;               THEN TRANSFER TO THE ADDRESS ON TOP OF THE SEQUENCER
;1638    ;               STACK.  THE STACK IS NOT POPPED.  IF THE ALU C-BIT
;1639    ;               IS ZERO THEN CONTINUE WITH THE NEXT INSTRUCTION
;1640    ;               (UPC+1) AND THE STACK IS POPPED.
;1641    ;
;1642    ;       LOOP.IF(NO INTERRUPT AND NO SYNC) -- IF THE INTERRUPT CONDITION
;1643    ;               IS NOT TRUE AND THE ACCELERATOR SYNC CONDITION IS NOT
;1644    ;               TRUE THEN TRANSFER TO THE ADDRESS ON THE TOP OF THE SEQUENCER
;1645    ;               STACK.  THE STACK IS NOT POPPED.  IF THE INTERRUPT
;1646    ;               CONDITION IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
;1647    ;               (UPC+1) AND THE SEQUENCER STACK IS NOT POPPED.  IF THE ACCELERATOR
;1648    ;               SYNC IS TRUE THEN CONTINUE WITH THE NEXT INSTRUCTION
;1649    ;               (UPC+1) AND THE SEQUENCER STACK IS POPPED.
;1650    ;
```

```
;1651    ;       SKIP.IF[] -- IF THE SKIP CONDITION IS SATISFIED THEN TRANSFER
;1652    ;               TO UPC+2 ELSE TRANSFER TO UPC+1.
;1653    ;
;1654
;1655    JMP []                          ''OPC2/JUMP,JUMP.ADRS/@1,JCTL/JUMP''
;1656    JMP.OS []                       ''OPC2/JUMP,JUMP.ADRS/@1,OS/WITH.OS,JCTL/JUMP''
;1657    JMP.IF[] TO []                  ''OPC2/JUMP,JCTL/@1,JUMP.ADRS/@2''
;1658    JSR []                          ''OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1''
;1659    JSR.OS []                       ''OPC2/JUMP,JCTL/JSR,JUMP.ADRS/@1,OS/WITH.OS''
;1660
;1661    RETURN                          ''SCTL/RETURN''
;1662    RETURN+1                        ''SCTL/RETURN+1''
;1663    RETURN+1.IF(MEM.REF.OK)         ''SCTL/RET.NOERR''
;1664
;1665    LOOP.IF(NEQ)                    ''SCTL/JMP.Z.CLR''
;1666    LOOP.IF(GEQU)                   ''SCTL/JMP.C.SET''
;1667    LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)    ''SCTL/LOOP.IF.NO.I.AND.SYNC''
;1668
;1669    SKIP.IF[]                       ''SCTL/@1''
;1670    SKIP                            ''SCTL/SKIP''
```

```
;1671    .PAGE
;1672    ;
;1673    ; INSTRUCTION STREAM MACROS
;1674    ;
;1675    ; *     THIS MACRO IS NOT FOR GENERAL USE!!!            *
;1676    ; *     THIS IS ONLY FOR USE IN THE FOLLOWING MACROS    *
;1677    DEC[]                           "OPC2/DECODE,DEC.ADRS/<.SHIFT[<JUMP.ADRS/@1>,-8]>"
;1678
;1679    DECODE.SPEC ADRS[]              "DEC[@1],IFUNC/SPEC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/S
;1680    DECODE.ASRC ADRS[]              "DEC[@1],IFUNC/ASRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/A
;1681    DECODE.ESRC ADRS[]              "DEC[@1],IFUNC/ESRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/E
;1682    DECODE.VSRC ADRS[]              "DEC[@1],IFUNC/VSRC,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/V
;1683    DECODE.FLOAT ADRS[]             "DEC[@1],IFUNC/FLOAT,BPC/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,LD.OS/LOAD.OS,R.DST/ENAB,OPC.SPEC/
;1684    DECODE.OPC1 ADRS[]              "DEC[@1],IFUNC/VAX.IRD,R.DST/NOP,CM.HI/LOW.BYTE,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD"
;1685    EXEC.DISPATCH ADRS[]            "DEC[@1],IFUNC/VAX.EXEC,IB.REQ/NOP,R.DST/NOP,CM.HI/LOW.BYTE,JCTL/JSR,OPC.SPEC/VAX.EXEC"
;1686
;1687    DECODE.CM.OPC ADRS[]            "DEC[@1],IFUNC/CM.IRD,CM.HI/HI.BYTE,OPC.SPEC/CM.IRD,LD.OS/LOAD.OS"
;1688    DECODE.CM.DST ADRS[]            "DEC[@1],IFUNC/CM.DST,OPC.SPEC/CM.DST,LD.OS/LOAD.OS,R.DST/ENAB"
;1689    DECODE.CM.SINGLE ADRS[]         "DEC[@1],IFUNC/CM.SINGLE,OPC.SPEC/CM.SINGLE"
;1690    CM.EXEC ADRS[]                  "DEC[@1],IFUNC/CM.EXEC,JCTL/JUMP,OPC.SPEC/CM.EXEC,LD.OS/LOAD.OS"
;1691
;1692    SAVE.PC WR0                     "BPC/SAVE.PC"
;1693    SAVE.PC WR4                     "BPC/SAVE.PC"
;1694    SAVE.CM.PC WR0                  "BPC/SAVE.PC"
;1695    SAVE.CM.PC WR4                  "BPC/SAVE.PC"
;1696
;1697
;1698    ; THESE SKIP CONDITIONS ARE USED FOR THE DECODE MICRO INSTRUCTION
;1699    ;
;1700    SKIP.IF(IB VALID)               "JCTL/NO.JUMP.TST"
;1701    JMP.IF(IB VALID)                "JCTL/JUMP.VALID"
;1702    JSR.IF(IB VALID)                "JCTL/JSR.VALID"
;1703    ;
;1704    ;       THESE SKIP CONDITIONS ARE TO BE USED WITH THE COMPATIBILITY MODE DECODE
;1705    ;       MACROS.
;1706    ;
;1707    JMP.TO []                       "JCTL/JUMP,DEC[@1]"
;1708    JSR.TO []                       "JCTL/JSR,DEC[@1]"
                ;1709   .BIN
```

```
;1710    .PAGE
;1711    .TOC    "FIELD DEFINTITIONS FOR DATA PATH CONTROL MICRO WORD    VER 00.01"
;1712
;1713    ; THIS SECTION IS FOR THE DATA PATH CONTROL ROMS
;1714    ;
;1715
;1716    .CCODE
;1717    SDP/=<8:0>,.ADDRESS,.VALIDITY=0
;1718    ;
;1719    ; THIS FIELD CORRESPONDS TO THE 2901 ALU FUNCTION CONTROL PINS ... I5,I4,I3
;1720    ;
;1721    ALU/=<2:0>,.DEFAULT=<ALU/R.OR.S>
;1722
;1723            R.PLUS.S=0              ; ADD R WITH S
;1724            S.MINUS.R=1             ; SUBTRACT R FROM S
;1725            R.MINUS.S=2             ; SUBTRACT S FROM R
;1726            R.OR.S=3                ; OR R WITH S
;1727            R.AND.S=4               ; AND R WITH S
;1728            NOTR.AND.S=5            ; COMPLEMENT R THEN AND WITH S
;1729            R.XOR.S=6               ; XOR R WITH S
;1730            R.XNOR.S=7              ; XOR R WITH S THEN COMPLEMENT THE RESULT
;1731    SRC/=<8:6>,.DEFAULT=<SRC/D.0>
;1732
;1733            0.Q=2                   ; RMX=0         SMX=Q
;1734            0.B=3                   ; RMX=0         SMX=B
;1735            A.Q=0                   ; RMX=A         SMX=Q
;1736            A.B=01                  ; RMX=A         SMX=B
;1737            D.Q=06                  ; RMX=D         SMX=Q
;1738            D.0=7                   ; RMX=D         SMX=0
;1739            0.A=4                   ; RMX=0         SMX=A
;1740            D.A=5                   ; RMX=D         SMX=A
;1741    ;
;1742    ; THIS FIELD CORRESPONDES TO THE 2901 ALU DESTINATION
;1743    ; CONTROL PINS I8, I7 AND I6.
;1744    ;
;1745    DST/=<5:3>,.DEFAULT=<DST/NOP>
;1746
;1747            LOAD.Q=0                ; LOAD Q-REGISTER
;1748            NOP=1                   ; HOLD ALL REGISTERS
;1749            WRITE.B.A=2             ; WRITE RAM B-PORT AND OUTPUT RAM A-PORT
;1750            WRITE.B.F=3             ; WRITE RAM B-PORT AND OUTPUT ALU
;1751            RSHF.RAM.Q=4            ; RIGHT SHIFT RAM AND Q
;1752            RSHF.RAM=5              ; RIGHT SHIFT RAM
;1753            LSHF.RAM.Q=6            ; LEFT SHIFT RAM AND Q
;1754            LSHF.RAM=7              ; LEFT SHIFT RAM
;1755
;1756    CIN/=<9>,.DEFAULT=0
;1757
;1758            NO.CIN=0                ; NO CARRY IN TO ALU
;1759            CIN=1                   ; CARRY IN TO ALU
;1760
;1761
;1762
```

```
                ;1763   .PAGE
                ;1764   .TOC    "CONTROL ROM CONTENTS                                          VER 00.01"
                ;1765   .SEQUENTIAL
                ;1766   ; THESE ARE THE DATA PATH CONTROL ROM CONTENTS
                ;1767   ;
                ;1768   ; THE FOLLOWING LOCATIONS ARE USED BY THE DECODE.IS MICRO INSTRUCTION
                ;1769   ;
                ;1770   ; THIS INSTRUCTION WILL DO PC+1 OPERATIONS
                ;1771   ;
                ;1772   ;               BACK UP PC IN 2901 RAM
                ;1773   000:
                ;1774   DECODE.IS:
C 0000, 3D8     ;1775           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0001, 3D8     ;1776           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0002, 3D8     ;1777           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0003, 3D8     ;1778           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0004, 3D8     ;1779           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0005, 3D8     ;1780           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0006, 3D8     ;1781           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0007, 3D8     ;1782           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0008, 3D8     ;1783           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0009, 3D8     ;1784           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000A, 3D8     ;1785           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000B, 3D8     ;1786           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000C, 3D8     ;1787           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000D, 3D8     ;1788           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000E, 3D8     ;1789           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 000F, 3D8     ;1790           SRC/D.0,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
                ;1791   ;
                ;1792   ;               DON'T BACK UP PC
                ;1793   ;
C 0010, 3C8     ;1794           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0011, 3C8     ;1795           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0012, 3C8     ;1796           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0013, 3C8     ;1797           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0014, 3C8     ;1798           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0015, 3C8     ;1799           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0016, 3C8     ;1800           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0017, 3C8     ;1801           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0018, 3C8     ;1802           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 0019, 3C8     ;1803           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001A, 3C8     ;1804           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001B, 3C8     ;1805           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001C, 3C8     ;1806           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001D, 3C8     ;1807           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001E, 3C8     ;1808           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 001F, 3C8     ;1809           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
```

```
                ;1810   .PAGE
                ;1811   ;  THESE ADDRESSES ARE USED BY THE 'JUMP' MICRO INSTRUCTION
                ;1812   ;
                ;1813   ; THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
                ;1814   ; THE 2901 DURING JUMP OR JSR OPERATIONS.
                ;1815   ;
                ;1816   020:
                ;1817   JUMP:
C 0020, 3CB     ;1818           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0021, 3CB     ;1819           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0022, 3CB     ;1820           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0023, 3CB     ;1821           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0024, 3CB     ;1822           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0025, 3CB     ;1823           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0026, 3CB     ;1824           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0027, 3CB     ;1825           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0028, 3CB     ;1826           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0029, 3CB     ;1827           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002A, 3CB     ;1828           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002B, 3CB     ;1829           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002C, 3CB     ;1830           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002D, 3CB     ;1831           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002E, 3CB     ;1832           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 002F, 3CB     ;1833           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0030, 3CB     ;1834           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0031, 3CB     ;1835           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0032, 3CB     ;1836           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0033, 3CB     ;1837           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0034, 3CB     ;1838           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0035, 3CB     ;1839           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0036, 3CB     ;1840           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0037, 3CB     ;1841           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0038, 3CB     ;1842           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 0039, 3CB     ;1843           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003A, 3CB     ;1844           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003B, 3CB     ;1845           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003C, 3CB     ;1846           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003D, 3CB     ;1847           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003E, 3CB     ;1848           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
C 003F, 3CB     ;1849           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/CIN
```

```
                ;1850   .PAGE
                ;1851   ; THESE LOCATIONS ARE USED BY THE MISC' MICRO INSTRUCTION
                ;1852   ;
                ;1853   ; THE FOLLOWING CONTROL CODES WILL INSURE THAT NO DATA IS DESTROYED WITHIN
                ;1854   ; THE 2901 DURING MISC INSTRUCTION EXECUTION.
                ;1855   ;
                ;1856   040:
                ;1857   MISC:
C 0040, 313     ;1858           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0041, 313     ;1859           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0042, 313     ;1860           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0043, 313     ;1861           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0044, 313     ;1862           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0045, 313     ;1863           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0046, 313     ;1864           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0047, 313     ;1865           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0048, 313     ;1866           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0049, 313     ;1867           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004A, 313     ;1868           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004B, 313     ;1869           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004C, 313     ;1870           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004D, 313     ;1871           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004E, 313     ;1872           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 004F, 313     ;1873           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0050, 313     ;1874           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0051, 313     ;1875           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0052, 313     ;1876           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0053, 313     ;1877           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0054, 313     ;1878           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0055, 313     ;1879           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0056, 313     ;1880           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0057, 313     ;1881           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0058, 313     ;1882           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0059, 313     ;1883           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005A, 313     ;1884           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005B, 313     ;1885           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005C, 313     ;1886           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005D, 313     ;1887           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005E, 313     ;1888           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 005F, 313     ;1889           SRC/0.A,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
```

```
                ;1890   .PAGE
                ;1891   ; THESE ADDRESSES ARE USED BY THE MEM.REQ MICRO INSTRUCTION
                ;1892   ;
                ;1893   ; THIS INSTRUCTION CAUSES WR[5] TO BE LOADED WITH THE VIRTUAL ADDRESS
                ;1894   ; OF THE MEMORY REFERENCE.
                ;1895   060:
                ;1896   MEM.REQ:
C 0060, 3DB     ;1897           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0061, 3DB     ;1898           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0062, 3DB     ;1899           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0063, 3DB     ;1900           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0064, 3DB     ;1901           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0065, 3DB     ;1902           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0066, 3DB     ;1903           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0067, 3DB     ;1904           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0068, 3DB     ;1905           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0069, 3DB     ;1906           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006A, 3DB     ;1907           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006B, 3DB     ;1908           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006C, 3DB     ;1909           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006D, 3DB     ;1910           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006E, 3DB     ;1911           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 006F, 3DB     ;1912           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0070, 3DB     ;1913           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0071, 3DB     ;1914           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0072, 3DB     ;1915           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0073, 3DB     ;1916           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0074, 3DB     ;1917           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0075, 3DB     ;1918           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0076, 3DB     ;1919           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0077, 3DB     ;1920           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0078, 3DB     ;1921           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 0079, 3DB     ;1922           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007A, 3DB     ;1923           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007B, 3DB     ;1924           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007C, 3DB     ;1925           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007D, 3DB     ;1926           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007E, 3DB     ;1927           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 007F, 3DB     ;1928           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
```

```
                ;1929   .PAGE
                ;1930   ; THESE ARE THE ADDRESSES FOR THE EXTENDED MICRO INSTRUCTION
                ;1931   ; ADDRESSES START AT 80-FF
                ;1932   080:
                ;1933
                ;1934   WR.PLUS.0.TO.WR:
C 0080, 118     ;1935           SRC/0.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
                ;1936   WR.INC.WR:
C 0081, 318     ;1937           SRC/0.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
                ;1938   WR.NEG.WR:
C 0082, 31A     ;1939           SRC/0.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
                ;1940   WR.COM.WR:
C 0083, 31F     ;1941           SRC/0.A,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
                ;1942
                ;1943   WR.DEC.WR:
C 0084, 119     ;1944           SRC/0.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
                ;1945   FILL85:
C 0085, 10B     ;1946           SRC/0.A
                ;1947   MOV.Q.WR:
C 0086, 29B     ;1948           SRC/0.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
                ;1949   FILL87:
C 0087, 08B     ;1950           SRC/0.Q
                ;1951
                ;1952   COND.ADD&SHIFT:
C 0088, 060     ;1953           SRC/A.B,ALU/R.PLUS.S,DST/RSHF.RAM.Q,CIN/NO.CIN
                ;1954   COND.SUB&SHIFT:
C 0089, 261     ;1955           SRC/A.B,ALU/S.MINUS.R,DST/RSHF.RAM.Q,CIN/CIN
                ;1956   FILL8A:
C 008A, 04B     ;1957           SRC/A.B
                ;1958   SHL2:
C 008B, 078     ;1959           SRC/A.B,ALU/R.PLUS.S,DST/LSHF.RAM,CIN/NO.CIN
                ;1960
                ;1961   ASHRC:
C 008C, 0E3     ;1962           SRC/0.B,ALU/R.OR.S,DST/RSHF.RAM.Q,CIN/NO.CIN
                ;1963   ASHR:
C 008D, 2EB     ;1964           SRC/0.B,ALU/R.OR.S,DST/RSHF.RAM,CIN/CIN
                ;1965   ASHLC:
C 008E, 2F3     ;1966           SRC/0.B,ALU/R.OR.S,DST/LSHF.RAM.Q,CIN/CIN
                ;1967   ASHL:
C 008F, 2FB     ;1968           SRC/0.B,ALU/R.OR.S,DST/LSHF.RAM,CIN/CIN
                ;1969
                ;1970   Q.PLUS.0:
C 0090, 088     ;1971           SRC/0.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
                ;1972   FILL91:
C 0091, 08B     ;1973           SRC/0.Q
                ;1974   WR.CMP.Q:
C 0092, 20A     ;1975           SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;1976   Q.CMP.WR:
C 0093, 209     ;1977           SRC/A.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                ;1978
                ;1979   DEC.Q:
C 0094, 081     ;1980           SRC/0.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/NO.CIN
                ;1981   INC.Q:
C 0095, 280     ;1982           SRC/0.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/CIN
                ;1983   NEG.Q:
```

```
C 0096, 282     ;1984           SRC/0.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
                ;1985   COM.Q:
C 0097, 287     ;1986           SRC/0.Q,ALU/R.XNOR.S,DST/LOAD.Q,CIN/CIN
                ;1987
                ;1988   WR.BIT.WR:
C 0098, 04C     ;1989           SRC/A.B,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
                ;1990   WR.CMP.WR:
C 0099, 24A     ;1991           SRC/A.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;1992   FILL9A:
C 009A, 04B     ;1993           SRC/A.B
                ;1994   WR.PLUS.WR+1:
C 009B, 258     ;1995           SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
                ;1996
                ;1997   FILL9C:
C 009C, 04B     ;1998           SRC/A.B
                ;1999   FILL9D:
C 009D, 04B     ;2000           SRC/A.B
                ;2001   FILL9E:
C 009E, 0CB     ;2002           SRC/0.B
                ;2003   FILL9F:
C 009F, 0CB     ;2004           SRC/0.B
                ;2005
                ;2006   WR.PLUS.Q:
C 00A0, 000     ;2007           SRC/A.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
                ;2008   WR.FROM.Q:
C 00A1, 201     ;2009           SRC/A.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
                ;2010   Q.FROM.WR.Q:
C 00A2, 202     ;2011           SRC/A.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
                ;2012   WR.OR.Q:
C 00A3, 203     ;2013           SRC/A.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
                ;2014
                ;2015   WR.AND.Q:
C 00A4, 004     ;2016           SRC/A.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
                ;2017   WR.MASK.Q:
C 00A5, 205     ;2018           SRC/A.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
                ;2019   WR.XOR.Q:
C 00A6, 206     ;2020           SRC/A.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
                ;2021   FILLA7:
C 00A7, 00B     ;2022           SRC/A.Q
                ;2023
                ;2024   WR.PLUS.WR.Q:
C 00A8, 040     ;2025           SRC/A.B,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
                ;2026   WR.FROM.WR.Q:
C 00A9, 241     ;2027           SRC/A.B,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
                ;2028   FILLAA:
C 00AA, 04B     ;2029           SRC/A.B
                ;2030   WR.OR.WR.Q:
C 00AB, 243     ;2031           SRC/A.B,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
                ;2032
                ;2033   WR.AND.WR.Q:
C 00AC, 044     ;2034           SRC/A.B,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
                ;2035   WR.MASK.WR.Q:
C 00AD, 245     ;2036           SRC/A.B,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/CIN
                ;2037   WR.XOR.WR.Q:
C 00AE, 246     ;2038           SRC/A.B,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
```

```
                ;2039   FILLAF:
C 00AF, 04B     ;2040           SRC/A.B
                ;2041
                ;2042   Q.PLUS.WR:
C 00B0, 018     ;2043           SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
                ;2044   WR.FROM.Q.WR:
C 00B1, 219     ;2045           SRC/A.Q,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
                ;2046   Q.FROM.WR:
C 00B2, 21A     ;2047           SRC/A.Q,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
                ;2048   Q.OR.WR:
C 00B3, 21B     ;2049           SRC/A.Q,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
                ;2050
                ;2051   Q.AND.WR:
C 00B4, 01C     ;2052           SRC/A.Q,ALU/R.AND.S,DST/WRITE.B.F,CIN/NO.CIN
                ;2053   FILLB5:
C 00B5, 00B     ;2054           SRC/A.Q
                ;2055   Q.XOR.WR:
C 00B6, 21E     ;2056           SRC/A.Q,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
                ;2057   Q.BIT.WR:
C 00B7, 20C     ;2058           SRC/A.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
                ;2059
                ;2060   WR.PLUS.WR:
C 00B8, 058     ;2061           SRC/A.B,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
                ;2062   WR.FROM.WR:
C 00B9, 259     ;2063           SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
                ;2064   WR.FROM.WR.WR:
C 00BA, 25A     ;2065           SRC/A.B,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
                ;2066   WR.OR.WR:
C 00BB, 25B     ;2067           SRC/A.B,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
                ;2068
                ;2069   WR.FROM.WR-1:
C 00BC, 059     ;2070           SRC/A.B,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
                ;2071   WR.MASK.WR:
C 00BD, 25D     ;2072           SRC/A.B,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
                ;2073   WR.XOR.WR:
C 00BE, 25E     ;2074           SRC/A.B,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
                ;2075   WR.AND.WR:
C 00BF, 25C     ;2076           SRC/A.B,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
                ;2077
```

```
                    ;2078   .PAGE
                    ;2079   ; THIS IS THE DP CODES FOR THE 'MOVE' MICRO INSTRUCTION
                    ;2080   ;
                    ;2081   0C0:
                    ;2082
                    ;2083   MOV.MEM.WR:
C 00C0, 1D3         ;2084           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C1, 1D3         ;2085           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C2, 1D3         ;2086           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C3, 1D3         ;2087           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C4, 1D3         ;2088           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C5, 1D3         ;2089           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C6, 1D3         ;2090           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 00C7, 1D3         ;2091           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
                    ;2092
                    ;2093   MOV.LS.MEM:
C 00C8, 1CB         ;2094           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00C9, 1CB         ;2095           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CA, 1CB         ;2096           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CB, 1CB         ;2097           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CC, 1CB         ;2098           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CD, 1CB         ;2099           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CE, 1CB         ;2100           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00CF, 1CB         ;2101           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
                    ;2102
                    ;2103   MOV.XWR.Q:
C 00D0, 0C3         ;2104           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D1, 0C3         ;2105           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D2, 0C3         ;2106           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D3, 0C3         ;2107           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D4, 0C3         ;2108           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D5, 0C3         ;2109           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D6, 0C3         ;2110           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 00D7, 0C3         ;2111           SRC/0.B,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
                    ;2112
                    ;2113   MOV.LS.WR:
C 00D8, 1DB         ;2114           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00D9, 1DB         ;2115           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DA, 1DB         ;2116           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DB, 1DB         ;2117           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DC, 1DB         ;2118           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DD, 1DB         ;2119           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DE, 1DB         ;2120           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
C 00DF, 1DB         ;2121           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.F,CIN/NO.CIN
                    ;2122
```

```
                ;2123   .PAGE
                ;2124   MOV.MEM.LS:
C 00E0, 1CB     ;2125           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E1, 1CB     ;2126           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E2, 1CB     ;2127           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E3, 1CB     ;2128           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E4, 1CB     ;2129           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E5, 1CB     ;2130           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E6, 1CB     ;2131           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E7, 1CB     ;2132           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
                ;2133   MOV.ACC.LS:
C 00E8, 1CB     ;2134           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00E9, 1CB     ;2135           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EA, 1CB     ;2136           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EB, 1CB     ;2137           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EC, 1CB     ;2138           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00ED, 1CB     ;2139           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EE, 1CB     ;2140           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00EF, 1CB     ;2141           SRC/D.0,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
                ;2142
                ;2143
                ;2144   WR.PLUS.OS:
C 00F0, 148     ;2145           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F1, 148     ;2146           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F2, 148     ;2147           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F3, 148     ;2148           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F4, 148     ;2149           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F5, 148     ;2150           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F6, 148     ;2151           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 00F7, 148     ;2152           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
                ;2153
                ;2154   MOV.WR.LS:
C 00F8, 10B     ;2155           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00F9, 10B     ;2156           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FA, 10B     ;2157           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FB, 10B     ;2158           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FC, 10B     ;2159           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FD, 10B     ;2160           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FE, 10B     ;2161           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
C 00FF, 10B     ;2162           SRC/0.A,ALU/R.OR.S,DST/NOP,CIN/NO.CIN
                ;2163
```

```
                ;2164   .PAGE
                ;2165   ; THIS GROUP OF INSTRUCTIONS IS FOR THE 'BASIC' MICRO INSTRUCTION
                ;2166   ; THIS USES ADDRESSES 100-1FF
                ;2167
                ;2168   100:
                ;2169   LS.PLUS.WR:
C 0100, 158     ;2170           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0101, 158     ;2171           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0102, 158     ;2172           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0103, 158     ;2173           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
                ;2174   LS.FROM.WR:
C 0104, 359     ;2175           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0105, 359     ;2176           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0106, 359     ;2177           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0107, 359     ;2178           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
                ;2179   LS.AND.WR:
C 0108, 35C     ;2180           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 0109, 35C     ;2181           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010A, 35C     ;2182           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
C 010B, 35C     ;2183           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.F,CIN/CIN
                ;2184   LS.XOR.WR:
C 010C, 35E     ;2185           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010D, 35E     ;2186           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010E, 35E     ;2187           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
C 010F, 35E     ;2188           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.F,CIN/CIN
                ;2189
                ;2190   LS.FROM.WR-1:
C 0110, 159     ;2191           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0111, 159     ;2192           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0112, 159     ;2193           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
C 0113, 159     ;2194           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/NO.CIN
                ;2195   LS.MASK.WR:
C 0114, 35D     ;2196           SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0115, 35D     ;2197           SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0116, 35D     ;2198           SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
C 0117, 35D     ;2199           SRC/D.A,ALU/NOTR.AND.S,DST/WRITE.B.F,CIN/CIN
                ;2200   LS.PLUS.WR+1:
C 0118, 358     ;2201           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 0119, 358     ;2202           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011A, 358     ;2203           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
C 011B, 358     ;2204           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/CIN
                ;2205   LS.OR.WR:
C 011C, 35B     ;2206           SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011D, 35B     ;2207           SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011E, 35B     ;2208           SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
C 011F, 35B     ;2209           SRC/D.A,ALU/R.OR.S,DST/WRITE.B.F,CIN/CIN
                ;2210
                ;2211   WR.PLUS.LS.Q:
C 0120, 140     ;2212           SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0121, 140     ;2213           SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0122, 140     ;2214           SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0123, 140     ;2215           SRC/D.A,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
                ;2216   LS.FROM.WR.Q:
C 0124, 341     ;2217           SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0125, 341     ;2218           SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
```

```
C 0126, 341     ;2219           SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0127, 341     ;2220           SRC/D.A,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
                ;2221   WR.FROM.LS.Q:
C 0128, 342     ;2222           SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0129, 342     ;2223           SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012A, 342     ;2224           SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 012B, 342     ;2225           SRC/D.A,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
                ;2226   WR.OR.LS.Q:
C 012C, 343     ;2227           SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012D, 343     ;2228           SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012E, 343     ;2229           SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 012F, 343     ;2230           SRC/D.A,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
                ;2231
                ;2232   WR.AND.LS.Q:
C 0130, 144     ;2233           SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0131, 144     ;2234           SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0132, 144     ;2235           SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0133, 144     ;2236           SRC/D.A,ALU/R.AND.S,DST/LOAD.Q,CIN/NO.CIN
                ;2237   WR.XOR.LS.Q:
C 0134, 346     ;2238           SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0135, 346     ;2239           SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0136, 346     ;2240           SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0137, 346     ;2241           SRC/D.A,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
                ;2242   LS.CMP.WR:
C 0138, 34A     ;2243           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0139, 34A     ;2244           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013A, 34A     ;2245           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 013B, 34A     ;2246           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;2247   WR.CMP.LS:
C 013C, 349     ;2248           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013D, 349     ;2249           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013E, 349     ;2250           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 013F, 349     ;2251           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                ;2252
                ;2253   LS.PLUS.Q:
C 0140, 180     ;2254           SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0141, 180     ;2255           SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0142, 180     ;2256           SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
C 0143, 180     ;2257           SRC/D.Q,ALU/R.PLUS.S,DST/LOAD.Q,CIN/NO.CIN
                ;2258   LS.FROM.Q:
C 0144, 381     ;2259           SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0145, 381     ;2260           SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0146, 381     ;2261           SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
C 0147, 381     ;2262           SRC/D.Q,ALU/S.MINUS.R,DST/LOAD.Q,CIN/CIN
                ;2263   Q.FROM.LS.Q:
C 0148, 382     ;2264           SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 0149, 382     ;2265           SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014A, 382     ;2266           SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
C 014B, 382     ;2267           SRC/D.Q,ALU/R.MINUS.S,DST/LOAD.Q,CIN/CIN
                ;2268   LS.OR.Q:
C 014C, 383     ;2269           SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014D, 383     ;2270           SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014E, 383     ;2271           SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
C 014F, 383     ;2272           SRC/D.Q,ALU/R.OR.S,DST/LOAD.Q,CIN/CIN
                ;2273
```

```
                    ;2274   LS.MASK.Q:
C 0150, 185         ;2275           SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0151, 185         ;2276           SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0152, 185         ;2277           SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
C 0153, 185         ;2278           SRC/D.Q,ALU/NOTR.AND.S,DST/LOAD.Q,CIN/NO.CIN
                    ;2279   LS.XOR.Q:
C 0154, 386         ;2280           SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0155, 386         ;2281           SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0156, 386         ;2282           SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
C 0157, 386         ;2283           SRC/D.Q,ALU/R.XOR.S,DST/LOAD.Q,CIN/CIN
                    ;2284   LS.AND.Q:
C 0158, 384         ;2285           SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 0159, 384         ;2286           SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015A, 384         ;2287           SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
C 015B, 384         ;2288           SRC/D.Q,ALU/R.AND.S,DST/LOAD.Q,CIN/CIN
                    ;2289   LS.BIT.Q:
C 015C, 38C         ;2290           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015D, 38C         ;2291           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015E, 38C         ;2292           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
C 015F, 38C         ;2293           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/CIN
                    ;2294
                    ;2295   MOV.LS.Q:
C 0160, 1C3         ;2296           SRC/D.0,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0161, 1C3         ;2297           SRC/D.0,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0162, 1C3         ;2298           SRC/D.0,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
C 0163, 1C3         ;2299           SRC/D.0,ALU/R.OR.S,DST/LOAD.Q,CIN/NO.CIN
                    ;2300   WR.BIT.LS:
C 0164, 34C         ;2301           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0165, 34C         ;2302           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0166, 34C         ;2303           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0167, 34C         ;2304           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
                    ;2305   LS.CMP.Q:
C 0168, 38A         ;2306           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 0169, 38A         ;2307           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016A, 38A         ;2308           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 016B, 38A         ;2309           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                    ;2310   Q.CMP.LS:
C 016C, 389         ;2311           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016D, 389         ;2312           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016E, 389         ;2313           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 016F, 389         ;2314           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                    ;2315
                    ;2316   Q.PLUS.LS.TO.WR:
C 0170, 198         ;2317           SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0171, 198         ;2318           SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0172, 198         ;2319           SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
C 0173, 198         ;2320           SRC/D.Q,ALU/R.PLUS.S,DST/WRITE.B.F,CIN/NO.CIN
                    ;2321   WRA.FROM.LS.WRB:
C 0174, 35A         ;2322           SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0175, 35A         ;2323           SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0176, 35A         ;2324           SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
C 0177, 35A         ;2325           SRC/D.A,ALU/R.MINUS.S,DST/WRITE.B.F,CIN/CIN
                    ;2326   LS.NEG.WR:
C 0178, 3D9         ;2327           SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 0179, 3D9         ;2328           SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
```

```
C 017A, 3D9     ;2329           SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
C 017B, 3D9     ;2330           SRC/D.0,ALU/S.MINUS.R,DST/WRITE.B.F,CIN/CIN
                ;2331   LS.COM.WR:
C 017C, 3DF     ;2332           SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017D, 3DF     ;2333           SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017E, 3DF     ;2334           SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
C 017F, 3DF     ;2335           SRC/D.0,ALU/R.XNOR.S,DST/WRITE.B.F,CIN/CIN
```

```
                    ;2336   .PAGE
                    ;2337   ;
                    ;2338   ; THE FOLLOWING LOCATIONS HAVE THE LOCAL STORE AS THEIR DESTINATION.
                    ;2339   ;
                    ;2340   ; THIS FACT IS USED BY HARDWARE TO GENERATE LOCAL STORE WRITE  PULSES
                    ;2341
                    ;2342   180:
                    ;2343
                    ;2344   LS.PLUS.WR.XCHG:
C 0180, 150         ;2345           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0181, 150         ;2346           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0182, 150         ;2347           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 0183, 150         ;2348           SRC/D.A,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
                    ;2349   LS.FROM.WR.XCHG:
C 0184, 351         ;2350           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0185, 351         ;2351           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0186, 351         ;2352           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
C 0187, 351         ;2353           SRC/D.A,ALU/S.MINUS.R,DST/WRITE.B.A,CIN/CIN
                    ;2354   LS.AND.WR.XCHG:
C 0188, 354         ;2355           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 0189, 354         ;2356           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 018A, 354         ;2357           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
C 018B, 354         ;2358           SRC/D.A,ALU/R.AND.S,DST/WRITE.B.A,CIN/CIN
                    ;2359   LS.XOR.WR.XCHG:
C 018C, 356         ;2360           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018D, 356         ;2361           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018E, 356         ;2362           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
C 018F, 356         ;2363           SRC/D.A,ALU/R.XOR.S,DST/WRITE.B.A,CIN/CIN
                    ;2364
                    ;2365   SWAP.LS.WR:
C 0190, 1D3         ;2366           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0191, 1D3         ;2367           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0192, 1D3         ;2368           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
C 0193, 1D3         ;2369           SRC/D.0,ALU/R.OR.S,DST/WRITE.B.A,CIN/NO.CIN
                    ;2370   CLR.LS:
C 0194, 3CC         ;2371           SRC/D.0,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0195, 3CC         ;2372           SRC/D.0,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0196, 3CC         ;2373           SRC/D.0,ALU/R.AND.S,DST/NOP,CIN/CIN
C 0197, 3CC         ;2374           SRC/D.0,ALU/R.AND.S,DST/NOP,CIN/CIN
                    ;2375   MOV.Q.WR&WR.LS:
C 0198, 293         ;2376           SRC/0.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 0199, 293         ;2377           SRC/0.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 019A, 293         ;2378           SRC/0.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
C 019B, 293         ;2379           SRC/0.Q,ALU/R.OR.S,DST/WRITE.B.A,CIN/CIN
                    ;2380   WR.PLUS.Q.XCHG:
C 019C, 010         ;2381           SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019D, 010         ;2382           SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019E, 010         ;2383           SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
C 019F, 010         ;2384           SRC/A.Q,ALU/R.PLUS.S,DST/WRITE.B.A,CIN/NO.CIN
                    ;2385
                    ;2386   WR.FROM.LS-1:
C 01A0, 14A         ;2387           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A1, 14A         ;2388           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A2, 14A         ;2389           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01A3, 14A         ;2390           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
```

```
                    ;2391   LS.FROM.WR.LS:
C 01A4, 349         ;2392           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A5, 349         ;2393           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A6, 349         ;2394           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01A7, 349         ;2395           SRC/D.A,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                    ;2396   WR.PLUS.LS+1:
C 01A8, 348         ;2397           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01A9, 348         ;2398           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AA, 348         ;2399           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01AB, 348         ;2400           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/CIN
                    ;2401   WR.FROM.LS:
C 01AC, 34A         ;2402           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AD, 34A         ;2403           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AE, 34A         ;2404           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01AF, 34A         ;2405           SRC/D.A,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                    ;2406
                    ;2407   WR.PLUS.LS:
C 01B0, 148         ;2408           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B1, 148         ;2409           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B2, 148         ;2410           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01B3, 148         ;2411           SRC/D.A,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
                    ;2412   WR.OR.LS:
C 01B4, 34B         ;2413           SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B5, 34B         ;2414           SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B6, 34B         ;2415           SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01B7, 34B         ;2416           SRC/D.A,ALU/R.OR.S,DST/NOP,CIN/CIN
                    ;2417   WR.AND.LS:
C 01B8, 34C         ;2418           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01B9, 34C         ;2419           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BA, 34C         ;2420           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
C 01BB, 34C         ;2421           SRC/D.A,ALU/R.AND.S,DST/NOP,CIN/CIN
                    ;2422   WR.XOR.LS:
C 01BC, 34E         ;2423           SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BD, 34E         ;2424           SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BE, 34E         ;2425           SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01BF, 34E         ;2426           SRC/D.A,ALU/R.XOR.S,DST/NOP,CIN/CIN
                    ;2427
                    ;2428   Q.PLUS.LS:
C 01C0, 188         ;2429           SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C1, 188         ;2430           SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C2, 188         ;2431           SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01C3, 188         ;2432           SRC/D.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
                    ;2433   LS.FROM.Q.LS:
C 01C4, 389         ;2434           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C5, 389         ;2435           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C6, 389         ;2436           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01C7, 389         ;2437           SRC/D.Q,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                    ;2438   Q.FROM.LS:
C 01C8, 38A         ;2439           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01C9, 38A         ;2440           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CA, 38A         ;2441           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01CB, 38A         ;2442           SRC/D.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                    ;2443   Q.OR.LS:
C 01CC, 38B         ;2444           SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CD, 38B         ;2445           SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
```

```
C 01CE, 38B     ;2446           SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01CF, 38B     ;2447           SRC/D.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
                ;2448
                ;2449   Q.AND.LS:
C 01D0, 18C     ;2450           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D1, 18C     ;2451           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D2, 18C     ;2452           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
C 01D3, 18C     ;2453           SRC/D.Q,ALU/R.AND.S,DST/NOP,CIN/NO.CIN
                ;2454   Q.XOR.LS:
C 01D4, 38E     ;2455           SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D5, 38E     ;2456           SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D6, 38E     ;2457           SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
C 01D7, 38E     ;2458           SRC/D.Q,ALU/R.XOR.S,DST/NOP,CIN/CIN
                ;2459   MOV.Q.LS:
C 01D8, 28B     ;2460           SRC/0.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01D9, 28B     ;2461           SRC/0.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DA, 28B     ;2462           SRC/0.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
C 01DB, 28B     ;2463           SRC/0.Q,ALU/R.OR.S,DST/NOP,CIN/CIN
                ;2464   Q.NEG.LS:
C 01DC, 28A     ;2465           SRC/0.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DD, 28A     ;2466           SRC/0.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DE, 28A     ;2467           SRC/0.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01DF, 28A     ;2468           SRC/0.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;2469
                ;2470   WR.COM.LS:
C 01E0, 0CF     ;2471           SRC/0.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E1, 0CF     ;2472           SRC/0.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E2, 0CF     ;2473           SRC/0.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
C 01E3, 0CF     ;2474           SRC/0.B,ALU/R.XNOR.S,DST/NOP,CIN/NO.CIN
                ;2475   WR.NEG.LS:
C 01E4, 2CA     ;2476           SRC/0.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E5, 2CA     ;2477           SRC/0.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E6, 2CA     ;2478           SRC/0.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01E7, 2CA     ;2479           SRC/0.B,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;2480   Q.PLUS.WR.TO.LS:
C 01E8, 008     ;2481           SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01E9, 008     ;2482           SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EA, 008     ;2483           SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
C 01EB, 008     ;2484           SRC/A.Q,ALU/R.PLUS.S,DST/NOP,CIN/NO.CIN
                ;2485   Q.FROM.WR.TO.LS:
C 01EC, 20A     ;2486           SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01ED, 20A     ;2487           SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EE, 20A     ;2488           SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
C 01EF, 20A     ;2489           SRC/A.Q,ALU/R.MINUS.S,DST/NOP,CIN/CIN
                ;2490
                ;2491   DEC.LS:
C 01F0, 1CA     ;2492           SRC/D.0,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F1, 1CA     ;2493           SRC/D.0,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F2, 1CA     ;2494           SRC/D.0,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
C 01F3, 1CA     ;2495           SRC/D.0,ALU/R.MINUS.S,DST/NOP,CIN/NO.CIN
                ;2496   COM.LS:
C 01F4, 3CF     ;2497           SRC/D.0,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F5, 3CF     ;2498           SRC/D.0,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F6, 3CF     ;2499           SRC/D.0,ALU/R.XNOR.S,DST/NOP,CIN/CIN
C 01F7, 3CF     ;2500           SRC/D.0,ALU/R.XNOR.S,DST/NOP,CIN/CIN
```

```
                    ;2501   NEG.LS:
C 01F8, 3C9         ;2502           SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01F9, 3C9         ;2503           SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01FA, 3C9         ;2504           SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
C 01FB, 3C9         ;2505           SRC/D.0,ALU/S.MINUS.R,DST/NOP,CIN/CIN
                    ;2506   INC.LS:
C 01FC, 3C8         ;2507           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FD, 3C8         ;2508           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FE, 3C8         ;2509           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
C 01FF, 3C8         ;2510           SRC/D.0,ALU/R.PLUS.S,DST/NOP,CIN/CIN
                    ;2511
                    ;2512   .UCODE
```

```
;2513   .PAGE   "DIAGNOSTIC MACROS AND DEFINITIONS"
;2514
;2515   D.ADRS/=<15:9>,.VALIDITY=<D.VAL>,.DEFAULT=0
;2516
;2517
;2518           SAV.WR0=1               ; STORAGE FOR WR0
;2519           SAV.WR1=2               ; STORAGE FOR WR1
;2520           SAV.WR2=3               ; STORAGE FOR WR2
;2521           SAV.WR3=4               ; STORAGE FOR WR3
;2522           T5.SUB=5                ; RESERVED FOR COMMON SUBROUTINES
;2523           T6.SUB=6                ; RESERVED FOR COMMON SUBROUTINES
;2524           T7.SUB=7                ; RESERVED FOR COMMON SUBROUTINES
;2525           TRANSFER.POINTER=7      ; POINTER FOR TRANSFER ROUTINE
;2526           MEMORY.SIZE=7           ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
;2527                                   ;  SIZING
;2528           FPA.FOUND=7             ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
;2529           UBE.FOUND=7             ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
;2530           T8=8                    ; TEMP LOCATION 8
;2531           T9=9                    ; TEMP LOCATION 9
;2532           T10=0A                  ; TEMP LOCATION 10
;2533           T11=0B                  ; TEMP LOCATION 11
;2534           T12=0C                  ; TEMP LOCATION 12
;2535           T13=0D                  ; TEMP LOCATION 13
;2536           T14=0E                  ; TEMP LOCATION 14
;2537           T15=0F                  ; TEMP LOCATION 15
;2538           PC=10                   ; MACRO PROGRAM COUNTER
;2539
;2540           WR.BACKUP=11            ; BACKUP WR FOR MOV MEM.DATA TO WR[]
;2541           MM.LASTADDR+1=12        ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
;2542                                   ;  1 (FIRST NON-EXISTANT MEMORY ADDRESS)
;2543           #FF=13                  ; MASK
;2544           #FFFF=14                ; MASK
;2545           #FF000000=15            ; MASK
;2546           #FFFFFF00=16            ; MASK
;2547           CSR0.MASK=16            ; MASK
;2548           CSR1.MASK=17            ; MASK (01003FFF)
;2549           CSR2.MASK=18            ; MASK (7FFE3FFF)
;2550           #FE7FFFFF=19            ; MASK
;2551           UBS.SET=1A              ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
;2552           UBS.DATA=1B             ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
;2553                                   ; DATA TEST
;2554
;2555           ERR.CON.NA=1E           ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
;2556                                   ; LOADING CONTROL AND STATUS REG IN INITIAL
;2557                                   ; TESTS
;2558           ERR.CON=1F              ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
;2559                                   ; LOADING CONTROL AND STATUS REG IN INITIAL
;2560                                   ; TESTS
;2561
;2562           #1=20                   ; PATTERN 00000001 (HEX)
;2563           #2=21                   ; PATTERN 00000002
;2564           #4=22                   ; PATTERN 00000004
;2565           #8=23                   ; PATTERN 00000008
;2566           #10=24                  ; PATTERN 00000010
;2567           #20=25                  ; PATTERN 00000020
```

```
;2568            #40=26                  ; PATTERN 00000040
;2569            #80=27                  ; PATTERN 00000080
;2570            #100=28                 ; PATTERN 00000100
;2571            #200=29                 ; PATTERN 00000200
;2572            #400=2A                 ; PATTERN 00000400
;2573            #800=2B                 ; PATTERN 00000800
;2574            #1000=2C                ; PATTERN 00001000
;2575            #2000=2D                ; PATTERN 00002000
;2576            #4000=2E                ; PATTERN 00004000
;2577            #8000=2F                ; PATTERN 00008000
;2578            #10000=30               ; PATTERN 00010000
;2579            #20000=31               ; PATTERN 00020000
;2580            #40000=32               ; PATTERN 00040000
;2581            #80000=33               ; PATTERN 00080000
;2582            #100000=34              ; PATTERN 00100000
;2583            #200000=35              ; PATTERN 00200000
;2584            #400000=36              ; PATTERN 00400000
;2585            #800000=37              ; PATTERN 00800000
;2586            #1000000=38             ; PATTERN 01000000
;2587            #2000000=39             ; PATTERN 02000000
;2588            #4000000=3A             ; PATTERN 04000000
;2589            #8000000=3B             ; PATTERN 08000000
;2590            #10000000=3C            ; PATTERN 10000000
;2591            #20000000=3D            ; PATTERN 20000000
;2592            #40000000=3E            ; PATTERN 40000000
;2593            #80000000=3F            ; PATTERN 80000000
;2594
;2595            BIT0=20                 ; PATTERN 00000001
;2596            BIT1=21                 ; PATTERN 00000002
;2597            BIT2=22                 ; PATTERN 00000004
;2598            BIT3=23                 ; PATTERN 00000008
;2599            BIT4=24                 ; PATTERN 00000010
;2600            BIT5=25                 ; PATTERN 00000020
;2601            BIT6=26                 ; PATTERN 00000040
;2602            BIT7=27                 ; PATTERN 00000080
;2603            BIT8=28                 ; PATTERN 00000100
;2604            BIT9=29                 ; PATTERN 00000200
;2605            BIT10=2A                ; PATTERN 00000400
;2606            BIT11=2B                ; PATTERN 00000800
;2607            BIT12=2C                ; PATTERN 00001000
;2608            BIT13=2D                ; PATTERN 00002000
;2609            BIT14=2E                ; PATTERN 00004000
;2610            BIT15=2F                ; PATTERN 00008000
;2611            BIT16=30                ; PATTERN 00010000
;2612            BIT17=31                ; PATTERN 00020000
;2613            BIT18=32                ; PATTERN 00040000
;2614            BIT19=33                ; PATTERN 00080000
;2615            BIT20=34                ; PATTERN 00100000
;2616            BIT21=35                ; PATTERN 00200000
;2617            BIT22=36                ; PATTERN 00400000
;2618            BIT23=37                ; PATTERN 00800000
;2619            BIT24=38                ; PATTERN 01000000
;2620            BIT25=39                ; PATTERN 02000000
;2621            BIT26=3A                ; PATTERN 04000000
;2622            BIT27=3B                ; PATTERN 08000000
```

```
;2623            BIT28=3C                ; PATTERN 10000000
;2624            BIT29=3D                ; PATTERN 20000000
;2625            BIT30=3E                ; PATTERN 40000000
;2626            BIT31=3F                ; PATTERN 80000000
;2627
;2628
;2629    ;CSR ADDRESS MNEMONICS
;2630
;2631            CSR0=4E                 ; ALL BITS CLEAR TO ACCESS CSR 0
;2632            CSR1=22                 ; BIT 2 SET TO ACCESS CSR 1
;2633            CSR2=23                 ; BIT 3 SET TO ACCESS CSR 2
;2634
;2635    ;CONTROL AND STATUS WORD BITS
;2636
;2637            PRINT.UBE=26            ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
;2638                                    ;  (INDICATOR IN LS 7)
;2639            PRINT.R80=27            ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
;2640                                    ; WHICH DRIVE (BIT 7).  INDICATOR IN LS 7,
;2641                                    ; DRIVE NUMBER IN LS 6.
;2642            ERROR=28                ; TEST ERROR INDICATION (BIT 8)
;2643            SET.PA.ERR=29           ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
;2644                                    ;  ADDRESS POINTED TO BY LS 7 (BIT 9)
;2645            CONSOLE.LOE=2A          ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
;2646                                    ;  (FOR INITIAL TESTS)
;2647            PRINT.MEM.SIZE=2B       ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
;2648                                    ;  (SIZE IN LS 7)
;2649            PRINT.FPA=2C            ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
;2650                                    ;  (INDICATOR IN LS 7)
;2651            XFER.DATA=2D            ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
;2652            EOS=2E                  ; END OF SECTION (BIT 14)
;2653            BEGIN.TEST=2F           ; BEGINNING OF TEST (BIT 15)
;2654            INTERUPT.EN=30          ; ENABLE INTERUPT OR SIGNAL SPECIFIED BY BITS
;2655                                    ;  17 THRU 22 AND 28 THRU 30 (BIT 16)
;2656
;2657             CONSOLE.HALT=31        ;   CONSOLE HALT INTERRUPT (BIT 17)
;2658             PWR.FAIL=32            ;   PWR FAIL INT (BIT 18)
;2659             INTERVAL.TIM=33        ;   INTRVL TIM INT (BIT 19)
;2660             CONSOLE.ATTN=34        ;   CONSOLE ATTN INTERRUPT (BIT 20)
;2661             CONSOLE.ACK=35         ;   CONSOLE ACK INTERRUPT (BIT 21)
;2662             DISABLE.MEM.REF=36     ;   DISABLE MEMORY REFERANCES (BIT 22)
;2663             CINIT=3C               ;   UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
;2664             UBS.DCLO=3D            ;   UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
;2665             UBS.BBSY=3E            ;   UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
;2666
;2667            NA.EXP.REC=37           ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
;2668            OTHER.DATA=38           ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
;2669            CONSOLE.LOOP=39         ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
;2670                                    ; COUNT IN LS (BIT 25)
;2671            PRINT.CRD=3A            ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
;2672            CLR.PA.ERR=3B           ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
;2673                                    ;  ADDRESS POINTED TO BY LS 7 (BIT 27)
;2674            PRINT.IDC=3F            ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
;2675                                    ;  (INDICATOR IN LS 7)
;2676
;2677    ;MODULE CODES
```

```
;2678
;2679            WCS=20                  ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
;2680            CPU=21                  ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
;2681            MCT=22                  ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
;2682            FPA=23                  ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
;2683            ARRAY=24                ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
;2684            IDC=25                  ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
;2685
;2686    ;CSR 1 ERROR BITS
;2687
;2688            VALID.ERR=2E            ; VALID NOT SET IF 1 (BIT 14)
;2689            TB.PAR.ERR=2F           ; TB PARITY ERROR (BIT 15)
;2690            NXM=30                  ; NON-EXISTANT MEMORY ERROR (BIT 16)
;2691            UB.BSY=31               ; UNIBUS BUSY (BIT 17)
;2692            ADP.REG=32              ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
;2693            WR.ACROSS.PG=33         ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
;2694            OP.ERR=34               ; OPERATION ERROR (BIT 20)
;2695            TB.MISS=35              ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
;2696                                    ;   BUFFER) (BIT 21)
;2697            ACCESS.REFUSED=36       ; PROTECTION ERROR ON ACCESS (BIT 22)
;2698            MODIFY.REFUSED=37       ; PROTECTION ERROR ON WRITE (BIT 23)
;2699
;2700            CRD=3E                  ; SINGLE BIT ERROR CORRECTED (BIT 30)
;2701            RDS=3F                  ; UNCORRECTABLE DATA ERROR (BIT 31)
;2702
;2703
;2704    ;CSR 1 CONTROL BITS
;2705
;2706            ECC.DIS=39              ; CSR1 ECC DISABLE BIT (BIT 25)
;2707            DIAG.CHK=3A             ; CSR1 DIAG CHK BIT (BIT 26)
;2708            MME=3B                  ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
;2709            INH.CRD=3C              ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
;2710            TB.PAR.DIAG=3D          ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
;2711                                    ;  (BIT 29)
;2712
;2713
;2714    ;UBE CONTROL REGISTER BITS
;2715
;2716      ;CONTROL REG 1
;2717            GO=20                   ; START UBE (BIT 0)
;2718            BR4=21                  ; ISSUE BUS REQUEST 4 (BIT 1)
;2719            BR5=22                  ; ISSUE BUS REQUEST 5 (BIT 2)
;2720            BR6=23                  ; ISSUE BUS REQUEST 6 (BIT 3)
;2721            BR7=24                  ; ISSUE BUS REQUEST 7 (BIT 4)
;2722            NPR=25                  ; ISSUE NPR (BIT 5)
;2723            INT.ODNE=26             ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
;2724            RDY=27                  ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
;2725            C00=28                  ; C0 BIT (BIT 8)
;2726            C01=29                  ; C1 BIT (BIT 9)
;2727            FUNA=2A                 ; FUNCTION BIT A (BIT 10
;2728            FUNB=2B                 ; FUNCTION BIT B (BIT 11)
;2729            CC+DAT=2C               ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
;2730            INH.DATIP.ROL=2D        ; INHIBIT ROL ON DATIP (BIT 13)
;2731            DATOB.ON.DATIP=2E       ; DO DATOB AFTER DATIP CYCLE (BIT 14)
;2732            ERR=2F                  ; EXERCISOR OR UNIBUS ERROR (BIT 15)
```

```
;2733
;2734      ;CONTROL REG 2
;2735            EXT.MEM0=20             ; EXTENDED MEMORY BIT 0 (BIT 0)
;2736            EXT.MEM1=21             ; EXTENDED MEMORY BIT 1 (BIT 1)
;2737            INH.INC.ADDR&CC=22      ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
;2738            INH.SACK=23             ; INHIBIT ASSERING BUS SACK ON BUS GRANT (BIT 3)
;2739            PWR.DWN.SEQ=24          ; ASSERT ACLO (BIT 4)
;2740            WNG.GNT.BCK=25          ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
;2741            MAX.LATE.ERR=26         ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
;2742            NO.SACK.ERR=27          ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
;2743            NO.SSYN.ERR=28          ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
;2744            WRG.A.LINES=29          ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
;2745            NO.GNT+NOT.ONE.GNT=2A   ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
;2746            INTR.SSYN.ERR=2B        ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
;2747            ENB.PB=2C               ; ENABLE PARITY BIT B (BIT 12)
;2748            CCOVF=2D                ; CYCLE COUNT OVERFLOW (BIT 13)
;2749            TM.DLY=2E               ; REQUEST BUS EVERY 10 USEC (BIT 14)
;2750
;2751      ;BG6 MNEMONIC
;2752            BG6=23                  ; BIT 3 SET FOR BG 6 ADDRESS
;2753
;2754      ;ERROR AND CONTROL INFORMATION
;2755
;2756            CONTROL.STATUS=40       ; CONTROL AND STATUS WORD
;2757            ERROR.NUMBER=41         ; ERROR NUMBER OF CURRENT TEST
;2758            EXPECTED.DATA=42        ; EXPECTED RESULT OF CURRENT TEST
;2759            RECEIVED.DATA=43        ; ACTUAL RESULT OF CURRENT TEST
;2760            ADDRESS.DATA=44         ; OTHER PERTINENT DATA
;2761            ERROR.MASK=45           ; BITS TO CHECK IN RESULT
;2762            MODULE.NUM=46           ; STORAGE OF MODULE NUMBER (CODED)
;2763            LOOP.CNT=47             ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
;2764                                    ; CONTROL.
;2765            ERROR.CONTROL=48        ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
;2766             LOE=20                 ;  LOOP ON ERROR IF SET (BIT0)
;2767             NER=21                 ;  NO ERROR REPORT IF SET (BIT1)
;2768             BELL=22                ;  BELL ON ERROR IF SET (BIT2)
;2769             HOE=23                 ;  HALT ON ERROR IF SET (BIT3)
;2770             SER=24                 ;  ENABLE ERROR REPORT ON CRD ERRORS IF SET
;2771             APT.PRESENT=25         ;  APT PRESENT IF SET (BIT5)
;2772             LOOP.COMMAND=26        ;  LOOP COMMAND TYPED IF SET (BIT 6)
;2773
;2774            APT.PARAM=49            ;APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
;2775                                    ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
;2776                                    ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
;2777            APT.RESERVED=4A         ;RESERVED FOR FUTURE APT USE
;2778
;2779      ;MISCELLANEOUS CONSTANTS AND STORAGE
;2780
;2781            #AAAAAAAA=4C            ; CONSTANT
;2782            ALTER.ODD=4C            ; CONSTANT
;2783            #55555555=4D            ; CONSTANT
;2784            ALTER.EVEN=4D           ; CONSTANT
;2785            #0=4E                   ; CONSTANT
;2786            ZERO=4E                 ; CONSTANT
;2787            #FFFFFFFF=4F            ; CONSTANT
```

```
;2788            ONES=4F                 ; CONSTANT
;2789            PREVIOUS.ERROR=50       ; ERROR NUMBER OF LAST ERROR
;2790
;2791            DATA.1=51               ; STORAGE FOR FPA DATA
;2792            DATA.2=52               ; STORAGE FOR FPA DATA
;2793            DATA.3=53               ; STORAGE FOR FPA DATA
;2794            FIRST.FLT=54            ; STORAGE FOR FPA DATA
;2795            SEC.FLT=55              ; STORAGE FOR FPA DATA
;2796            THIRD.FLT=56            ; STORAGE FOR FPA DATA
;2797            T57=57                  ; TEMP LOCATION 57
;2798            T58=58                  ; TEMP LOCATION 58
;2799            T59=59                  ; TEMP LOCATION 59
;2800
;2801            BEDB=5A                 ;UBE (BUS EXERCISOR) DATA BUF REG
;2802            BECC=5B                 ;UBE (BUS EXERCISOR) CYCLE COUNT REG
;2803            BEBA=5C                 ;UBE (BUS EXERCISOR) ADDR REG
;2804            BECR1=5D                ;UBE (BUS EXERCISOR) CONTROL REG 1
;2805            CLEAR.ERR.ADDR=5E       ;UBE CLEAR ERROR REG ADDRESS
;2806            BECR2=5F                ;UBE (BUS EXERCISOR) CONTROL REG 2
;2807
;2808            #3(H)=60                ; CONSTANT
;2809            #12(H)=61               ; CONSTANT
;2810            #33333333=62            ; CONSTANT
;2811            #0F0F0F0F=63            ; CONSTANT
;2812            #00FF00FF=64            ; CONSTANT
;2813            #162(H)=65              ; CONSTANT FOR LS TRANSFER POINTER
;2814            BR7.IDENT=66            ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
;2815            BG7=67                  ; BG7 ADDRESS (BITS 2 AND 3 SET)
;2816
;2817    ;TEMPS FOR CPU TESTS
;2818            L30=30                  ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
;2819            L31=31                  ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
;2820            L53=53                  ;LS 53 TEMP
;2821            L73=73                  ;LS 73 TEMP
;2822
;2823    ;REGISTER ADDRESSES
;2824
;2825            CONTROL.OS=78           ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
;2826            SHIFT.OS(3-0)=79        ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;2827                                    ;  (LS 20-2F)
;2828            SHIFT.OS(4-0)=7B        ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;2829                                    ;  (LS 20-3F)
;2830            OS=7C                   ; OS REGISTER
;2831            DEC.CON=7D              ; DECIMAL CARRIES
;2832            SIZE=7E                 ; SIZE REGISTER
;2833            MDT= 7E                 ; MEMORY REFERENCE DATA SIZE
;2834            INTERRUPT.VEC=7E        ; HARDWARE INTERRUPT VECTOR
;2835    ;
;2836    ;       THIS WORD IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE READ
;2837    ;       OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;2838    ;
;2839            PSL.CC=7F               ; PSL CONDITION CODES
```

```
;2840   .PAGE
;2841   XD.ADRS/=<16:9>,.VALIDITY=<XD.VAL>
;2842
;2843
;2844           SAV.WR0=1                ; STORAGE FOR WR0
;2845           SAV.WR1=2                ; STORAGE FOR WR1
;2846           SAV.WR2=3                ; STORAGE FOR WR2
;2847           SAV.WR3=4                ; STORAGE FOR WR3
;2848           T5.SUB=5                 ; RESERVED FOR COMMON SUBROUTINES
;2849           T6.SUB=6                 ; RESERVED FOR COMMON SUBROUTINES
;2850           T7.SUB=7                 ; RESERVED FOR COMMON SUBROUTINES
;2851           TRANSFER.POINTER=7       ; POINTER FOR TRANSFER ROUTINE
;2852           MEMORY.SIZE=7            ; STORAGE FOR MEMORY SIZE IN 256KB BLOCKS AFTER
;2853                                    ;  SIZING
;2854           FPA.FOUND=7              ; STORAGE FOR RESULT OF SIZING FOR FPA PRESENT
;2855           UBE.FOUND=7              ; STORAGE FOR RESULT OF SIZING FOR UBE PRESENT
;2856           T8=8                     ; TEMP LOCATION 8
;2857           T9=9                     ; TEMP LOCATION 9
;2858           T10=0A                   ; TEMP LOCATION 10
;2859           T11=0B                   ; TEMP LOCATION 11
;2860           T12=0C                   ; TEMP LOCATION 12
;2861           T13=0D                   ; TEMP LOCATION 13
;2862           T14=0E                   ; TEMP LOCATION 14
;2863           T15=0F                   ; TEMP LOCATION 15
;2864           PC=10                    ; MACRO PROGRAM COUNTER
;2865
;2866           WR.BACKUP=11             ; BACKUP WR FOR MOV MEM.DATA TO WR[]
;2867           MM.LASTADDR+1=12         ; STORAGE OF LAST ADDRESS IN MAIN MEMORY PLUS
;2868                                    ;  1 (FIRST NON-EXISTANT MEMORY ADDRESS)
;2869           #FF=13                   ; MASK
;2870           #FFFF=14                 ; MASK
;2871           #FF000000=15             ; MASK
;2872           #FFFFFF00=16             ; MASK
;2873           CSR0.MASK=16             ; MASK
;2874           CSR1.MASK=17             ; MASK (01003FFF)
;2875           CSR2.MASK=18             ; MASK (FFFE3FFF)
;2876           #FE7FFFFF=19             ; MASK
;2877           UBS.SET=1A               ; CONSTANT (7FF80000) USED BY UBS ADDRESS TEST
;2878           UBS.DATA=1B              ; MASK (7FFF8000) USED AS ERROR MASK FOR UBS
;2879                                    ; DATA TEST
;2880
;2881           ERR.CON.NA=1E            ; CONSTANT OF 800500(H) BITS 23,10,8 SET FOR
;2882                                    ; LOADING CONTROL AND STATUS REG IN INITIAL
;2883                                    ; TESTS
;2884           ERR.CON=1F               ; CONSTANT OF 500(H) BITS 10 & 8 SET FOR
;2885                                    ; LOADING CONTROL AND STATUS REG IN INITIAL
;2886                                    ; TESTS
;2887
;2888           #1=20                    ; PATTERN 00000001 (HEX)
;2889           #2=21                    ; PATTERN 00000002
;2890           #4=22                    ; PATTERN 00000004
;2891           #8=23                    ; PATTERN 00000008
;2892           #10=24                   ; PATTERN 00000010
;2893           #20=25                   ; PATTERN 00000020
;2894           #40=26                   ; PATTERN 00000040
```

```
;2895            #80=27                  ; PATTERN 00000080
;2896            #100=28                 ; PATTERN 00000100
;2897            #200=29                 ; PATTERN 00000200
;2898            #400=2A                 ; PATTERN 00000400
;2899            #800=2B                 ; PATTERN 00000800
;2900            #1000=2C                ; PATTERN 00001000
;2901            #2000=2D                ; PATTERN 00002000
;2902            #4000=2E                ; PATTERN 00004000
;2903            #8000=2F                ; PATTERN 00008000
;2904            #10000=30               ; PATTERN 00010000
;2905            #20000=31               ; PATTERN 00020000
;2906            #40000=32               ; PATTERN 00040000
;2907            #80000=33               ; PATTERN 00080000
;2908            #100000=34              ; PATTERN 00100000
;2909            #200000=35              ; PATTERN 00200000
;2910            #400000=36              ; PATTERN 00400000
;2911            #800000=37              ; PATTERN 00800000
;2912            #1000000=38             ; PATTERN 01000000
;2913            #2000000=39             ; PATTERN 02000000
;2914            #4000000=3A             ; PATTERN 04000000
;2915            #8000000=3B             ; PATTERN 08000000
;2916            #10000000=3C            ; PATTERN 10000000
;2917            #20000000=3D            ; PATTERN 20000000
;2918            #40000000=3E            ; PATTERN 40000000
;2919            #80000000=3F            ; PATTERN 80000000
;2920
;2921            BIT0=20                 ; PATTERN 00000001
;2922            BIT1=21                 ; PATTERN 00000002
;2923            BIT2=22                 ; PATTERN 00000004
;2924            BIT3=23                 ; PATTERN 00000008
;2925            BIT4=24                 ; PATTERN 00000010
;2926            BIT5=25                 ; PATTERN 00000020
;2927            BIT6=26                 ; PATTERN 00000040
;2928            BIT7=27                 ; PATTERN 00000080
;2929            BIT8=28                 ; PATTERN 00000100
;2930            BIT9=29                 ; PATTERN 00000200
;2931            BIT10=2A                ; PATTERN 00000400
;2932            BIT11=2B                ; PATTERN 00000800
;2933            BIT12=2C                ; PATTERN 00001000
;2934            BIT13=2D                ; PATTERN 00002000
;2935            BIT14=2E                ; PATTERN 00004000
;2936            BIT15=2F                ; PATTERN 00008000
;2937            BIT16=30                ; PATTERN 00010000
;2938            BIT17=31                ; PATTERN 00020000
;2939            BIT18=32                ; PATTERN 00040000
;2940            BIT19=33                ; PATTERN 00080000
;2941            BIT20=34                ; PATTERN 00100000
;2942            BIT21=35                ; PATTERN 00200000
;2943            BIT22=36                ; PATTERN 00400000
;2944            BIT23=37                ; PATTERN 00800000
;2945            BIT24=38                ; PATTERN 01000000
;2946            BIT25=39                ; PATTERN 02000000
;2947            BIT26=3A                ; PATTERN 04000000
;2948            BIT27=3B                ; PATTERN 08000000
;2949            BIT28=3C                ; PATTERN 10000000
```

```
;2950            BIT29=3D                ; PATTERN 20000000
;2951            BIT30=3E                ; PATTERN 40000000
;2952            BIT31=3F                ; PATTERN 80000000
;2953
;2954
;2955    ;CSR ADDRESS MNEMONICS
;2956
;2957            CSR0=4E                 ; ALL BITS CLEAR TO ACCESS CSR 0
;2958            CSR1=22                 ; BIT 2 SET TO ACCESS CSR 1
;2959            CSR2=23                 ; BIT 3 SET TO ACCESS CSR 2
;2960
;2961
;2962    ;CONTROL AND STATUS WORD BITS
;2963
;2964            PRINT.UBE=26            ; PRINT IF UBE PRESENT OR NOT PRESENT (BIT 6)
;2965                                    ;  (INDICATOR IN LS 7)
;2966            PRINT.R80=27            ; PRINT IF R80 PRESENT OR NOT PRESENT AND ON
;2967                                    ; WHICH DRIVE (BIT 7).  INDICATOR IN LS 7,
;2968                                    ; DRIVE NUMBER IN LS 6.
;2969            ERROR=28                ; TEST ERROR INDICATION (BIT 8)
;2970            SET.PA.ERR=29           ; TELLS CONSOLE TO CREATE A PARITY ERROR AT WCS
;2971                                    ;  ADDRESS POINTED TO BY LS 7 (BIT 9)
;2972            CONSOLE.LOE=2A          ; INDICATES CONSOLE DOES ERROR LOOPING (BIT 10)
;2973                                    ;  (FOR INITIAL TESTS)
;2974            PRINT.MEM.SIZE=2B       ; PRINT MEMORY SIZE IN 256K BLOCKS (BIT 11)
;2975                                    ;  (SIZE IN LS 7)
;2976            PRINT.FPA=2C            ; PRINT IF FPA PRESENT OR NOT PRESENT (BIT 12)
;2977                                    ;  (INDICATOR IN LS 7)
;2978            XFER.DATA=2D            ; INDICATES A DATA TRANSFER TO LS OR MM (BIT 13)
;2979            EOS=2E                  ; END OF SECTION (BIT 14)
;2980            BEGIN.TEST=2F           ; BEGINNING OF TEST (BIT 15)
;2981            INTERUPT.EN=30          ; ENABLE INTERUPT OR SIGNAL SPECIFIED BY BITS
;2982                                    ;  17 THRU 22 AND 28 THRU 30 (BIT 16)
;2983
;2984             CONSOLE.HALT=31        ;   CONSOLE HALT INTERRUPT (BIT 17)
;2985             PWR.FAIL=32            ;   PWR FAIL INT (BIT 18)
;2986             INTERVAL.TIM=33        ;   INTRVL TIM INT (BIT 19)
;2987             CONSOLE.ATTN=34        ;   CONSOLE ATTN INTERRUPT (BIT 20)
;2988             CONSOLE.ACK=35         ;   CONSOLE ACK INTERRUPT (BIT 21
;2989             DISABLE.MEM.REF=36     ;   DISABLE MEMORY REFERANCES (BIT 22)
;2990             CINIT=3C               ;   UNIBUS INIT SIGNAL TO THE MCT (BIT 28)
;2991             UBS.DCLO=3D            ;   UNIBUS DC LO INDICATION TO THE MCT (BIT 29)
;2992             UBS.BBSY=3E            ;   UNIBUS BUS BUSY INDICATOR TO MCT (BIT 30)
;2993
;2994            NA.EXP.REC=37           ; PRINT N/A UNDER EXPECTED AND RECEIVED (BIT 23)
;2995            OTHER.DATA=38           ; PRINT A VALUE UNDER OTHER DATA (BIT 24)
;2996            CONSOLE.LOOP=39         ; CONSOLE PERFORMS A LOOP ACCORDING TO LOOP
;2997                                    ; COUNT IN LS (BIT 25)
;2998            PRINT.CRD=3A            ; PRINT NUMBER OF SINGLE BIT ERRORS (BIT 26)
;2999            CLR.PA.ERR=3B           ; TELLS CONSOLE TO CLEAR PARITY ERROR AT WCS
;3000                                    ;  ADDRESS POINTED TO BY LS 7 (BIT 27)
;3001            PRINT.IDC=3F            ; PRINT IF IDC PRESENT OR NOT PRESENT (BIT 31)
;3002                                    ;  (INDICATOR IN LS 7)
;3003
;3004
```

```
;3005    ;MODULE CODES
;3006
;3007            WCS=20                  ; MODULE CODE FOR WCS BOARD (BIT 0 SET)
;3008            CPU=21                  ; MODULE CODE FOR CPU BOARD (BIT 1 SET)
;3009            MCT=22                  ; MODULE CODE FOR MCT BOARD (BIT 2 SET)
;3010            FPA=23                  ; MODULE CODE FOR FPA BOARD (BIT 3 SET)
;3011            ARRAY=24                ; MODULE CODE FOR ARRAY BOARD (BIT 4 SET)
;3012            IDC=25                  ; MODULE CODE FOR IDC BOARD (BIT 5 SET)
;3013
;3014    ;CSR 1 ERROR BITS
;3015
;3016            VALID.ERR=2E            ; VALID NOT SET IF 1 (BIT 14)
;3017            TB.PAR.ERR=2F           ; TB PARITY ERROR (BIT 15)
;3018            NXM=30                  ; NON-EXISTANT MEMORY ERROR (BIT 16)
;3019            UB.BSY=31               ; UNIBUS BUSY (BIT 17)
;3020            ADP.REG=32              ; UNIBUS ADAPTER REGISTER SELECT (BIT 18)
;3021            WR.ACROSS.PG=33         ; WRITE ACROSS PAGE BOUNDARY ERROR (BIT 19)
;3022            OP.ERR=34               ; OPERATION ERROR (BIT 20)
;3023            TB.MISS=35              ; TB MISS (VIRTUAL ADDRESS TRANSLATION NOT IN
;3024                                    ;   BUFFER) (BIT 21)
;3025            ACCESS.REFUSED=36       ; PROTECTION ERROR ON ACCESS (BIT 22)
;3026            MODIFY.REFUSED=37       ; PROTECTION ERROR ON WRITE (BIT 23)
;3027
;3028            CRD=3E                  ; SINGLE BIT ERROR CORRECTED (BIT 30)
;3029            RDS=3F                  ; UNCORRECTABLE DATA ERROR (BIT 31)
;3030
;3031
;3032    ;CSR 1 CONTROL BITS
;3033
;3034            ECC.DIS=39              ; CSR1 ECC DISABLE BIT (BIT 25)
;3035            DIAG.CHK=3A             ; CSR1 DIAG CHK BIT (BIT 26)
;3036            MME=3B                  ; CSR1 MEMORY MANAGMENT ENABLE BIT (BIT 27)
;3037            INH.CRD=3C              ; CSR1 INHIBIT REPORT CRD BIT (BIT 28)
;3038            TB.PAR.DIAG=3D          ; CSR1 TB PAR DIAG BIT (FORCE TB PARITY ERR)
;3039                                    ;  (BIT 29)
;3040
;3041
;3042    ;UBE CONTROL REGISTER BITS
;3043
;3044      ;CONTROL REG 1
;3045            GO=20                   ; START UBE (BIT 0)
;3046            BR4=21                  ; ISSUE BUS REQUEST 4 (BIT 1)
;3047            BR5=22                  ; ISSUE BUS REQUEST 5 (BIT 2)
;3048            BR6=23                  ; ISSUE BUS REQUEST 6 (BIT 3)
;3049            BR7=24                  ; ISSUE BUS REQUEST 7 (BIT 4)
;3050            NPR=25                  ; ISSUE NPR (BIT 5)
;3051            INT.ODNE=26             ; INTERRUPT ON DONE (WHEN CCOVF) (BIT 6)
;3052            RDY=27                  ; READY BIT, SET WHEN READY TO BEGIN FUNC (BIT 7)
;3053            C00=28                  ; C0 BIT (BIT 8)
;3054            C01=29                  ; C1 BIT (BIT 9)
;3055            FUNA=2A                 ; FUNCTION BIT A (BIT 10
;3056            FUNB=2B                 ; FUNCTION BIT B (BIT 11)
;3057            CC+DAT=2C               ; DATA FROM CYCLE COUNT IF SET, DATA REG IF CLEAR
;3058            INH.DATIP.ROL=2D        ; INHIBIT ROL ON DATIP (BIT 13)
;3059            DATOB.ON.DATIP=2E       ; DO DATOB AFTER DATIP CYCLE (BIT 14)
```

```
;3060            ERR=2F                      ; EXERCISOR OR UNIBUS ERROR (BIT 15)
;3061
;3062          ;CONTROL REG 2
;3063            EXT.MEM0=20                 ; EXTENDED MEMORY BIT 0 (BIT 0)
;3064            EXT.MEM1=21                 ; EXTENDED MEMORY BIT 1 (BIT 1)
;3065            INH.INC.ADDR&CC=22          ; INHIBIT INC ON CYCLE COUNT AND ADDR REG (BIT 2)
;3066            INH.SACK=23                 ; INHIBIT ASSERING BUS SACK ON BUS GRANT (BIT 3)
;3067            PWR.DWN.SEQ=24              ; ASSERT ACLO (BIT 4)
;3068            WNG.GNT.BCK=25              ; NO BUS GRANT RECV TO HIGEST REQUEST (BIT 5)
;3069            MAX.LATE.ERR=26             ; NPR AND BR LATENCY TIME EXCEEDED (BIT 6)
;3070            NO.SACK.ERR=27              ; NO TIMEOUT AT CPU WHEN NO SACK (BIT 7)
;3071            NO.SSYN.ERR=28              ; NO SSYN RECEIVED AFTER ASSERTING MSYN (BIT 8)
;3072            WRG.A.LINES=29              ; ADDRESS ERR ON SENT & RECEIVED ADDRESS (BIT 9)
;3073            NO.GNT+NOT.ONE.GNT=2A       ; NO GRANT OR MORE THAN 1 GRANT AFTER REQ. (BIT 10)
;3074            INTR.SSYN.ERR=2B            ; NO SSYN RECEIVED AFTER BUS INTR (BIT 11)
;3075            ENB.PB=2C                   ; ENABLE PARITY BIT B (BIT 12)
;3076            CCOVF=2D                    ; CYCLE COUNT OVERFLOW (BIT 13)
;3077            TM.DLY=2E                   ; REQUEST BUS EVERY 10 USEC (BIT 14)
;3078
;3079          ;BG6 MNEMONIC
;3080            BG6=23                      ; BIT 3 SET FOR BG 6 ADDRESS
;3081
;3082          ;ERROR AND CONTROL INFORMATION
;3083
;3084            CONTROL.STATUS=40           ; CONTROL AND STATUS WORD
;3085            ERROR.NUMBER=41             ; ERROR NUMBER OF CURRENT TEST
;3086            EXPECTED.DATA=42            ; EXPECTED RESULT OF CURRENT TEST
;3087            RECEIVED.DATA=43            ; ACTUAL RESULT OF CURRENT TEST
;3088            ADDRESS.DATA=44             ; OTHER PERTINENT DATA
;3089            ERROR.MASK=45               ; BITS TO CHECK IN RESULT
;3090            MODULE.NUM=46               ; STORAGE OF MODULE NUMBER (CODED)
;3091            LOOP.CNT=47                 ; STORAGE OF LOOP COUNT FOR CONSOLE LOOP
;3092                                        ; CONTROL.
;3093            ERROR.CONTROL=48            ; STORAGE FOR ERROR CONTROL (LOOP ON ERROR ETC.)
;3094             LOE=20                     ;  LOOP ON ERROR IF SET (BIT0)
;3095             NER=21                     ;  NO ERROR REPORT IF SET (BIT1)
;3096             BELL=22                    ;  BELL ON ERROR IF SET (BIT2)
;3097             HOE=23                     ;  HALT ON ERROR IF SET (BIT3)
;3098             SER=24                     ;  ENABLE ERROR REPORT ON CRD ERRORS IF SET
;3099             APT.PRESENT=25             ;  APT PRESENT IF SET (BIT5)
;3100             LOOP.COMMAND=26            ;  LOOP COMMAND TYPED IF SET (BIT 6)
;3101
;3102            APT.PARAM=49                ;APT RUN TIME PARAMETER REGS (MEM SIZE, HARD-
;3103                                        ; WARE CONFIG, SOFTWARE CONFIG IN BYTES 0, 1
;3104                                        ; AND 2 RESPECTIVELY. BYTE 3 FOR FUTURE USE)
;3105            APT.RESERVED=4A             ;RESERVED FOR FUTURE APT USE
;3106
;3107        ;MISCELLANEOUS CONSTANTS AND STORAGE
;3108
;3109            #AAAAAAAA=4C                ; CONSTANT
;3110            ALTER.ODD=4C                ; CONSTANT
;3111            #55555555=4D                ; CONSTANT
;3112            ALTER.EVEN=4D               ; CONSTANT
;3113            #0=4E                       ; CONSTANT
;3114            ZERO=4E                     ; CONSTANT
```

```
;3115            #FFFFFFFF=4F            ; CONSTANT
;3116            ONES=4F                 ; CONSTANT
;3117            PREVIOUS.ERROR=50       ; ERROR NUMBER OF LAST ERROR
;3118
;3119            DATA.1=51               ; STORAGE FOR FPA DATA
;3120            DATA.2=52               ; STORAGE FOR FPA DATA
;3121            DATA.3=53               ; STORAGE FOR FPA DATA
;3122            FIRST.FLT=54            ; STORAGE FOR FPA DATA
;3123            SEC.FLT=55              ; STORAGE FOR FPA DATA
;3124            THIRD.FLT=56            ; STORAGE FOR FPA DATA
;3125            T57=57                  ; TEMP LOCATION 57
;3126            T58=58                  ; TEMP LOCATION 58
;3127            T59=59                  ; TEMP LOCATION 59
;3128
;3129            BEDB=5A                 ;UBE (BUS EXERCISOR) DATA BUF REG
;3130            BECC=5B                 ;UBE (BUS EXERCISOR) CYCLE COUNT REG
;3131            BEBA=5C                 ;UBE (BUS EXERCISOR) ADDR REG
;3132            BECR1=5D                ;UBE (BUS EXERCISOR) CONTROL REG 1
;3133            CLEAR.ERR.ADDR=5E       ;UBE CLEAR ERROR REG ADDRESS
;3134            BECR2=5F                ;UBE (BUS EXERCISOR) CONTROL REG 2
;3135
;3136            #3(H)=60                ; CONSTANT
;3137            #12(H)=61               ; CONSTANT
;3138            #33333333=62            ; CONSTANT
;3139            #0F0F0F0F=63            ; CONSTANT
;3140            #00FF00FF=64            ; CONSTANT
;3141            #162(H)=65              ; CONSTANT FOR LS TRANSFER POINTER
;3142            BR7.IDENT=66            ; BR7 IDENTIFIER (1010(B) IN BITS 5-2)
;3143            BG7=67                  ; BG7 ADDRESS (BITS 2 AND 3 SET)
;3144
;3145    ;TEMPS FOR CPU TESTS
;3146            L30=30                  ;LS 30 TEMP (RESTORED TO 10000 AFTER USE)
;3147            L31=31                  ;LS 31 TEMP (RESTORE TO 20000 AFTER USE)
;3148            L53=53                  ;LS 53 TEMP
;3149            L73=73                  ;LS 73 TEMP
;3150
;3151    ;REGISTER ADDRESSES
;3152
;3153            CONTROL.OS=78           ; HARDWARE INDEX TO CONTROL WORDS (LS 40-4F)
;3154            SHIFT.OS(3-0)=79        ; 16 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;3155                                    ;  (LS 20-2F)
;3156            SHIFT.OS(4-0)=7B        ; 32 LOCATION HARDWARE INDEX TO SHIFT PATTERN
;3157                                    ;  (LS 20-3F)
;3158            OS=7C                   ; OS REGISTER
;3159            DEC.CON=7D              ; DECIMAL CARRIES
;3160            SIZE=7E                 ; SIZE REGISTER
;3161            MDT= 7E                 ; MEMORY REFERENCE DATA SIZE
;3162            INTERRUPT.VEC=7E        ; HARDWARE INTERRUPT VECTOR
;3163    ;
;3164    ;       THIS WORD IS THE HARDWARE CONDITION CODES.  THE CC'S CAN BE READ
;3165    ;       OR ARE WRITTEN HERE.  NO OTHER BITS IN THE PSL ARE EFFECTED.
;3166    ;
;3167            PSL.CC=7F               ; PSL CONDITION CODES
;3168
;3169
```

```
;3170    .TOC    "COMMON LS ASSIGNMENTS (TO REGISTERS)"
;3171    ;
;3172    ;UPPER HALF OF LS
;3173    ;
;3174            XGPR.OS=0F8             ; HARDWARE INDEX TO XGPRS
;3175            USER.INDEX(3-0)=0F9     ; 16 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
;3176                                    ;   AREA (LS LOCATIONS 0A0 THROUGH 0AF)
;3177            USER.INDEX(4-0)=0FB     ; 32 LOCATION HARDWARE INDEX TO DIAGNOSTIC DATA
;3178                                    ;   AREA (LS LOCATIONS 0A0 THROUGH 0BF)
;3179            CCR=0FC                 ; CONSOLE READ REGISTER
;3180            TIMER=0FD               ; INTERVAL TIMER VALUE.
;3181            ALU.CC=0FE              ; ALU CONDITION CODES
;3182    ;
;3183    ;       THIS LOCATION IS A WRITE ONLY REGISTER.  THE FOLLOWING BITS IN THE
;3184    ;       PSL ARE AFFECTED:
;3185    ;
;3186    ;       COMPATIBILITY MODE - BIT<31>
;3187    ;       CURRENT MODE - BITS<25:24>
;3188    ;       INTERRUPT PRIORITY LEVEL (IPL) - BITS<20:16>
;3189    ;       TRACE TRAP ENABLE (REALLY T OR TP) - BIT<4>
;3190    ;       NEGATIVE CONDITION CODE - BIT<3>
;3191    ;       ZERO CONDITION CODE - BIT<2>
;3192    ;       OVERFLOW CONDITION CODE - BIT<1>
;3193    ;       CARRY CONDITION CODE - BIT<0>
;3194    ;
;3195            PSL.HW=0FF              ; HARDWARE PSL BITS
;3196
;3197
;3198    ; These addresses are only accessible with the MOV micro instruction
;3199    ; starting at address 80(H).
;3200
;3201            FFFFFC00.MASK=80        ;A mask to mask out the upper 22 bits
;3202            T.4POINTER=81           ;Pointer into memory for the begining
;3203                                    ; of test 4.
;3204            LST.BPW=82              ;The address of the last bad parity word
;3205            FFFFFFF8.MASK=83        ;Mask to mask out the upper 29 bits.
;3206            T84=084                 ;temp location
;3207            #254=85                 ;254(H)
;3208            T7.POINTER=86           ;Pointer into memory for the begining of
;3209                                    ; test 7.
;3210            T8.POINTER=87           ;Pointer into memory for the begining of
;3211                                    ; test 8.
;3212            TA.POINTER=88           ;Pointer into memory for the begining of
;3213                                    ; test A.
;3214            #5=89                   ;constant
;3215            #6=8A                   ;constant
;3216            #300=8B                 ;constant  **FPA U-ADDRESS**
;3217            #00040300=8C            ;constant
;3218            #FFFF807F=8D            ;constant
;3219            TB.POINTER=8E           ;test B pointer
;3220            #FFFFBFFF=8F            ;constant
;3221            #7=90                   ;constant
;3222            TC.POINTER=91           ;test C pointer
;3223            #FFFF00FF=92            ;constant
;3224            TD.POINTER=93           ;test D pointer
```

```
;3225            #9=94                   ;constant
;3226            TE.POINTER=95           ;test E pointer
;3227            #1A=96                  ;constant
;3228            #2A7=97                 ; FPA u-address of double load
;3229            #2DC=98                 ; FPA u-address of third load
;3230            #368=99                 ; FPA u-address of first float store
;3231            #370=9A                 ; FPA u-address of second float store
;3232            #2DD=9B                 ; FPA u-address of third float store
;3233            T13.POINTER=9C          ;test 13 pointer
;3234            T14.POINTER=9D          ;test 14 pointer
;3235            T2A.POINTER=9E          ;test 2A pointer
;3236            #301=9F                 ;constant
;3237            #302=0A0                ;constant
;3238            #303=0A1                ;constant
;3239            #60300=0A2              ;constant
;3240            T2B.POINTER=0A3         ;test 2B pointer
;3241            #70300=0A4              ;constant
;3242            T2C.POINTER=0A5         ;test 2C pointer
;3243            #BFFFFFFF=0A6           ;constant
;3244            T2D.POINTER=0A7         ;test 2D pointer
;3245            #1F000=0A8              ;constant
;3246            BIT16.BEGIN.TEST=0A9    ;constant
;3247            #44300=0AA              ;constant
;3248            #40700=0AB              ;constant
;3249            #40B00=0AC              ;constant
;3250            #40F00=0AD              ;constant
;3251            #41300=0AE              ;constant
;3252            T2E.POINTER=0AF         ;constant
;3253            T2F.POINTER=0B0         ;constant
;3254            #42300=0B1              ;constant
;3255            #41700=0B2              ;constant
;3256            #41B00=0B3              ;constant
;3257            T30.POINTER=0B4         ;constant
;3258            #45F00=0B5              ;constant
;3259            #43F00=0B6              ;constant
;3260            #41F00=0B7              ;constant
;3261            T31.POINTER=0B8         ;constant
;3262            T32.POINTER=0B9         ;constant
;3263            T33.POINTER=0BA         ;constant
;3264            #8300=0BB               ;constant
;3265            T34.POINTER=0BC         ;constant
;3266            T35.POINTER=0BD         ;constant
;3267            T36.POINTER=0BE         ;constant
;3268            T37.POINTER=0BF         ;constant
;3269            T38.POINTER=0C0         ;constant
;3270            T39.POINTER=0C1         ;constant
;3271            T3A.POINTER=0C2         ;constant
;3272            T3B.POINTER=0C3         ;constant
;3273            T3C.POINTER=0C4         ;constant
;3274            T3D.POINTER=0C5         ;constant
;3275            TC5=0C6                 ;temp location
;3276            T6.POINTER=0C7          ;Pointer into memory for the begining
;3277                                    ; of test 6.
;3278            TC8=0C8                 ;temp location
;3279            TC9=0C9                 ;temp location
```

```
;3280            T3E.POINTER=0CA         ;constant
;3281            T3F.POINTER=0CB         ;constant
;3282            T40.POINTER=0CC         ;constant
;3283            T5.POINTER=0CD          ;constant
;3284            T9.POINTER=0CE          ;constant
;3285            TF.POINTER=0CF          ;constant
;3286            T10.POINTER=0D0         ;constant
;3287            T11.POINTER=0D1         ;constant
;3288            T12.POINTER=0D2         ;constant
;3289            T41.POINTER=0D3         ;constant
;3290            T42.POINTER=0D4         ;constant
;3291            T43.POINTER=0D5         ;constant
;3292            T26.POINTER=0D6         ;constant
;3293            T27.POINTER=0D7         ;constant
;3294            T28.POINTER=0D8         ;constant
;3295            T29.POINTER=0D9         ;constant
;3296            T44.POINTER=0DA         ;constant
;3297            T45.POINTER=0DB         ;constant
;3298            #11D=0DD                ; FPA u-address of store FWR 0, first float and
;3299                                    ;  EWR 0
;3300            #3BA=0DE                ; FPA u-address of store FWR 0, second float
;3301            #3BC=0DF                ; FPA u-address of store FWR 1, third float
;3302
;3303            T15.POINTER=0E0         ;test 15 pointer
;3304            T16.POINTER=0E1         ;test 16 pointer
;3305            T17.POINTER=0E2         ;test 17 pointer
;3306            T18.POINTER=0E3         ;test 18 pointer
;3307            T19.POINTER=0E4         ;test 19 pointer
;3308            T1A.POINTER=0E5         ;test 1A pointer
;3309            T1B.POINTER=0E6         ;test 1B pointer
;3310            T1C.POINTER=0E7         ;test 1C pointer
;3311            T1D.POINTER=0E8         ;test 1D pointer
;3312            T1E.POINTER=0E9         ;test 1E pointer
;3313            T1F.POINTER=0EA         ;test 1F pointer
;3314            T20.POINTER=0EB         ;test 20 pointer
;3315            T21.POINTER=0EC         ;test 21 pointer
;3316            T22.POINTER=0ED         ;test 22 pointer
;3317            T23.POINTER=0EE         ;test 23 pointer
;3318            T24.POINTER=0EF         ;test 24 pointer
;3319            SHIFT.RIGHT.EWR=0F5     ;FPA SHIFT EWR AND FWR RIGHT INSTRUCTION
;3320            SHIFT.LEFT.EWR=0F6      ;FPA SHIFT EWR AND FWR LEFT INSTRUCTION
;3321            SHIFT.LEFT.FWR=0F7      ;FPA SHIFT FWR LEFT INSTRUCTION
;3322
```

```
;3323   .page 'Microinstruction Formats"
;3324   ;
;3325   ;       The following is taken directly from the NEBULA Hardware
;3326   ;       Specification.
;3327   ;
;3328   ;                       MICROINSTRUCTION FORMATS
;3329   ;
;3330   ;
;3331   ;
;3332   ;
;3333   ;                   22 21       16 15          9 8 7 6 5 4       0
;3334   ;                   +-----+-------+-------+-------+-------+-------+
;3335   ;1. BASIC           |1|     DP    |    D ADRS   | B |CC |  SCTL   |
;3336   ;                   +-+-----------+-------------+---+---+---------+
;3337   ;
;3338   ;                   22  20 19 17 16            9 8 7 6 5 4       0
;3339   ;                   +-----+-------+-------+-------+-------+-------+
;3340   ;2. MOVE            |0 1 1| MDP |    XD ADRS    | B |CC |  SCTL   |
;3341   ;                   +-----+-----+---------------+---+---+---------+
;3342   ;
;3343   ;                   22  20 19       14 13 11   9 8 7 6 5 4       0
;3344   ;                   +-----+-------+-------+-------+-------+-------+
;3345   ;3. EXTENDED        |0 1 0|    XDP    | SI  | A | B |CC |  SCTL   |
;3346   ;                   +-----+-----------+-----+---+---+---+---------+
;3347   ;
;3348   ;                   22    19 18 16 15          9 8 7 6 5 4       0
;3349   ;                   +-----+-------+-------+-------+-------+-------+
;3350   ;4. MEM.REQ         |0 0 1 1| MF1 |    D ADRS   |MF2|DT |  SCTL   |
;3351   ;                   +-------+-----+-------------+---+---+---------+
;3352   ;
;3353   ;                   22    19 18 16 15   12 11  9 8 7 6 5 4       0
;3354   ;                   +-----+-------+-------+-------+-------+-------+
;3355   ;5. MISC            |0 0 1 0|P|0 0|   M1  |  M2   |0|R|0 0|  SCTL   |
;3356   ;                   +-------+-+---+-------+-------+-+-+---+---------+
;3357   ;
;3358   ;                   22    19 18 17                       4 3     0
;3359   ;                   +-----+-------+-------+-------+-------+-------+
;3360   ;6. JUMP            |0 0 0 1|OS|    JUMP ADDRESS           | JCTL  |
;3361   ;                   +-------+--+---------------------------+-------+
;3362   ;
;3363   ;                   22    19 18 17      12 11  9 8 7 6 5 4 3     0
;3364   ;                   +-----+-------+-------+-------+-------+-------+
;3365   ;7. DECODE          |0 0 0 0| | DEC.ADRS  |IFUNC| | | | | | JCTL  |
;3366   ;                   +-------++-----------+-----++-+-+-+-+-+-------+
;3367   ;                           |                   |   |   |   |   |
;3368   ;                   BACK UP PC          IB REQ---+   |   |   +---OPC/SPEC
;3369   ;                                                    |   |
;3370   ;                               SEL CM HI IR BYTE----+   +---LOAD RDEST
;3371   ;                                                    |
;3372   ;                                          LOAD OS---+
```

```
;3373    .page
;3374    ;
;3375    ;
;3376    ;1. "BASIC" Microinstruction
;3377    ;
;3378    ;    CSR<22> = 1    (OPCODE)
;3379    ;
;3380    ;        This opcode bit causes LS Adrs <7> to be cleared.
;3381    ;
;3382    ;    CSR<21:16>  (Data Path Control)
;3383    ;
;3384    ;        This field controls the data path. It controls the 2901 ALU,
;3385    ;        the ALU data source, the data destination in the 2901 array,
;3386    ;        and the write to the Local Store.
;3387    ;
;3388    ;    CSR<15:9>   (D Adrs <6:0>)
;3389    ;
;3390    ;        This field selects which of the 128 accessable Local Store
;3391    ;        locations to use.
;3392    ;
;3393    ;    CSR<8:7>    (B Adrs)
;3394    ;
;3395    ;        This field selects one of the four Working Registers.
;3396    ;
;3397    ;    CSR<6:5>    (Condition Codes)
;3398    ;
;3399    ;        This field specifies the data type and condition code control.
;3400    ;
;3401    ;    CSR<4:0>    (SCTL)
;3402    ;
;3403    ;        This field specifies the skip condition/micro sequencer
;3404    ;        function.
```

```
;3405    .page
;3406    ;
;3407    ;
;3408    ;2. 'MOVE'' Microinstruction
;3409    ;
;3410    ;     CSR<22:20> = 011 (OPCODE)
;3411    ;
;3412    ;        This combination of opcode bits allows CSR<16> to control
;3413    ;        LS Adrs <7>. This permits access of LS locations 80 - FF.
;3414    ;
;3415    ;     CSR<19:17> (Data Path Control)
;3416    ;
;3417    ;        This field is decoded to produce some data path control
;3418    ;        information.
;3419    ;
;3420    ;          0 LS[XD] <- WR[B], WR[B] <- MEM DATA* 4 LS[XD] <- MEM DATA*
;3421    ;          1 MEMORY <- LS [XD]*                  5 LS[XD] <- ACC/PORT
;3422    ;          2 Q <- XWR[B]                         6 LS[XD] <- WR[B] + OS
;3423    ;          3 WR[B] <- LS [XD]                    7 LS[XD] <- WR[B]
;3424    ;
;3425    ;     CSR<16:9>  (XD<7:0>)
;3426    ;
;3427    ;        This field specifies which of the 256 Local Store locations to
;3428    ;        use.
;3429    ;
;3430    ;     CSR<8:7>   (B Adrs)
;3431    ;
;3432    ;        This field specifies which of the four Working Registers to
;3433    ;        use. For MDP = 2, this field selects either the Backup PC or
;3434    ;        the VAR.
;3435    ;
;3436    ;     CSR<6:5>   (CC)
;3437    ;
;3438    ;        This field specifies the data type and condition code control.
;3439    ;
;3440    ;     CSR<4:0>   (SCTL)
;3441    ;
;3442    ;        This field specifies the Skip control. See Table 3.
;3443    ;
;3444    ;
;3445    ;* For information on which Working Registers and Local Store locations
;3446    ;  may  be  used for Memory Addresses and data source/destinations, see
;3447    ;  the Memory Management documentation.  Note that  these  restrictions
;3448    ;  are due to microcoding conventions and not hardware.
```

```
;3449    .page
;3450    ;
;3451    ;
;3452    ;3. ''EXTENDED'' Microinstruction
;3453    ;
;3454    ;    CSR<22:20> = 010    (OPCODE)
;3455    ;
;3456    ;        This opcode causes a function internal to the 2901 array; that
;3457    ;        is, an operation involving only Working Registers and the Q-
;3458    ;        Register.
;3459    ;
;3460    ;    CSR<19:14>  (Data Path Control)
;3461    ;
;3462    ;        This field is decoded to supply the 2901 ALU function code,
;3463    ;        the ALU Source control, the destination code, and the ALU
;3464    ;        Carry-In.
;3465    ;
;3466    ;    CSR<13:11>  (Shift Input)
;3467    ;
;3468    ;        This field selects the data to be shifted into a register,
;3469    ;        when a Shift function is being done.
;3470    ;
;3471    ;    CSR<10:9>   (A Adrs)
;3472    ;
;3473    ;        This field selects which Working Register to use as a source
;3474    ;        of data, when RAM A is selected to the ALU.
;3475    ;
;3476    ;    CSR<8:7>    (B Adrs)
;3477    ;
;3478    ;        This field selects which Working Register to use as a source
;3479    ;        (when RAM B is selected to the ALU) and/or destination (when
;3480    ;        the RAM is written) of data.
;3481    ;
;3482    ;    CSR<6:5>    (CC)
;3483    ;
;3484    ;        This field specifies the data type and condition code control.
;3485    ;
;3486    ;    CSR<4:0>    (SCTL)
;3487    ;
;3488    ;        This field specifies the skip condition/micro sequencer
;3489    ;        function.
```

```
;3490   .page
;3491   ;
;3492   ;
;3493   ;4. 'MEM.REQ'' Microinstruction
;3494   ;
;3495   ;    CSR<22:19> = 0011    (OPCODE)
;3496   ;
;3497   ;        This opcode causes a Memory Request to be started. The Local
;3498   ;        Store is output on on the D-Bus, to be used as the virtual
;3499   ;        address. Working Register 5 is written with this address.
;3500   ;
;3501   ;    CSR<18:16>, <8:7>   (Memory Function)
;3502   ;
;3503   ;        This field specifies to the Memory Controller the type of
;3504   ;        request: physical, virtual, type of access, etc. For the
;3505   ;        list of functions and the codes, see the NEBULA Memery Spec.
;3506   ;
;3507   ;    CSR<15:9>   (D Adrs <6:0>)
;3508   ;
;3509   ;        This field selects which of the 128 accessable Local Store
;3510   ;        locations to use as the virtual address.
;3511   ;
;3512   ;    CSR<6:5>    (Data Type)
;3513   ;
;3514   ;        This field specifies the data type of the request.
;3515   ;
;3516   ;                00:=BYTE
;3517   ;                01:=WORD
;3518   ;                10:=SIZE Reg
;3519   ;                11:=LONG
;3520   ;
;3521   ;    CSR<4:0>    (SCTL)
;3522   ;
;3523   ;        This field specifies the skip condition/micro sequencer
;3524   ;        function.
```

```
;3525   .page
;3526   ;
;3527   ;
;3528   ;5. 'MISC" Microinstruction
;3529   ;
;3530   ;    CSR<22:19> = 0010 (OPCODE)
;3531   ;
;3532   ;        This combination of opcode bits causes the data path to no-op,
;3533   ;        and enables the MISC decoders.
;3534   ;
;3535   ;    CSR<18>              (Port/Misc Select)
;3536   ;
;3537   ;        This bit defines the instruction to be either a Misc or a Port
;3538   ;        instruction.   For CSR<18> = 1, CSR<17:10> is the Command Byte
;3539   ;        to the Port, CSR<9:5> must be zero and CSR<4:0>  is  the  SCTL
;3540   ;        field.  For CSR<18> = 0, it is a Misc instruction and the rest
;3541   ;        of the word is defined as follows.
;3542   ;
;3543   ;    CSR<17:16>           (Not Used)
;3544   ;
;3545   ;    CSR<15:12>  (Misc Function 1)
;3546   ;
;3547   ;        These four bits are decoded to do the following functions:
;3548   ;
;3549   ;                0 = CLEAR State Reg<1:0>
;3550   ;                1 = CLEAR State Reg<1>, SET State Reg<0>
;3551   ;                2 = SET State Reg<1>, CLEAR State Reg<0>
;3552   ;                3 = CLEAR State Reg<0>
;3553   ;                4 = CLEAR State Reg<1>
;3554   ;                5 = SET State Reg<0>
;3555   ;                6 = SET State Reg<1>
;3556   ;                7 = SET REGISTER BACKUP MASK FLAG
;3557   ;                8 = SET ACC I/O MODE
;3558   ;                9 = SET PORT I/O MODE
;3559   ;                A = CLEAR WCS HI ADRS
;3560   ;                B = SET WCS HI ADRS
;3561   ;                C = CLEAR CPU ATTN AND CPU ACK
;3562   ;                D = SET CPU ACK
;3563   ;                E = SET CPU ATTN
;3564   ;                F = NOP
;3565   ;
;3566   ;    CSR<11:9>   (Misc Function 2)
;3567   ;
;3568   ;        This field is decoded to do the following functions:
;3569   ;
;3570   ;                0 = NOP
;3571   ;                1 = Mask IRD Interrupt Detection during Next State
;3572   ;                2 = Assert CPU Data Available and pass WR to ACC/Port
;3573   ;                3 = Trap ACC
;3574   ;                4 = Read ACC uPC
;3575   ;                5 = Assert XFER GRANT to Port
;3576   ;                6 = Mask Halt and T-Bit Traps
;3577   ;                7 = Write WR to Console Read Register
```

```
;3578   .page
;3579   ;
;3580   ;    CSR<8>      (MUST BE ZERO)
;3581   ;
;3582   ;    CSR<7>      (WR Address)
;3583   ;
;3584   ;
;3585   ;        This bit selects WR[0] or WR[1] to be put on the Y-Bus.
;3586   ;
;3587   ;    CSR<6:5>            (Not Used)
;3588   ;
;3589   ;    CSR<4:0>    (SCTL)
;3590   ;
;3591   ;        This  field  specifies  the  skip  condition/micro   sequencer
;3592   ;        function.
;3593   ;
```

```
;3594    .page
;3595    ;
;3596    ;
;3597    ;6. "JUMP" Microinstruction
;3598    ;
;3599    ;    CSR<22:19> = 0001    (OPCODE)
;3600    ;
;3601    ;        This opcode causes the data path to no-op, and the micro
;3602    ;        sequencer to execute a JUMP.
;3603    ;
;3604    ;    CSR<18>     (Select OS)
;3605    ;
;3606    ;        This bit, when asserted, causes OS<4:0> to be OR'ed with the
;3607    ;        five low bits of the of the jump microaddress.
;3608    ;
;3609    ;    CSR<17:4>   (Jump Microaddress)
;3610    ;
;3611    ;        If Select OS is CLEAR, this field specifies the fourteen-bit
;3612    ;        jump micro address. If Select OS is SET, CSR<17:9> is Jump
;3613    ;        Adrs<13:5> and OS<4:0> OR'ed with CSR<8:4> is Jump Adrs<4:0>.
;3614    ;
;3615    ;    CSR<3:0>    (JCTL)
;3616    ;
;3617    ;        This field specifies the jump condition/micro sequencer
;3618    ;        function.
```

```
;3619    .page
;3620    ;
;3621    ;
;3622    ;7. "DECODE" Microinstruction
;3623    ;
;3624    ;    A.  DESCRIPTION
;3625    ;
;3626    ;    CSR<22:19> = 0000 (OPCODE)
;3627    ;
;3628    ;        This combination of opcode bits causes the Local Store to
;3629    ;        access location 10 (the PC.) The Local Store drives the D-Bus.
;3630    ;        The 2901 is set up to input the data on the D-Bus, increment
;3631    ;        it, and output to the Y-Bus.  Thus, the Y-Bus contains the PC
;3632    ;        + 1.
;3633    ;
;3634    ;    CSR<18>     (BPC [asserted low])
;3635    ;
;3636    ;        If this bit is CLEAR, the incremented PC is written into the
;3637    ;        2901 RAM.  If it is SET, the RAM is not written.
;3638    ;
;3639    ;    CSR<17:12>  (DEC.ADRS <13:8>)
;3640    ;
;3641    ;        This data is taken to be the upper six bits of the next
;3642    ;        microaddress.  The low eight bits are supplied by the IRD ROM.
;3643    ;        The actual JUMP/JSR/NO-OP is determined by the JCTL field.
;3644    ;
;3645    ;    CSR<11:9>   (IFUNC)
;3646    ;
;3647    ;        This is the IFUNC field.  It defines which fork is being
;3648    ;        taken.
;3649    ;
;3650    ;    CSR<8>      (IB.REQ)
;3651    ;
;3652    ;        This is the IB.REQ bit.  When it is CLEAR, the LS is not
;3653    ;        written, no interaction with memory is initiated and the
;3654    ;        instructin buffer is not affected.
;3655    ;
;3656    ;        When it is SET:
;3657    ;
;3658    ;        The incremented PC (valid on the Y-Bus by the end of the
;3659    ;        cycle) is re-written to Local Store location 0.  This
;3660    ;        completes the PC increment function.
;3661    ;
;3662    ;        A Conditional Memory Request is executed.  If the two low bits
;3663    ;        of the PC are not 11, the request is not initiated.  If the
;3664    ;        two low bits are set, an IB PREFETCH is done.  The
;3665    ;        (unincremented) PC is output to the memory and the CPU grant
;3666    ;        (CPUG) signal is examined.  If CPUG is low by T120, the CPU
;3667    ;        will stall until the CPUG signal becomes true.  After the
;3668    ;        request is completed, the next state entered is dependent upon
;3669    ;        the JCTL field.
```

```
;3670    .page
;3671    ;
;3672    ;    CSR<7>      (SEL CM HI IR BYTE)
;3673    ;
;3674    ;        This bit enables the upper  byte  of  the  Compatibility  Mode
;3675    ;        instruction  in the Prefetch register to drive the IB-Bus.  It
;3676    ;        is used only while in Compatibility Mode, when  IRD  is  being
;3677    ;        done.  It must never be asserted in VAX Mode.
;3678    ;
;3679    ;
;3680    ;    CSR<6>      (LD.OS)
;3681    ;
;3682    ;        If this bit is SET, the eight-bit data  on  the  IB  Bus  (the
;3683    ;        output  of  the  instruction  buffer)  is  loaded  into the OS
;3684    ;        register.  This load is dependent upon Compatibility Mode  and
;3685    ;        LD  R  Dest  (If  CM.AND.LD R Dest, OS<3> <- 0.) If the bit is
;3686    ;        CLEAR, OS is not affected.
;3687    ;
;3688    ;    CSR<5>      (LD R DEST)
;3689    ;
;3690    ;        If this bit is SET, the R  Dest  Skip  bit  will  be  SET  for
;3691    ;        Specifier Mode equal to 5 and CLEARED for Mode not equal to 5,
;3692    ;        in VAX mode, or mode 0 while in Compatibility Mode.   If  this
;3693    ;        bit is CLEAR, the R Dest bit will be unaffected.
;3694    ;
;3695    ;    CSR<4>              (OPC/SPEC)
;3696    ;
;3697    ;        This bit (effectively another IFUNC bit) defines the  data  to
;3698    ;        be  decoded;   that is, if it is an Opcode or a Specifier.  In
;3699    ;        hardware, it selects which ROM is to drive the  micro  address
;3700    ;        bus.
;3701    ;
;3702    ;    CSR<3:0>    (JCTL)
;3703    ;
;3704    ;        These four bits are decoded to make one of 16 functions.   See
;3705    ;        Table 3.
```

```
;3706    .page "Tables"
;3707    ;
;3708    ;                                   TABLES
;3709    ;
;3710    ;
;3711    ;
;3712    ;                       Table 1.  2901 Control Decoding
;3713    ;
;3714    ;
;3715    ;       The 2901 control decoder examines CSR<22:14>. This nine-bit
;3716    ;       field  of  the microword changes in meaning for the different
;3717    ;       cases of micro opcode.
;3718    ;
;3719    ;       CSR      22  21  20  19  18  17  16  15  14
;3720    ;                --  --  --  --  --  --  --  --  --
;3721    ;1. BASIC         1  [Six bit function code]  X   X
;3722    ;2. MOVE          0   1   1  [function]   X   X   X
;3723    ;3. EXTENDED      0   1   0  [Six bit function code]
;3724    ;4. MEM.REQ       0   0   1   1   X   X   X   X   X      WR[5] <- D
;3725    ;5. MISC          0   0   1   0   X   X   X   X   X      Y-Bus <- WR
;3726    ;6. JUMP          0   0   0   1   X   X   X   X   X      NO-OP
;3727    ;7. DECODE        0   0   0   0  BPC* X   X   X   X
;3728    ;
;3729    ;  If BPC = 0, Write WR; else, No-op
;3730    ;
;3731    ;     This decoder is implemented using a ROM and a PAL. 512
;3732    ;     locations are needed for CSR<22:14> to index into. The
;3733    ;     decoding logic supplies 10 control bits:
;3734    ;
;3735    ;                 ALU Source Select   (3)
;3736    ;                 ALU Function   (3)
;3737    ;                 ALU Result Destination   (3)
;3738    ;                 ALU Carry-In   (1)
;3739    ;
;3740    ;     The ROM addresses are therefore allocated in the following way:
;3741    ;
;3742    ;     Location     0 -   F       Incr. D-Bus, Output ALU, No Write.
;3743    ;                 10 -  1F       Incr. D-Bus, Output ALU, Write RAM.
;3744    ;                 20 -  3F       No-Op.
;3745    ;                 40 -  5F       Output WR, bypassing ALU.
;3746    ;                 60 -  7F       Pass D-Bus through ALU and write RAM.
;3747    ;                 80 -  BF       CSR<19:14> indicates 2901 function.
;3748    ;                 C0 -  FF       CSR<19:17> indicates 2901 function.
;3749    ;                100 - 1FF       CSR<21:16> indicates 2901 function.
;3750    ;
;3751    ;     For the detailed function table see the Microcode Listing.
```

```
;3752    .page
;3753    ;
;3754    ;
;3755    ;                          Table 2. Local Store Write
;3756    ;
;3757    ;       The Local Store Write Controller examines CSR<22:17>, CSR<6>, the
;3758    ;       SIZE  register  (SIZE<1:0>)  and the Compatibility Mode bit (CM).
;3759    ;       These eight bits are decoded to determine which parts of  the  LS
;3760    ;       to  write, if any.  The LS is written with data from the 2901 ALU
;3761    ;       (Y-Bus).
;3762    ;
;3763    ;                    CSR      22  21  20  19  18  17
;3764    ;                             --  --  --  --  --  --
;3765    ;          1. BASIC            1   0   X   X   X   X     No Write
;3766    ;                              1   1   X   X   X   X     Write *
;3767    ;          2. MOVE             0   1   1   0   X   1     No Write
;3768    ;                              0   1   1   0   1   X     No Write
;3769    ;                              0   1   1   1   X   X     Write *
;3770    ;          3. EXTENDED         0   1   1   X   X   X     No Write
;3771    ;          4. MEM.REQ          0   0   1   1   X   X     No Write
;3772    ;          5. MISC             0   0   1   0   X   X     No Write
;3773    ;          6. JUMP             0   0   0   1   X   X     No Write
;3774    ;          7. DECODE           0   0   0   0   X   X     Conditional Write **
;3775    ;
;3776    ;     * LS Write, Data Type Dependent. These cases now examine CSR<6>.
;3777    ;
;3778    ;       If CSR<6> = 0, and CM = 0   Write LS<31:0>  (Data Type = LONG)
;3779    ;       If CSR<6> = 0, and CM = 1   Write LS<15:0>  (Compatibility
;3780    ;                                                    Mode. DT = WORD)
;3781    ;
;3782    ;       If CSR<6> = 1, the Write is conditional on the SIZE register.
;3783    ;
;3784    ;       If CM = 0   SIZE<1:0> = 00  Write LS<7:0>  (BYTE)
;3785    ;                             = 01  Write LS<15:0> (WORD)
;3786    ;                             = 1X  Write LS<31:0> (LONG)
;3787    ;
;3788    ;       If CM = 1  Write LS<15:0>  (WORD)
;3789    ;
;3790    ;     ** If CSR<8> = 0, No Write
;3791    ;        If CSR<8> = 1, Write LS<31:0>  (Writes incremented PC)
```

```
;3792    .page
;3793    ;
;3794    ;
;3795    ;                    Table 3. SKIP/JUMP Conditions
;3796    ;
;3797    ;This table summarizes the Skip and Jump conditions, and the other
;3798    ;special functions of the micro sequencer.
;3799    ;
;3800    ;SCTL codes 0-F, 17-1F are Skip conditions. The codes 10-17 execute
;3801    ;special functions. JCTL codes 0-B are Jump conditions, and C-F
;3802    ;execute special functions.
;3803    ;
;3804    ;
;3805    ;Sctl    Function                Sctl    Function
;3806    ;----    --------                ----    --------
;3807    ; 00     Not ALU N                10     RTN+1 if No Mem Err
;3808    ; 01     Not ALU Z                11     Loop if Not ALU Z
;3809    ; 02     Not ALU V                12     POP Stack
;3810    ; 03     ALU C                    13     Loop if ALU C
;3811    ; 04     Not PSL C                14     Return
;3812    ; 05     Branch False             15     No Skip (NOP)
;3813    ; 06     RDEST                    16     RTN+1
;3814    ; 07     Not Inter. Req.          17     Loop if No Inter. and No Sync
;3815    ; 08     ALU N                    18     Console ATTN
;3816    ; 09     ALU Z                    19     Console ACK
;3817    ; 0A     ALU V                    1A     Port Interrupt Pending
;3818    ; 0B     Not ALU C                1B     Reg. Backup Mask Flag
;3819    ; 0C     State 0                  1C     ALU N.XOR.ALU V
;3820    ; 0D     State 1                  1D     Not Memory Error
;3821    ; 0E     Not State 0              1E     Skip
;3822    ; 0F     Not State 1              1F     Sync
;3823    ;
;3824    ;
;3825    ;Jctl    Function                Jctl    Function
;3826    ;----    --------                ----    --------
;3827    ; 00     Not ALU N                08     ALU N
;3828    ; 01     Not ALU Z                09     ALU Z
;3829    ; 02     Not ALU V                0A     ALU V
;3830    ; 03     ALU C                    0B     Not ALU C
;3831    ; 04     JUMP                     0C     JSR
;3832    ; 05     Branch False             0D     JSR if IB Valid
;3833    ; 06     RDEST                    0E     No Jump; Skip if IB Valid
;3834    ; 07     Not Interr. Req          0F     IB Valid
;3835    ;
;3836
```

```
;3837   .PAGE  "2901 ALU Control Description"
;3838
;3839   ;
;3840   ;           The 2901 ALU is the heart  of  the  CPU  module.   It  is
;3841   ;       responsible for all CPU calculations.  The control lines going
;3842   ;       to I0 through I8 control the function, input,  and  output  of
;3843   ;       the  2901.   When  a  failure  involving  the 2901 occurs, the
;3844   ;       logical place to start looking  is  the  control  lines.   The
;3845   ;       following  tables  describe what the control signals should be
;3846   ;       for various functions.  Use it whenever the control lines  are
;3847   ;       to  be  checked.  The test descriptions tell the operator what
;3848   ;       function the 2901 is supposed to be doing.
;3849   ;
;3850   ;           The SRC CTL on lines I0 through I2 (pins  12  through  14
;3851   ;       respectively)  control  where  the  inputs to the internal ALU
;3852   ;       within the 2901 come from.  The inputs to the internal ALU are
;3853   ;       labeled R and S.  R is one operand (4 bits) and S is the other
;3854   ;       operand (also 4 bits).  The R input can come  from  a  Working
;3855   ;       Register  (labeled  A), the D bus (direct input), or be forced
;3856   ;       to 0.  The S  input  can  come  from  either  set  of  working
;3857   ;       registers  (labeled  A and B), the Q register, or forced to 0.
;3858   ;       A table of these control signals follow:
;3859   ;
;3860   ;
;3861   ;
;3862   ;
;3863   ;
;3864   ;
;3865   ;
;3866   ;
;3867   ;
;3868   ;
;3869   ;
;3870   ;
;3871   ;
;3872   ;
;3873   ;
;3874   ;
;3875   ;
;3876   ;
;3877   ;
;3878   ;
;3879   ;
;3880   ;
;3881   ;
;3882   ;           The ALU CTL on lines I3  through  I5  (pins  26,  28,  27
;3883   ;       respectively)  control  what function the ALU is to perform on
;3884   ;       the two operands R and S.  A table follows:
```

				SRC operands	
I2	I1	I0	octal code	R	S
L	L	L	0	A	Q
L	L	H	1	A	B
L	H	L	2	0	Q
L	H	H	3	0	B
H	L	L	4	0	A
H	L	H	5	D	A
H	H	L	6	D	Q
H	H	H	7	D	0

ALU SOURCE OPERAND CONTROL

```
;3885    .page
;3886    ;
;3887    ;
;3888    ;
;3889    ;
;3890    ;
;3891    ;
;3892    ;
;3893    ;
;3894    ;
;3895    ;
;3896    ;
;3897    ;
;3898    ;
;3899    ;
;3900    ;
;3901    ;
;3902    ;
;3903    ;
;3904    ;
;3905    ;
;3906    ;
;3907    ;
;3908    ;
;3909    ;
;3910    ;
;3911    ;
;3912    ;
;3913    ;
;3914    ;
;3915    ;
;3916    ;
;3917    ;
;3918    ;
;3919    ;
;3920    ;
;3921    ;
;3922    ;
;3923    ;
;3924    ;
;3925    ;
;3926    ;
;3927    ;
;3928    ;
;3929    ;
;3930    ;
;3931    ;
;3932    ;
;3933    ;
;3934    ;
;3935    ;
;3936    ;
;3937    ;
;3938    ;
```

I5	I4	I3	octal code	ALU function
L	L	L	0	R PLUS S
L	L	H	1	S MINUS R
L	H	L	2	R MINUS S
L	H	H	3	R OR S
H	L	L	4	R AND S
H	L	H	5	R NOT AND S
H	H	L	6	R XOR S
H	H	H	7	R XNOR S

ALU FUNCTION CONTROL

The DST CTL on lines I6 through I8 (pins 5, 7, 6 respectively) control what goes out onto the Y bus (either the internal ALU result or a WR directly) and what happens internal to the 2901 (shifting, writing the Q register etc). Internal shifting left or right is accomplished with this control. Below is a table explaining the various destination possibilities. The arrow refers to where the output will end up. B indicates the WR addressed by the B ADDR, F refers to the output of the internal ALU, and A refers to the output of the WR addressed by the A ADDR. Up is toward the MSB, while down is toward the LSB.

nomenclature in listing	I8	I7	I6	octal code	RAM function shift	RAM function load	Q-reg function shift	Q-reg function load	Y output
LOAD Q	L	L	L	0	NONE	NONE	NONE	F->Q	F
NOP	L	L	H	1	NONE	NONE	NONE	NONE	F
WRITE.B.A	L	H	L	2	NONE	F->B	NONE	NONE	A
WRITE.B.F	L	H	H	3	NONE	F->B	NONE	NONE	F
RSHF.RAM.Q	H	L	L	4	DOWN	F/2->B	DOWN	Q/2->Q	F
RSHF.RAM	H	L	H	5	DOWN	F/2->B	NONE	NONE	F
LSHF.RAM.Q	H	H	L	6	UP	2F->B	UP	2Q->Q	F
LSHF.RAM	H	H	H	7	UP	2F->B	NONE	NONE	F

ALU DESTINATION CONTROL

```
;3939   .page
;3940   .TOC    "DIAGNOSTIC MACROS"
;3941
;3942   LS.DATA.WORD/=<15:0>
;3943   UPPER.BYTE/=<23:16>
;3944
;3945   WORD[]          "UPPER.BYTE/0,LS.DATA.WORD/@1"
;3946   COUNT.LS[]      "UPPER.BYTE/0,LS.DATA.WORD/@1"
;3947   COUNT.MM[]      "UPPER.BYTE/1,LS.DATA.WORD/@1"
;3948   START.ADR[]     "UPPER.BYTE/0,LS.DATA.WORD/@1"
;3949
;3950   ASCII.LIT/=<15:0>
;3951
;3952           CA=4341         ;ASCII VALUE FOR FILE NAMES
;3953           CB=4342         ;  |
;3954           CC=4343         ;  |
;3955           CD=4344         ;  |
;3956           CE=4345         ;  |
;3957           CF=4346         ;  |
;3958           CG=4347         ;  V
;3959           CH=4348
;3960
;3961   ASCII []        "UPPER.BYTE/0,ASCII.LIT/@1"
;3962
;3963   .REGION/0,6F
```

```
                ;3964   .PAGE   "TEST POINTERS"
                ;3965   ;
                ;3966   ;       This portion contains the pointers to all tests in this section.  The
                ;3967   ;       WCS address of the JUMP instruction corresponds to the test number.
                ;3968   ;       E.g. WCS 5 contains a JUMP to test 5.  All unused test numbers will
                ;3969   ;       contain a JUMP to an error routine.  This portion is 63. locations
                ;3970   ;       long, allowing for up to 63. tests per section, from WCS address 1 to
                ;3971   ;       WCS address 3F.
                ;3972   ;
                ;3973   1:                                      ;TEST POINTERS BEGIN AT WCS ADDRESS 1
                ;3974
U 0001, 887F,74 ;3975           JMP [T.1]                       ;GO TO TEST 1
U 0002, 0882,14 ;3976           JMP [T.2]                       ;GO TO TEST 2
U 0003, 0883,F4 ;3977           JMP [T.3]                       ;GO TO TEST 3
U 0004, 0888,D4 ;3978           JMP [T.4]                       ;GO TO TEST 4
U 0005, 888B,04 ;3979           JMP [T.5]                       ;GO TO TEST 5
U 0006, 0890,E4 ;3980           JMP [T.6]                       ;GO TO TEST 6
U 0007, 8892,44 ;3981           JMP [T.7]                       ;GO TO TEST 7
U 0008, 0894,34 ;3982           JMP [T.8]                       ;GO TO TEST 8
U 0009, 8896,54 ;3983           JMP [T.9]                       ;GO TO TEST 9
U 000A, 0898,94 ;3984           JMP [T.A]                       ;GO TO TEST A
U 000B, 88A1,74 ;3985           JMP [T.B]                       ;GO TO TEST B
U 000C, 88AB,84 ;3986           JMP [T.C]                       ;GO TO TEST C
U 000D, 08B0,F4 ;3987           JMP [T.D]                       ;GO TO TEST D
U 000E, 08B3,A4 ;3988           JMP [T.E]                       ;GO TO TEST E
U 000F, 88B7,C4 ;3989           JMP [T.F]                       ;GO TO TEST F
U 0010, 08C0,44 ;3990           JMP [T.10]                      ;GO TO TEST 10
U 0011, 88CF,04 ;3991           JMP [T.11]                      ;GO TO TEST 11
U 0012, 88DA,E4 ;3992           JMP [T.12]                      ;GO TO TEST 12
U 0013, 88E1,64 ;3993           JMP [T.13]                      ;GO TO TEST 13
U 0014, 88E7,C4 ;3994           JMP [T.14]                      ;GO TO TEST 14
U 0015, 08EF,A4 ;3995           JMP [T.15]                      ;GO TO TEST 15
U 0016, 88F7,84 ;3996           JMP [T.16]                      ;GO TO TEST 16
U 0017, 08FF,B4 ;3997           JMP [T.17]                      ;GO TO TEST 17
U 0018, 0912,64 ;3998           JMP [T.18]                      ;GO TO TEST 18
U 0019, 8920,64 ;3999           JMP [T.19]                      ;GO TO TEST 19
U 001A, 892D,24 ;4000           JMP [T.1A]                      ;GO TO TEST 1A
U 001B, 0933,04 ;4001           JMP [T.1B]                      ;GO TO TEST 1B
U 001C, 0938,E4 ;4002           JMP [T.1C]                      ;GO TO TEST 1C
U 001D, 893E,34 ;4003           JMP [T.1D]                      ;GO TO TEST 1D
U 001E, 0943,84 ;4004           JMP [T.1E]                      ;GO TO TEST 1E
U 001F, 894A,94 ;4005           JMP [T.1F]                      ;GO TO TEST 1F
U 0020, 894E,B4 ;4006           JMP [T.20]                      ;GO TO TEST 20
U 0021, 0952,84 ;4007           JMP [T.21]                      ;GO TO TEST 21
U 0022, 0956,64 ;4008           JMP [T.22]                      ;GO TO TEST 22
U 0023, 095B,24 ;4009           JMP [T.23]                      ;GO TO TEST 23
U 0024, 0960,94 ;4010           JMP [T.24]                      ;GO TO TEST 24
U 0025, 8963,B4 ;4011           JMP [T.25]                      ;GO TO TEST 25
U 0026, 8966,D4 ;4012           JMP [T.26]                      ;GO TO TEST 26
U 0027, 098A,E4 ;4013           JMP [T.27]                      ;GO TO TEST 27
U 0028, 8999,B4 ;4014           JMP [T.28]                      ;GO TO TEST 28
U 0029, 09A6,64 ;4015           JMP [T.29]                      ;GO TO TEST 29
U 002A, 89AE,F4 ;4016           JMP [T.2A]                      ;GO TO TEST 2A
U 002B, 89B3,34 ;4017           JMP [T.2B]                      ;GO TO TEST 2B
U 002C, 89B5,54 ;4018           JMP [T.2C]                      ;GO TO TEST 2C
```

```
U 002D, 89B7,D4 ;4019          JMP [T.2D]                ;GO TO TEST 2D
U 002E, 09BC,44 ;4020          JMP [T.2E]                ;GO TO TEST 2E
U 002F, 89C1,A4 ;4021          JMP [T.2F]                ;GO TO TEST 2F
U 0030, 89C6,24 ;4022          JMP [T.30]                ;GO TO TEST 30
U 0031, 89CC,14 ;4023          JMP [T.31]                ;GO TO TEST 31
U 0032, 09CE,D4 ;4024          JMP [T.32]                ;GO TO TEST 32
U 0033, 09D4,C4 ;4025          JMP [T.33]                ;GO TO TEST 33
U 0034, 09D7,F4 ;4026          JMP [T.34]                ;GO TO TEST 34
U 0035, 89DD,14 ;4027          JMP [T.35]                ;GO TO TEST 35
U 0036, 89E0,34 ;4028          JMP [T.36]                ;Go TO TEST 36
U 0037, 09E3,24 ;4029          JMP [T.37]                ;GO TO TEST 37
U 0038, 09EC,84 ;4030          JMP [T.38]                ;GO TO TEST 38
U 0039, 89F7,C4 ;4031          JMP [T.39]                ;GO TO TEST 39
U 003A, 09FA,64 ;4032          JMP [T.3A]                ;GO TO TEST 3A
U 003B, 8A03,14 ;4033          JMP [T.3B]                ;GO TO TEST 3B
U 003C, 8A05,B4 ;4034          JMP [T.3C]                ;GO TO TEST 3C
U 003D, 0A0B,14 ;4035          JMP [T.3D]                ;GO TO TEST 3D
U 003E, 0A16,B4 ;4036          JMP [T.3E]                ;GO TO TEST 3E
U 003F, 8A1C,C4 ;4037          JMP [T.3F]                ;GO TO TEST 3F
U 0040, 0A2D,A4 ;4038          JMP [T.40]                ;GO TO TEST 40
U 0041, 8A41,E4 ;4039          JMP [T.41]                ;GO TO TEST 41
U 0042, 8A5A,B4 ;4040          JMP [T.42]                ;GO TO TEST 42
U 0043, 0A6F,94 ;4041          JMP [T.43]                ;GO TO TEST 43
U 0044, 0AD4,A4 ;4042          JMP [T.44]                ;GO TO TEST 44
U 0045, 0AE2,04 ;4043          JMP [T.45]                ;GO TO TEST 45
U 0046, 886B,E4 ;4044          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4045                                    ; IN THIS OVERLAY
U 0047, 886B,E4 ;4046          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4047                                    ; IN THIS OVERLAY
U 0048, 886B,E4 ;4048          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4049                                    ; IN THIS OVERLAY
U 0049, 886B,E4 ;4050          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4051                                    ; IN THIS OVERLAY
U 004A, 886B,E4 ;4052          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4053                                    ; IN THIS OVERLAY
U 004B, 886B,E4 ;4054          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4055                                    ; IN THIS OVERLAY
U 004C, 886B,E4 ;4056          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4057                                    ; IN THIS OVERLAY
U 004D, 886B,E4 ;4058          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4059                                    ; IN THIS OVERLAY
U 004E, 886B,E4 ;4060          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4061                                    ; IN THIS OVERLAY
U 004F, 886B,E4 ;4062          JMP [ERR.NO.TEST]         ;GO TO ERROR ROUTINE. NO SUCH TEST NUMBER
                ;4063                                    ; IN THIS OVERLAY
```

```
                ;4064   .PAGE   "DIAGNOSTIC POINTERS"
                ;4065   ;
                ;4066   ;       This section contains all the pointers needed for this diagnostic
                ;4067   ;       overlay.  The first pointer (at WCS location 0) points to a data
                ;4068   ;       transfer routine.  It is the first location executed after a new
                ;4069   ;       WCS load.  The insruction here is a JUMP to TRANSFER.POINT which may
                ;4070   ;       contain instructions to transfer data.  If no data is to be trans-
                ;4071   ;       ferred, or at the completion of the data transfers, the CPU will
                ;4072   ;       issue CPU ATTN with all bits of the control and status word clear.
                ;4073   ;
                ;4074   ;       The next 79 locations (from WCS location 1 to WCS location 4F)
                ;4075   ;       contain pointers to each test in this overlay.  The pointers are
                ;4076   ;       JUMP instructions to the test number corresponding to the location
                ;4077   ;       number.  E.g. location 5 will contain a JUMP to test 5.  All unused
                ;4078   ;       test numbers will contain a JUMP to an error routine.  This portion
                ;4079   ;       appears after the definitions in this listing.
                ;4080   ;
                ;4081   ;       Finally the next 32. locations (from WCS location 50 to 6F) contain
                ;4082   ;       pointers (Jump instructions) to WCS subroutines which are to be used by
                ;4083   ;       the console diagnostic monitor.
                ;4084   ;
U 0000, 086C,24 ;4085   0:      JMP [TRANSFER.POINT]                    ;GO TO ROUTINE THAT LOADS LS 7 WITH
                ;4086                                                   ; POINTER TO DATA SECTION AND ISSUES
                ;4087                                                   ; CPU ATTN WITH TRANSFER BIT SET.
                ;4088
                ;4089   50:                                             ;START AT WCS LOCATION 50 FOR SUB-
                ;4090                                                   ; ROUTINE POINTERS
                ;4091   ;
                ;4092   ; SUBROUTINE POINTERS
                ;4093   ;
U 0050, 0860,84 ;4094           JMP [DEPOSIT.CSR]                       ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4095                                                   ; MCT CSR FOR DEPOSIT CSR COMMAND
U 0051, 0860,D4 ;4096           JMP [EXAMINE.CSR]                       ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4097                                                   ; MCT CSR FOR EXAMINE CSR COMMAND
U 0052, 0861,F4 ;4098           JMP [DEPOSIT.TB]                        ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4099                                                   ; MCT TRANSLATION BUFFER FOR DEPOSIT
                ;4100                                                   ; TB COMMAND
U 0053, 8863,A4 ;4101           JMP [EXAMINE.TB]                        ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4102                                                   ; MCT TRANSLATION BUFFER FOR EXAMINE
                ;4103                                                   ; TB COMMAND
U 0054, 8865,C4 ;4104           JMP [DEPOSIT.MM]                        ;GO TO WCS SUBROUTINE WHICH WRITES MAIN
                ;4105                                                   ; MEMORY FOR DEPOSIT MM COMMAND.  ALSO
                ;4106                                                   ; USED TRANSFERING DATA FROM WCS TO
                ;4107                                                   ; MAIN MEMORY.
U 0055, 0866,24 ;4108           JMP [EXAMINE.MM]                        ;GO TO WCS SUBROUTINE WHICH READS MAIN
                ;4109                                                   ; MEMORY FOR EXAMINE MM COMMAND.
U 0056, 0868,94 ;4110           JMP [SAVE.WR]                           ;GO TO WCS SUBROUTINE WHICH STORES THE
                ;4111                                                   ; CONTENTS OF WR0-WR3 IN LS 0-3
U 0057, 8868,E4 ;4112           JMP [RESTORE.WR]                        ;GO TO WCS SUBROUTINE WHICH RESTORES THE
                ;4113                                                   ; CONTENTS OF WR0-WR3 FROM LS 0-3
U 0058, 8864,24 ;4114           JMP [DEPOSIT.UBS]                       ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4115                                                   ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 0059, 0865,44 ;4116           JMP [EXAMINE.UBS]                       ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4117                                                   ; UNIBUS MAP ON THE MEMORY CONTROLLER
U 005A, 0866,84 ;4118           JMP [EXAMINE.ICSR]                      ;GO TO WCS SUBROUTINE WHICH READS THE
```

```
                ;4119                                   ; IDC CSR
U 005B, 0866,B4 ;4120           JMP [EXAMINE.IDAR]      ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4121                                   ; IDC DAR
U 005C, 0866,E4 ;4122           JMP [EXAMINE.DBUF]      ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4123                                   ; IDC DBUF
U 005D, 8867,14 ;4124           JMP [EXAMINE.PATT]      ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4125                                   ; IDC ECC PATTERN
U 005E, 8867,44 ;4126           JMP [EXAMINE.POSIT]     ;GO TO WCS SUBROUTINE WHICH READS THE
                ;4127                                   ; IDC ECC POSITION
U 005F, 0867,94 ;4128           JMP [DEPOSIT.ICSR]      ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4129                                   ; IDC CSR
U 0060, 0867,C4 ;4130           JMP [DEPOSIT.IDAR]      ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4131                                   ; IDC DAR
U 0061, 0867,F4 ;4132           JMP [DEPOSIT.DBUF]      ;GO TO WCS SUBROUTINE WHICH WRITES THE
                ;4133                                   ; IDC DBUF
U 0062, 8868,24 ;4134           JMP [CLEAR.FIFO.ADDR]   ;GO TO WCS THE SUBROUTINE WHICH CLEARS
                ;4135                                   ; THE IDC FIFO ADDRESS
U 0063, 8868,44 ;4136           JMP [SELECT.FIFO.A]     ;GO TO THE WCS SUBROUTINE WHICH SELECTS
                ;4137                                   ; FIFO A
U 0064, 0868,64 ;4138           JMP [SELECT.FIFO.B]     ;GO TO THE WCS SUBROUTINE WHICH SELECTS
                ;4139                                   ; FIFO B
U 0065, DB00,15 ;4140           NOP                     ;NOT USED
U 0066, DB00,15 ;4141           NOP                     ;NOT USED
U 0067, DB00,15 ;4142           NOP                     ;NOT USED
U 0068, DB00,15 ;4143           NOP                     ;NOT USED
U 0069, DB00,15 ;4144           NOP                     ;NOT USED
U 006A, DB00,15 ;4145           NOP                     ;NOT USED
U 006B, DB00,15 ;4146           NOP                     ;NOT USED
U 006C, DB00,15 ;4147           NOP                     ;NOT USED
U 006D, DB00,15 ;4148           NOP                     ;NOT USED
U 006E, DB00,15 ;4149           NOP                     ;NOT USED
U 006F, DB00,15 ;4150           NOP                     ;NOT USED
```

```
                ;4151   .PAGE
                ;4152   .REGION/70,5FF
                ;4153   .TOC    "LS DATA SECTION"
                ;4154
                ;4155   ;+++
                ;4156   ;       This section lists the data that will be transfered to LS by the con-
                ;4157   ;       sole as part of the initialization performed in test 0.  The TRANSFER
                ;4158   ;       DATA routine will point to this data area.  LS contains the constants,
                ;4159   ;       control and status word, and temporary storage locations used by the
                ;4160   ;       tests and subroutines of the microdiagnostic.
                ;4161   ;         The LS is 32 bits wide.  Each of these words is 16 bits, so two con-
                ;4162   ;       secutive words listed here will be loaded into each consecutive LS loc-
                ;4163   ;       ation.  The first word listed will be bits 0-15 of the LS location, the
                ;4164   ;       second word listed will be bits 16-31 of the same LS location.
                ;4165   ;         The locations 78-7F in the first half of LS and F8-FF in the second
                ;4166   ;       half of LS are not actual LS locations.  They force an access to one of
                ;4167   ;       several registers in the CPU or force an indexing feature to another LS
                ;4168   ;       location.  For that reason, they are not written via the transfer rou-
                ;4169   ;       tine.
                ;4170   ;         Note:  This table is in hexadecimal format i.e. each digit represents
                ;4171   ;       four bits, so four digits represnts a full 16 bit word.  The micro-
                ;4172   ;       assembler requires that a hexidecimal value that starts with a letter
                ;4173   ;       (e.g. FFFF) be preceeded by a 0 (0FFFF).  So some table values will
                ;4174   ;       contain 5 hexidecimal digits.  The fifth digit is not actually used.
                ;4175   ;
                ;4176   ;---
                ;4177
                ;4178   TRANSFER.DATA.LS1:
                ;4179
U 0070, 0000,78 ;4180           COUNT.LS[0078]          ;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
                ;4181                                   ; FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0071, 0000,00 ;4182           START.ADR[0000]         ;DESTINATION START ADDRESS (START AT LS 0)
                ;4183
U 0072, 0000,00 ;4184           WORD[0000]              ;LOCATION 0
U 0073, 0000,00 ;4185           WORD[0000]
                ;4186
U 0074, 0000,00 ;4187           WORD[0000]              ;LOCATION 1
U 0075, 0000,00 ;4188           WORD[0000]
                ;4189
U 0076, 0000,00 ;4190           WORD[0000]              ;LOCATION 2
U 0077, 0000,00 ;4191           WORD[0000]
                ;4192
U 0078, 0000,00 ;4193           WORD[0000]              ;LOCATION 3
U 0079, 0000,00 ;4194           WORD[0000]
                ;4195
U 007A, 0000,00 ;4196           WORD[0000]              ;LOCATION 4
U 007B, 0000,00 ;4197           WORD[0000]
                ;4198
U 007C, 0000,00 ;4199           WORD[0000]              ;LOCATION 5
U 007D, 0000,00 ;4200           WORD[0000]
                ;4201
U 007E, 0000,00 ;4202           WORD[0000]              ;LOCATION 6
U 007F, 0000,00 ;4203           WORD[0000]
                ;4204
U 0080, 0000,00 ;4205           WORD[0000]              ;LOCATION 7
```

```
U 0081, 0000,00 ;4206           WORD[0000]
                ;4207
U 0082, 0000,00 ;4208           WORD[0000]              ;LOCATION 8
U 0083, 0000,00 ;4209           WORD[0000]
                ;4210
U 0084, 0000,00 ;4211           WORD[0000]              ;LOCATION 9
U 0085, 0000,00 ;4212           WORD[0000]
                ;4213
U 0086, 0000,00 ;4214           WORD[0000]              ;LOCATION 0A
U 0087, 0000,00 ;4215           WORD[0000]
                ;4216
U 0088, 0000,00 ;4217           WORD[0000]              ;LOCATION 0B
U 0089, 0000,00 ;4218           WORD[0000]
                ;4219
U 008A, 0000,00 ;4220           WORD[0000]              ;LOCATION 0C
U 008B, 0000,00 ;4221           WORD[0000]
                ;4222
U 008C, 0000,00 ;4223           WORD[0000]              ;LOCATION 0D
U 008D, 0000,00 ;4224           WORD[0000]
                ;4225
U 008E, 0000,00 ;4226           WORD[0000]              ;LOCATION 0E
U 008F, 0000,00 ;4227           WORD[0000]
                ;4228
U 0090, 0000,00 ;4229           WORD[0000]              ;LOCATION 0F
U 0091, 0000,00 ;4230           WORD[0000]
                ;4231
U 0092, 0000,00 ;4232           WORD[0000]              ;LOCATION 10
U 0093, 0000,00 ;4233           WORD[0000]
                ;4234
U 0094, 0000,00 ;4235           WORD[0000]              ;LOCATION 11
U 0095, 0000,00 ;4236           WORD[0000]
                ;4237
U 0096, 0000,00 ;4238           WORD[0000]              ;LOCATION 12
U 0097, 0000,00 ;4239           WORD[0000]
                ;4240
U 0098, 0000,FF ;4241           WORD[00FF]              ;LOCATION 13
U 0099, 0000,00 ;4242           WORD[0000]
                ;4243
U 009A, 00FF,FF ;4244           WORD[0FFFF]             ;LOCATION 14
U 009B, 0000,00 ;4245           WORD[0000]
                ;4246
U 009C, 0000,00 ;4247           WORD[0000]              ;LOCATION 15
U 009D, 00FF,00 ;4248           WORD[0FF00]
                ;4249
U 009E, 00FF,00 ;4250           WORD[0FF00]             ;LOCATION 16
U 009F, 00FF,FF ;4251           WORD[0FFFF]
                ;4252
U 00A0, 003F,FF ;4253           WORD[3FFF]              ;LOCATION 17
U 00A1, 0001,00 ;4254           WORD[0100]
                ;4255
U 00A2, 003F,FF ;4256           WORD[3FFF]              ;LOCATION 18
U 00A3, 007F,FE ;4257           WORD[7FFE]
                ;4258
U 00A4, 00FF,FF ;4259           WORD[0FFFF]             ;LOCATION 19
U 00A5, 00FE,7F ;4260           WORD[0FE7F]
```

```
                    ;4261
U 00A6, 0000,00     ;4262           WORD[0000]              ;LOCATION 1A
U 00A7, 007F,F8     ;4263           WORD[7FF8]
                    ;4264
U 00A8, 0080,00     ;4265           WORD[8000]              ;LOCATION 1B
U 00A9, 007F,FF     ;4266           WORD[7FFF]
                    ;4267
U 00AA, 0000,00     ;4268           WORD[0000]              ;LOCATION 1C
U 00AB, 0000,00     ;4269           WORD[0000]
                    ;4270
U 00AC, 0000,00     ;4271           WORD[0000]              ;LOCATION 1D
U 00AD, 0000,00     ;4272           WORD[0000]
                    ;4273
U 00AE, 0005,00     ;4274           WORD[0500]              ;LOCATION 1E
U 00AF, 0000,80     ;4275           WORD[0080]
                    ;4276
U 00B0, 0005,00     ;4277           WORD[0500]              ;LOCATION 1F
U 00B1, 0000,00     ;4278           WORD[0000]
                    ;4279
U 00B2, 0000,01     ;4280           WORD[0001]              ;LOCATION 20
U 00B3, 0000,00     ;4281           WORD[0000]
                    ;4282
U 00B4, 0000,02     ;4283           WORD[0002]              ;LOCATION 21
U 00B5, 0000,00     ;4284           WORD[0000]
                    ;4285
U 00B6, 0000,04     ;4286           WORD[0004]              ;LOCATION 22
U 00B7, 0000,00     ;4287           WORD[0000]
                    ;4288
U 00B8, 0000,08     ;4289           WORD[0008]              ;LOCATION 23
U 00B9, 0000,00     ;4290           WORD[0000]
                    ;4291
U 00BA, 0000,10     ;4292           WORD[0010]              ;LOCATION 24
U 00BB, 0000,00     ;4293           WORD[0000]
                    ;4294
U 00BC, 0000,20     ;4295           WORD[0020]              ;LOCATION 25
U 00BD, 0000,00     ;4296           WORD[0000]
                    ;4297
U 00BE, 0000,40     ;4298           WORD[0040]              ;LOCATION 26
U 00BF, 0000,00     ;4299           WORD[0000]
                    ;4300
U 00C0, 0000,80     ;4301           WORD[0080]              ;LOCATION 27
U 00C1, 0000,00     ;4302           WORD[0000]
                    ;4303
U 00C2, 0001,00     ;4304           WORD[0100]              ;LOCATION 28
U 00C3, 0000,00     ;4305           WORD[0000]
                    ;4306
U 00C4, 0002,00     ;4307           WORD[0200]              ;LOCATION 29
U 00C5, 0000,00     ;4308           WORD[0000]
                    ;4309
U 00C6, 0004,00     ;4310           WORD[0400]              ;LOCATION 2A
U 00C7, 0000,00     ;4311           WORD[0000]
                    ;4312
U 00C8, 0008,00     ;4313           WORD[0800]              ;LOCATION 2B
U 00C9, 0000,00     ;4314           WORD[0000]
                    ;4315
```

```
U 00CA, 0010,00 ;4316          WORD[1000]              ;LOCATION 2C
U 00CB, 0000,00 ;4317          WORD[0000]
                ;4318
U 00CC, 0020,00 ;4319          WORD[2000]              ;LOCATION 2D
U 00CD, 0000,00 ;4320          WORD[0000]
                ;4321
U 00CE, 0040,00 ;4322          WORD[4000]              ;LOCATION 2E
U 00CF, 0000,00 ;4323          WORD[0000]
                ;4324
U 00D0, 0080,00 ;4325          WORD[8000]              ;LOCATION 2F
U 00D1, 0000,00 ;4326          WORD[0000]
                ;4327
U 00D2, 0000,00 ;4328          WORD[0000]              ;LOCATION 30
U 00D3, 0000,01 ;4329          WORD[0001]
                ;4330
U 00D4, 0000,00 ;4331          WORD[0000]              ;LOCATION 31
U 00D5, 0000,02 ;4332          WORD[0002]
                ;4333
U 00D6, 0000,00 ;4334          WORD[0000]              ;LOCATION 32
U 00D7, 0000,04 ;4335          WORD[0004]
                ;4336
U 00D8, 0000,00 ;4337          WORD[0000]              ;LOCATION 33
U 00D9, 0000,08 ;4338          WORD[0008]
                ;4339
U 00DA, 0000,00 ;4340          WORD[0000]              ;LOCATION 34
U 00DB, 0000,10 ;4341          WORD[0010]
                ;4342
U 00DC, 0000,00 ;4343          WORD[0000]              ;LOCATION 35
U 00DD, 0000,20 ;4344          WORD[0020]
                ;4345
U 00DE, 0000,00 ;4346          WORD[0000]              ;LOCATION 36
U 00DF, 0000,40 ;4347          WORD[0040]
                ;4348
U 00E0, 0000,00 ;4349          WORD[0000]              ;LOCATION 37
U 00E1, 0000,80 ;4350          WORD[0080]
                ;4351
U 00E2, 0000,00 ;4352          WORD[0000]              ;LOCATION 38
U 00E3, 0001,00 ;4353          WORD[0100]
                ;4354
U 00E4, 0000,00 ;4355          WORD[0000]              ;LOCATION 39
U 00E5, 0002,00 ;4356          WORD[0200]
                ;4357
U 00E6, 0000,00 ;4358          WORD[0000]              ;LOCATION 3A
U 00E7, 0004,00 ;4359          WORD[0400]
                ;4360
U 00E8, 0000,00 ;4361          WORD[0000]              ;LOCATION 3B
U 00E9, 0008,00 ;4362          WORD[0800]
                ;4363
U 00EA, 0000,00 ;4364          WORD[0000]              ;LOCATION 3C
U 00EB, 0010,00 ;4365          WORD[1000]
                ;4366
U 00EC, 0000,00 ;4367          WORD[0000]              ;LOCATION 3D
U 00ED, 0020,00 ;4368          WORD[2000]
                ;4369
U 00EE, 0000,00 ;4370          WORD[0000]              ;LOCATION 3E
```

```
U 00EF, 0040,00 ;4371           WORD[4000]
                ;4372
U 00F0, 0000,00 ;4373           WORD[0000]              ;LOCATION 3F
U 00F1, 0080,00 ;4374           WORD[8000]
                ;4375
U 00F2, 0000,00 ;4376           WORD[0000]              ;LOCATION 40
U 00F3, 0000,00 ;4377           WORD[0000]
                ;4378
U 00F4, 0000,00 ;4379           WORD[0000]              ;LOCATION 41
U 00F5, 0000,00 ;4380           WORD[0000]
                ;4381
U 00F6, 0000,00 ;4382           WORD[0000]              ;LOCATION 42
U 00F7, 0000,00 ;4383           WORD[0000]
                ;4384
U 00F8, 0000,00 ;4385           WORD[0000]              ;LOCATION 43
U 00F9, 0000,00 ;4386           WORD[0000]
                ;4387
U 00FA, 0000,00 ;4388           WORD[0000]              ;LOCATION 44
U 00FB, 0000,00 ;4389           WORD[0000]
                ;4390
U 00FC, 0000,00 ;4391           WORD[0000]              ;LOCATION 45
U 00FD, 0000,00 ;4392           WORD[0000]
                ;4393
U 00FE, 0000,00 ;4394           WORD[0000]              ;LOCATION 46
U 00FF, 0000,00 ;4395           WORD[0000]
                ;4396
U 0100, 0000,00 ;4397           WORD[0000]              ;LOCATION 47
U 0101, 0000,00 ;4398           WORD[0000]
                ;4399
U 0102, 0000,00 ;4400           WORD[0000]              ;LOCATION 48
U 0103, 0000,00 ;4401           WORD[0000]
                ;4402
U 0104, 0000,00 ;4403           WORD[0000]              ;LOCATION 49
U 0105, 0000,00 ;4404           WORD[0000]
                ;4405
U 0106, 0000,00 ;4406           WORD[0000]              ;LOCATION 4A
U 0107, 0000,00 ;4407           WORD[0000]
                ;4408
U 0108, 0000,00 ;4409           WORD[0000]              ;LOCATION 4B
U 0109, 0000,00 ;4410           WORD[0000]
                ;4411
U 010A, 00AA,AA ;4412           WORD[0AAAA]             ;LOCATION 4C
U 010B, 00AA,AA ;4413           WORD[0AAAA]
                ;4414
U 010C, 0055,55 ;4415           WORD[5555]              ;LOCATION 4D
U 010D, 0055,55 ;4416           WORD[5555]
                ;4417
U 010E, 0000,00 ;4418           WORD[0000]              ;LOCATION 4E
U 010F, 0000,00 ;4419           WORD[0000]
                ;4420
U 0110, 00FF,FF ;4421           WORD[0FFFF]             ;LOCATION 4F
U 0111, 00FF,FF ;4422           WORD[0FFFF]
                ;4423
U 0112, 0000,00 ;4424           WORD[0000]              ;LOCATION 50
U 0113, 0000,00 ;4425           WORD[0000]
```

```
                ;4426
U 0114, 0000,00 ;4427           WORD[0000]              ;LOCATION 51
U 0115, 0000,00 ;4428           WORD[0000]
                ;4429
U 0116, 0000,00 ;4430           WORD[0000]              ;LOCATION 52
U 0117, 0000,00 ;4431           WORD[0000]
                ;4432
U 0118, 0000,00 ;4433           WORD[0000]              ;LOCATION 53
U 0119, 0000,00 ;4434           WORD[0000]
                ;4435
U 011A, 0000,00 ;4436           WORD[0000]              ;LOCATION 54
U 011B, 0000,00 ;4437           WORD[0000]
                ;4438
U 011C, 0000,00 ;4439           WORD[0000]              ;LOCATION 55
U 011D, 0000,00 ;4440           WORD[0000]
                ;4441
U 011E, 0000,00 ;4442           WORD[0000]              ;LOCATION 56
U 011F, 0000,00 ;4443           WORD[0000]
                ;4444
U 0120, 0000,00 ;4445           WORD[0000]              ;LOCATION 57
U 0121, 0000,00 ;4446           WORD[0000]
                ;4447
U 0122, 0000,00 ;4448           WORD[0000]              ;LOCATION 58
U 0123, 0000,00 ;4449           WORD[0000]
                ;4450
U 0124, 0000,00 ;4451           WORD[0000]              ;LOCATION 59
U 0125, 0000,00 ;4452           WORD[0000]
                ;4453
U 0126, 00F0,00 ;4454           WORD[0F000]             ;LOCATION 5A
U 0127, 00FF,FF ;4455           WORD[0FFFF]
                ;4456
U 0128, 00F0,02 ;4457           WORD[0F002]             ;LOCATION 5B
U 0129, 00FF,FF ;4458           WORD[0FFFF]
                ;4459
U 012A, 00F0,04 ;4460           WORD[0F004]             ;LOCATION 5C
U 012B, 00FF,FF ;4461           WORD[0FFFF]
                ;4462
U 012C, 00F0,06 ;4463           WORD[0F006]             ;LOCATION 5D
U 012D, 00FF,FF ;4464           WORD[0FFFF]
                ;4465
U 012E, 00F0,08 ;4466           WORD[0F008]             ;LOCATION 5E
U 012F, 00FF,FF ;4467           WORD[0FFFF]
                ;4468
U 0130, 00F0,0E ;4469           WORD[0F00E]             ;LOCATION 5F
U 0131, 00FF,FF ;4470           WORD[0FFFF]
                ;4471
U 0132, 0000,03 ;4472           WORD[0003]              ;LOCATION 60
U 0133, 0000,00 ;4473           WORD[0000]
                ;4474
U 0134, 0000,12 ;4475           WORD[0012]              ;LOCATION 61
U 0135, 0000,00 ;4476           WORD[0000]
                ;4477
U 0136, 0033,33 ;4478           WORD[3333]              ;LOCATION 62
U 0137, 0033,33 ;4479           WORD[3333]
                ;4480
```

```
U 0138, 000F,0F ;4481          WORD[0F0F]              ;LOCATION 63
U 0139, 000F,0F ;4482          WORD[0F0F]
                ;4483
U 013A, 0000,FF ;4484          WORD[00FF]              ;LOCATION 64
U 013B, 0000,FF ;4485          WORD[00FF]
                ;4486
U 013C, 0001,62 ;4487          WORD[0162]              ;LOCATION 65
U 013D, 0000,00 ;4488          WORD[0000]
                ;4489
U 013E, 0000,28 ;4490          WORD[0028]              ;LOCATION 66
U 013F, 0000,00 ;4491          WORD[0000]
                ;4492
U 0140, 0000,0C ;4493          WORD[000C]              ;LOCATION 67
U 0141, 0000,00 ;4494          WORD[0000]
                ;4495
U 0142, 0000,00 ;4496          WORD[0000]              ;LOCATION 68
U 0143, 0000,00 ;4497          WORD[0000]
                ;4498
U 0144, 0000,00 ;4499          WORD[0000]              ;LOCATION 69
U 0145, 0000,00 ;4500          WORD[0000]
                ;4501
U 0146, 0000,00 ;4502          WORD[0000]              ;LOCATION 6A
U 0147, 0000,00 ;4503          WORD[0000]
                ;4504
U 0148, 0000,00 ;4505          WORD[0000]              ;LOCATION 6B
U 0149, 0000,00 ;4506          WORD[0000]
                ;4507
U 014A, 0000,00 ;4508          WORD[0000]              ;LOCATION 6C
U 014B, 0000,00 ;4509          WORD[0000]
                ;4510
U 014C, 0000,00 ;4511          WORD[0000]              ;LOCATION 6D
U 014D, 0000,00 ;4512          WORD[0000]
                ;4513
U 014E, 0000,00 ;4514          WORD[0000]              ;LOCATION 6E
U 014F, 0000,00 ;4515          WORD[0000]
                ;4516
U 0150, 0000,00 ;4517          WORD[0000]              ;LOCATION 6F
U 0151, 0000,00 ;4518          WORD[0000]
                ;4519
U 0152, 0000,00 ;4520          WORD[0000]              ;LOCATION 70
U 0153, 0000,00 ;4521          WORD[0000]
                ;4522
U 0154, 0000,00 ;4523          WORD[0000]              ;LOCATION 71
U 0155, 0000,00 ;4524          WORD[0000]
                ;4525
U 0156, 0000,00 ;4526          WORD[0000]              ;LOCATION 72
U 0157, 0000,00 ;4527          WORD[0000]
                ;4528
U 0158, 0000,00 ;4529          WORD[0000]              ;LOCATION 73
U 0159, 0000,00 ;4530          WORD[0000]
                ;4531
U 015A, 0000,00 ;4532          WORD[0000]              ;LOCATION 74
U 015B, 0000,00 ;4533          WORD[0000]
                ;4534
U 015C, 0000,00 ;4535          WORD[0000]              ;LOCATION 75
```

```
U 015D, 0000,00 ;4536           WORD[0000]
                ;4537
U 015E, 0000,00 ;4538           WORD[0000]              ;LOCATION 76
U 015F, 0000,00 ;4539           WORD[0000]
                ;4540
U 0160, 0000,00 ;4541           WORD[0000]              ;LOCATION 77
U 0161, 0000,00 ;4542           WORD[0000]
                ;4543
                ;4544   .TOC    "PROGRAM SPECIFIC DATA SECTION (LS 80-F7)"
                ;4545   ;
                ;4546   ;       This section represents the second half of LS.  It is only acceseble
                ;4547   ;       with a MOV instruction, or one that uses the MOVE microinstruction for-
                ;4548   ;       mat.  This section will be loaded in a seperate transfer.
                ;4549   ;
                ;4550   TRANSFER.DATA.LS2:
U 0162, 0000,78 ;4551           COUNT.LS[0078]          ;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
                ;4552                                   ; FER TO LS = 120 DECIMAL) DESTINATION IS LS
U 0163, 0000,80 ;4553           START.ADR[0080]         ;DESTINATION START ADDRESS (START AT LS 80(H))
                ;4554
U 0164, 00FC,00 ;4555           WORD[0FC00]             ;LOCATION 80
U 0165, 00FF,FF ;4556           WORD[0FFFF]
                ;4557
U 0166, 0000,3A ;4558           WORD[003A]              ;LOCATION 81
U 0167, 0000,00 ;4559           WORD[0000]
                ;4560
U 0168, 0000,62 ;4561           WORD[0062]              ;LOCATION 82
U 0169, 0000,00 ;4562           WORD[0000]
                ;4563
U 016A, 00FF,F8 ;4564           WORD[0FFF8]             ;LOCATION 83
U 016B, 00FF,FF ;4565           WORD[0FFFF]
                ;4566
U 016C, 0000,00 ;4567           WORD[0000]              ;LOCATION 84
U 016D, 0000,00 ;4568           WORD[0000]
                ;4569
U 016E, 0002,54 ;4570           WORD[0254]              ;LOCATION 85
U 016F, 0000,00 ;4571           WORD[0000]
                ;4572
U 0170, 0000,64 ;4573           WORD[0064]              ;LOCATION 86
U 0171, 0000,00 ;4574           WORD[0000]
                ;4575
U 0172, 0000,68 ;4576           WORD[0068]              ;LOCATION 87
U 0173, 0000,00 ;4577           WORD[0000]
                ;4578
U 0174, 0000,6E ;4579           WORD[006E]              ;LOCATION 88
U 0175, 0000,00 ;4580           WORD[0000]
                ;4581
U 0176, 0000,05 ;4582           WORD[0005]              ;LOCATION 89
U 0177, 0000,00 ;4583           WORD[0000]
                ;4584
U 0178, 0000,06 ;4585           WORD[0006]              ;LOCATION 8A
U 0179, 0000,00 ;4586           WORD[0000]
                ;4587
U 017A, 0003,00 ;4588           WORD[0300]              ;LOCATION 8B
U 017B, 0000,00 ;4589           WORD[0000]
                ;4590
```

```
U 017C, 0003,00 ;4591           WORD[0300]              ;LOCATION 8C
U 017D, 0000,04 ;4592           WORD[0004]
                ;4593
U 017E, 0080,7F ;4594           WORD[0807F]             ;LOCATION 8D
U 017F, 00FF,FF ;4595           WORD[0FFFF]
                ;4596
U 0180, 0000,D6 ;4597           WORD[00D6]              ;LOCATION 8E
U 0181, 0000,00 ;4598           WORD[0000]
                ;4599
U 0182, 00BF,FF ;4600           WORD[0BFFF]             ;LOCATION 8F
U 0183, 00FF,FF ;4601           WORD[0FFFF]
                ;4602
U 0184, 0000,07 ;4603           WORD[0007]              ;LOCATION 90
U 0185, 0000,00 ;4604           WORD[0000]
                ;4605
U 0186, 0000,F8 ;4606           WORD[00F8]              ;LOCATION 91
U 0187, 0000,00 ;4607           WORD[0000]
                ;4608
U 0188, 0000,FF ;4609           WORD[000FF]             ;LOCATION 92
U 0189, 00FF,FF ;4610           WORD[0FFFF]
                ;4611
U 018A, 0001,1E ;4612           WORD[011E]              ;LOCATION 93
U 018B, 0000,00 ;4613           WORD[0000]
                ;4614
U 018C, 0000,09 ;4615           WORD[0009]              ;LOCATION 94
U 018D, 0000,00 ;4616           WORD[0000]
                ;4617
U 018E, 0001,44 ;4618           WORD[0144]              ;LOCATION 95
U 018F, 0000,00 ;4619           WORD[0000]
                ;4620
U 0190, 0000,1A ;4621           WORD[001A]              ;LOCATION 96
U 0191, 0000,00 ;4622           WORD[0000]
                ;4623
U 0192, 0002,A7 ;4624           WORD[02A7]              ;LOCATION 97
U 0193, 0000,00 ;4625           WORD[0000]
                ;4626
U 0194, 0002,DC ;4627           WORD[02DC]              ;LOCATION 98
U 0195, 0000,00 ;4628           WORD[0000]
                ;4629
U 0196, 0003,68 ;4630           WORD[0368]              ;LOCATION 99
U 0197, 0000,00 ;4631           WORD[0000]
                ;4632
U 0198, 0003,70 ;4633           WORD[0370]              ;LOCATION 9A
U 0199, 0000,00 ;4634           WORD[0000]
                ;4635
U 019A, 0002,DD ;4636           WORD[02DD]              ;LOCATION 9B
U 019B, 0000,00 ;4637           WORD[0000]
                ;4638
U 019C, 0001,80 ;4639           WORD[0180]              ;LOCATION 9C
U 019D, 0000,00 ;4640           WORD[0000]
                ;4641
U 019E, 0001,84 ;4642           WORD[0184]              ;LOCATION 9D
U 019F, 0000,00 ;4643           WORD[0000]
                ;4644
U 01A0, 0001,8A ;4645           WORD[018A]              ;LOCATION 9E
```

```
U 01A1, 0000,00 ;4646           WORD[0000]
                ;4647
U 01A2, 0003,01 ;4648           WORD[0301]              ;LOCATION 9F
U 01A3, 0000,00 ;4649           WORD[0000]
                ;4650
U 01A4, 0003,02 ;4651           WORD[0302]              ;LOCATION 0A0
U 01A5, 0000,00 ;4652           WORD[0000]
                ;4653
U 01A6, 0003,03 ;4654           WORD[0303]              ;LOCATION 0A1
U 01A7, 0000,00 ;4655           WORD[0000]
                ;4656
U 01A8, 0003,00 ;4657           WORD[0300]              ;LOCATION 0A2
U 01A9, 0000,06 ;4658           WORD[0006]
                ;4659
U 01AA, 0001,94 ;4660           WORD[0194]              ;LOCATION 0A3
U 01AB, 0000,00 ;4661           WORD[0000]
                ;4662
U 01AC, 0003,00 ;4663           WORD[0300]              ;LOCATION 0A4
U 01AD, 0000,07 ;4664           WORD[0007]
                ;4665
U 01AE, 0001,9C ;4666           WORD[019C]              ;LOCATION 0A5
U 01AF, 0000,00 ;4667           WORD[0000]
                ;4668
U 01B0, 00FF,FF ;4669           WORD[0FFFF]             ;LOCATION 0A6
U 01B1, 00BF,FF ;4670           WORD[0BFFF]
                ;4671
U 01B2, 0001,9E ;4672           WORD[019E]              ;LOCATION 0A7
U 01B3, 0000,00 ;4673           WORD[0000]
                ;4674
U 01B4, 00F0,00 ;4675           WORD[0F000]             ;LOCATION 0A8
U 01B5, 0000,01 ;4676           WORD[0001]
                ;4677
U 01B6, 0080,00 ;4678           WORD[8000]              ;LOCATION 0A9
U 01B7, 0000,01 ;4679           WORD[0001]
                ;4680
U 01B8, 0043,00 ;4681           WORD[4300]              ;LOCATION 0AA
U 01B9, 0000,04 ;4682           WORD[0004]
                ;4683
U 01BA, 0007,00 ;4684           WORD[0700]              ;LOCATION 0AB
U 01BB, 0000,04 ;4685           WORD[0004]
                ;4686
U 01BC, 000B,00 ;4687           WORD[0B00]              ;LOCATION 0AC
U 01BD, 0000,04 ;4688           WORD[0004]
                ;4689
U 01BE, 000F,00 ;4690           WORD[0F00]              ;LOCATION 0AD
U 01BF, 0000,04 ;4691           WORD[0004]
                ;4692
U 01C0, 0013,00 ;4693           WORD[1300]              ;LOCATION 0AE
U 01C1, 0000,04 ;4694           WORD[0004]
                ;4695
U 01C2, 0001,A6 ;4696           WORD[01A6]              ;LOCATION 0AF
U 01C3, 0000,00 ;4697           WORD[0000]
                ;4698
U 01C4, 0001,A8 ;4699           WORD[01A8]              ;LOCATION 0B0
U 01C5, 0000,00 ;4700           WORD[0000]
```

```
                ;4701
U 01C6, 0023,00 ;4702           WORD[2300]              ;LOCATION 0B1
U 01C7, 0000,04 ;4703           WORD[0004]
                ;4704
U 01C8, 0017,00 ;4705           WORD[1700]              ;LOCATION 0B2
U 01C9, 0000,04 ;4706           WORD[0004]
                ;4707
U 01CA, 001B,00 ;4708           WORD[1B00]              ;LOCATION 0B3
U 01CB, 0000,04 ;4709           WORD[0004]
                ;4710
U 01CC, 0001,AE ;4711           WORD[01AE]              ;LOCATION 0B4
U 01CD, 0000,00 ;4712           WORD[0000]
                ;4713
U 01CE, 005F,00 ;4714           WORD[5F00]              ;LOCATION 0B5
U 01CF, 0000,04 ;4715           WORD[0004]
                ;4716
U 01D0, 003F,00 ;4717           WORD[3F00]              ;LOCATION 0B6
U 01D1, 0000,04 ;4718           WORD[0004]
                ;4719
U 01D2, 001F,00 ;4720           WORD[1F00]              ;LOCATION 0B7
U 01D3, 0000,04 ;4721           WORD[0004]
                ;4722
U 01D4, 0001,B2 ;4723           WORD[01B2]              ;LOCATION 0B8
U 01D5, 0000,00 ;4724           WORD[0000]
                ;4725
U 01D6, 0001,B8 ;4726           WORD[01B8]              ;LOCATION 0B9
U 01D7, 0000,00 ;4727           WORD[0000]
                ;4728
U 01D8, 0001,BC ;4729           WORD[01BC]              ;LOCATION 0BA
U 01D9, 0000,00 ;4730           WORD[0000]
                ;4731
U 01DA, 0083,00 ;4732           WORD[8300]              ;LOCATION 0BB
U 01DB, 0000,00 ;4733           WORD[0000]
                ;4734
U 01DC, 0001,C4 ;4735           WORD[01C4]              ;LOCATION 0BC
U 01DD, 0000,00 ;4736           WORD[0000]
                ;4737
U 01DE, 0001,CE ;4738           WORD[01CE]              ;LOCATION 0BD
U 01DF, 0000,00 ;4739           WORD[0000]
                ;4740
U 01E0, 0001,D6 ;4741           WORD[01D6]              ;LOCATION 0BE
U 01E1, 0000,00 ;4742           WORD[0000]
                ;4743
U 01E2, 0002,60 ;4744           WORD[0260]              ;LOCATION 0BF
U 01E3, 0000,00 ;4745           WORD[0000]
                ;4746
U 01E4, 0002,64 ;4747           WORD[0264]              ;LOCATION 0C0
U 01E5, 0000,00 ;4748           WORD[0000]
                ;4749
U 01E6, 0002,6C ;4750           WORD[026C]              ;LOCATION 0C1
U 01E7, 0000,00 ;4751           WORD[0000]
                ;4752
U 01E8, 0002,76 ;4753           WORD[0276]              ;LOCATION 0C2
U 01E9, 0000,00 ;4754           WORD[0000]
                ;4755
```

```
U 01EA, 0002,7E ;4756          WORD[027E]              ;LOCATION 0C3
U 01EB, 0000,00 ;4757          WORD[0000]
                ;4758
U 01EC, 0002,84 ;4759          WORD[0284]              ;LOCATION 0C4
U 01ED, 0000,00 ;4760          WORD[0000]
                ;4761
U 01EE, 0002,8A ;4762          WORD[028A]              ;LOCATION 0C5
U 01EF, 0000,00 ;4763          WORD[0000]
                ;4764
U 01F0, 0000,00 ;4765          WORD[0000]              ;LOCATION 0C6
U 01F1, 0000,00 ;4766          WORD[0000]
                ;4767
U 01F2, 0000,62 ;4768          WORD[0062]              ;LOCATION 0C7
U 01F3, 0000,00 ;4769          WORD[0000]
                ;4770
U 01F4, 0000,00 ;4771          WORD[0000]              ;LOCATION 0C8
U 01F5, 0000,00 ;4772          WORD[0000]
                ;4773
U 01F6, 0000,00 ;4774          WORD[0000]              ;LOCATION 0C9
U 01F7, 0000,00 ;4775          WORD[0000]
                ;4776
U 01F8, 0002,A0 ;4777          WORD[02A0]              ;LOCATION 0CA
U 01F9, 0000,00 ;4778          WORD[0000]
                ;4779
U 01FA, 0002,B2 ;4780          WORD[02B2]              ;LOCATION 0CB
U 01FB, 0000,00 ;4781          WORD[0000]
                ;4782
U 01FC, 0002,CC ;4783          WORD[02CC]              ;LOCATION 0CC
U 01FD, 0000,00 ;4784          WORD[0000]
                ;4785
U 01FE, 0000,00 ;4786          WORD[0000]              ;LOCATION 0CD
U 01FF, 0000,00 ;4787          WORD[0000]
                ;4788
U 0200, 0002,DC ;4789          WORD[02DC]              ;LOCATION 0CE
U 0201, 0000,00 ;4790          WORD[0000]
                ;4791
U 0202, 0002,DE ;4792          WORD[02DE]              ;LOCATION 0CF
U 0203, 0000,00 ;4793          WORD[0000]
                ;4794
U 0204, 0003,38 ;4795          WORD[0338]              ;LOCATION 0D0
U 0205, 0000,00 ;4796          WORD[0000]
                ;4797
U 0206, 0003,78 ;4798          WORD[0378]              ;LOCATION 0D1
U 0207, 0000,00 ;4799          WORD[0000]
                ;4800
U 0208, 0003,BA ;4801          WORD[03BA]              ;LOCATION 0D2
U 0209, 0000,00 ;4802          WORD[0000]
                ;4803
U 020A, 0003,BC ;4804          WORD[03BC]              ;LOCATION 0D3
U 020B, 0000,00 ;4805          WORD[0000]
                ;4806
U 020C, 0003,C6 ;4807          WORD[03C6]              ;LOCATION 0D4
U 020D, 0000,00 ;4808          WORD[0000]
                ;4809
U 020E, 0003,DA ;4810          WORD[03DA]              ;LOCATION 0D5
```

```
U 020F, 0000,00 ;4811           WORD[0000]
                ;4812
U 0210, 0003,FA ;4813           WORD[03FA]              ;LOCATION 0D6
U 0211, 0000,00 ;4814           WORD[0000]
                ;4815
U 0212, 0004,4E ;4816           WORD[044E]              ;LOCATION 0D7
U 0213, 0000,00 ;4817           WORD[0000]
                ;4818
U 0214, 0004,5E ;4819           WORD[045E]              ;LOCATION 0D8
U 0215, 0000,00 ;4820           WORD[0000]
                ;4821
U 0216, 0004,70 ;4822           WORD[0470]              ;LOCATION 0D9
U 0217, 0000,00 ;4823           WORD[0000]
                ;4824
U 0218, 0004,88 ;4825           WORD[0488]              ;LOCATION 0DA
U 0219, 0000,00 ;4826           WORD[0000]
                ;4827
U 021A, 0005,40 ;4828           WORD[0540]              ;LOCATION 0DB
U 021B, 0000,00 ;4829           WORD[0000]
                ;4830
U 021C, 0000,00 ;4831           WORD[0000]              ;LOCATION 0DC
U 021D, 0000,00 ;4832           WORD[0000]
                ;4833
U 021E, 0001,1D ;4834           WORD[011D]              ;LOCATION 0DD
U 021F, 0000,00 ;4835           WORD[0000]
                ;4836
U 0220, 0003,BA ;4837           WORD[03BA]              ;LOCATION 0DE
U 0221, 0000,00 ;4838           WORD[0000]
                ;4839
U 0222, 0003,BC ;4840           WORD[03BC]              ;LOCATION 0DF
U 0223, 0000,00 ;4841           WORD[0000]
                ;4842
U 0224, 0001,E0 ;4843           WORD[01E0]              ;LOCATION 0E0
U 0225, 0000,00 ;4844           WORD[0000]
                ;4845
U 0226, 0001,E6 ;4846           WORD[01E6]              ;LOCATION 0E1
U 0227, 0000,00 ;4847           WORD[0000]
                ;4848
U 0228, 0001,F0 ;4849           WORD[01F0]              ;LOCATION 0E2
U 0229, 0000,00 ;4850           WORD[0000]
                ;4851
U 022A, 0002,00 ;4852           WORD[0200]              ;LOCATION 0E3
U 022B, 0000,00 ;4853           WORD[0000]
                ;4854
U 022C, 0002,12 ;4855           WORD[0212]              ;LOCATION 0E4
U 022D, 0000,00 ;4856           WORD[0000]
                ;4857
U 022E, 0002,1C ;4858           WORD[021C]              ;LOCATION 0E5
U 022F, 0000,00 ;4859           WORD[0000]
                ;4860
U 0230, 0002,22 ;4861           WORD[0222]              ;LOCATION 0E6
U 0231, 0000,00 ;4862           WORD[0000]
                ;4863
U 0232, 0002,28 ;4864           WORD[0228]              ;LOCATION 0E7
U 0233, 0000,00 ;4865           WORD[0000]
```

```
                    ;4866
U 0234, 0002,2E     ;4867           WORD[022E]              ;LOCATION 0E8
U 0235, 0000,00     ;4868           WORD[0000]
                    ;4869
U 0236, 0002,34     ;4870           WORD[0234]              ;LOCATION 0E9
U 0237, 0000,00     ;4871           WORD[0000]
                    ;4872
U 0238, 0002,3E     ;4873           WORD[023E]              ;LOCATION 0EA
U 0239, 0000,00     ;4874           WORD[0000]
                    ;4875
U 023A, 0002,42     ;4876           WORD[0242]              ;LOCATION 0EB
U 023B, 0000,00     ;4877           WORD[0000]
                    ;4878
U 023C, 0002,48     ;4879           WORD[0248]              ;LOCATION 0EC
U 023D, 0000,00     ;4880           WORD[0000]
                    ;4881
U 023E, 0002,4E     ;4882           WORD[024E]              ;LOCATION 0ED
U 023F, 0000,00     ;4883           WORD[0000]
                    ;4884
U 0240, 0002,54     ;4885           WORD[0254]              ;LOCATION 0EE
U 0241, 0000,00     ;4886           WORD[0000]
                    ;4887
U 0242, 0002,5E     ;4888           WORD[025E]              ;LOCATION 0EF
U 0243, 0000,00     ;4889           WORD[0000]
                    ;4890
U 0244, 0000,00     ;4891           WORD[0000]              ;LOCATION 0F0
U 0245, 0000,00     ;4892           WORD[0000]
                    ;4893
U 0246, 0000,00     ;4894           WORD[0000]              ;LOCATION 0F1
U 0247, 0000,00     ;4895           WORD[0000]
                    ;4896
U 0248, 0000,00     ;4897           WORD[0000]              ;LOCATION 0F2
U 0249, 0000,00     ;4898           WORD[0000]
                    ;4899
U 024A, 0000,00     ;4900           WORD[0000]              ;LOCATION 0F3
U 024B, 0000,00     ;4901           WORD[0000]
                    ;4902
U 024C, 0000,00     ;4903           WORD[0000]              ;LOCATION 0F4
U 024D, 0000,00     ;4904           WORD[0000]
                    ;4905
U 024E, 0000,83     ;4906           WORD[0083]              ;LOCATION 0F5
U 024F, 0000,00     ;4907           WORD[0000]
                    ;4908
U 0250, 0003,42     ;4909           WORD[0342]              ;LOCATION 0F6
U 0251, 0000,00     ;4910           WORD[0000]
                    ;4911
U 0252, 0000,8D     ;4912           WORD[008D]              ;LOCATION 0F7
U 0253, 0000,00     ;4913           WORD[0000]
                    ;4914
```

```
                ;4915   .PAGE
                ;4916   .TOC    'MAIN MEMORY DATA SECTION''
                ;4917   ;+++
                ;4918   ;       This section lists the data that will be transfered to main memory by
                ;4919   ;       the console as part of the initialization performed in test 0.  The
                ;4920   ;       TRANSFER DATA routine will point to this data area.  Main memory con-
                ;4921   ;       tains the addresses of FPA u-words used for the u-diagnostics, as well
                ;4922   ;       as other data used by the tests.
                ;4923   ;         Note:  This table is in hexadecimal format i.e. each digit represents
                ;4924   ;       four bits, so four digits represnts a full 16 bit word.  The micro-
                ;4925   ;       assembler requires that a hexidecimal value that starts with a letter
                ;4926   ;       (e.g. FFFF) be preceeded by a 0 (0FFFF).  So some table values will
                ;4927   ;       contain 5 hexidecimal digits.  The fifth digit is not actually used.
                ;4928   ;
                ;4929   ;---
                ;4930
                ;4931
U 0254, 0101,C0 ;4932           COUNT.MM[1C0]           ;LONGWORD COUNT (NUMBER OF LONGWORDS TO TRANS-
                ;4933                                   ; FER TO MEMORY) DESTINATION IS MAIN MEMORY
                ;4934                                   ;  DECIMAL LONGWORDS
U 0255, 0000,00 ;4935           START.ADR[0000]         ;DESTINATION START ADDRESS (START AT MAIN
                ;4936                                   ; MEMORY ADDRESS 0)
                ;4937
U 0256, 0000,10 ;4938           WORD[0010]              ;LOCATION 0
U 0257, 0000,01 ;4939           WORD[0001]
                ;4940
U 0258, 0000,0E ;4941           WORD[000E]              ;LOCATION 4
U 0259, 0000,40 ;4942           WORD[0040]
                ;4943
U 025A, 0000,AB ;4944           WORD[00AB]              ;LOCATION 8
U 025B, 0000,14 ;4945           WORD[0014]
                ;4946
U 025C, 0000,82 ;4947           WORD[0082]              ;LOCATION A
U 025D, 0000,08 ;4948           WORD[0008]
                ;4949
U 025E, 0000,AC ;4950           WORD[00AC]              ;LOCATION 10
U 025F, 0000,3C ;4951           WORD[003C]
                ;4952
U 0260, 0000,0E ;4953           WORD[000E]              ;LOCATION 14
U 0261, 0000,00 ;4954           WORD[0000]
                ;4955
U 0262, 0000,01 ;4956           WORD[0001]              ;LOCATION 18
U 0263, 0000,34 ;4957           WORD[0034]
                ;4958
U 0264, 0000,09 ;4959           WORD[0009]              ;LOCATION 1C
U 0265, 0000,20 ;4960           WORD[0020]
                ;4961
U 0266, 0000,AB ;4962           WORD[00AB]              ;LOCATION 20
U 0267, 0000,38 ;4963           WORD[0038]
                ;4964
U 0268, 0000,22 ;4965           WORD[0022]              ;LOCATION 24
U 0269, 0002,AD ;4966           WORD[02AD]
                ;4967
U 026A, 0000,30 ;4968           WORD[0030]              ;LOCATION 28
U 026B, 0000,9B ;4969           WORD[009B]
```

```
                ;4970
U 026C, 0000,24 ;4971           WORD[0024]              ;LOCATION 2C
U 026D, 0000,22 ;4972           WORD[0022]
                ;4973
U 026E, 0000,2C ;4974           WORD[002C]              ;LOCATION 30
U 026F, 0000,7F ;4975           WORD[007F]
                ;4976
U 0270, 0000,81 ;4977           WORD[0081]              ;LOCATION 34
U 0271, 0000,0E ;4978           WORD[000E]
                ;4979
U 0272, 0000,38 ;4980           WORD[0038]              ;LOCATION 38
U 0273, 0000,07 ;4981           WORD[0007]
                ;4982
U 0274, 0002,B2 ;4983           WORD[02B2]              ;LOCATION 3C
U 0275, 0000,07 ;4984           WORD[0007]
                ;4985
U 0276, 0002,B4 ;4986           WORD[02B4]              ;LOCATION 40
U 0277, 0000,07 ;4987           WORD[0007]
                ;4988
U 0278, 0002,B6 ;4989           WORD[02B6]              ;LOCATION 44
U 0279, 0000,07 ;4990           WORD[0007]
                ;4991
U 027A, 0002,B8 ;4992           WORD[02B8]              ;LOCATION 48
U 027B, 0000,07 ;4993           WORD[0007]
                ;4994
U 027C, 0002,BC ;4995           WORD[02BC]              ;LOCATION 4C
U 027D, 0000,07 ;4996           WORD[0007]
                ;4997
U 027E, 0002,BD ;4998           WORD[02BD]              ;LOCATION 50
U 027F, 0000,07 ;4999           WORD[0007]
                ;5000
U 0280, 0002,C0 ;5001           WORD[02C0]              ;LOCATION 54
U 0281, 0000,07 ;5002           WORD[0007]
                ;5003
U 0282, 0002,C4 ;5004           WORD[02C4]              ;LOCATION 58
U 0283, 0000,05 ;5005           WORD[0005]
                ;5006
U 0284, 0002,C5 ;5007           WORD[02C5]              ;LOCATION 5C
U 0285, 0000,03 ;5008           WORD[0003]
                ;5009
U 0286, 0002,C9 ;5010           WORD[02C9]              ;LOCATION 60
U 0287, 0002,99 ;5011           WORD[0299]
                ;5012
U 0288, 0002,98 ;5013           WORD[0298]              ;LOCATION 64
U 0289, 0002,99 ;5014           WORD[0299]
                ;5015
U 028A, 0003,02 ;5016           WORD[0302]              ;LOCATION 68
U 028B, 0003,90 ;5017           WORD[0390]
                ;5018
U 028C, 0003,00 ;5019           WORD[0300]              ;LOCATION 6C
U 028D, 0002,A7 ;5020           WORD[02A7]
                ;5021
U 028E, 0000,59 ;5022           WORD[0059]              ;LOCATION 70
U 028F, 0000,6E ;5023           WORD[006E]
                ;5024
```

```
U 0290, 0003,A8 ;5025          WORD[03A8]              ;LOCATION 74
U 0291, 0003,98 ;5026          WORD[0398]
                ;5027
U 0292, 0003,A0 ;5028          WORD[03A0]              ;LOCATION 78
U 0293, 0001,68 ;5029          WORD[0168]
                ;5030
U 0294, 0003,68 ;5031          WORD[0368]              ;LOCATION 7C
U 0295, 0003,70 ;5032          WORD[0370]
                ;5033
U 0296, 0003,B0 ;5034          WORD[03B0]              ;LOCATION 80
U 0297, 0003,6A ;5035          WORD[036A]
                ;5036
U 0298, 0003,72 ;5037          WORD[0372]              ;LOCATION 84
U 0299, 0003,7A ;5038          WORD[037A]
                ;5039
U 029A, 0003,82 ;5040          WORD[0382]              ;LOCATION 88
U 029B, 0003,8A ;5041          WORD[038A]
                ;5042
U 029C, 0003,92 ;5043          WORD[0392]              ;LOCATION 8C
U 029D, 0003,9A ;5044          WORD[039A]
                ;5045
U 029E, 0003,A2 ;5046          WORD[03A2]              ;LOCATION 90
U 029F, 0003,AA ;5047          WORD[03AA]
                ;5048
U 02A0, 0003,B2 ;5049          WORD[03B2]              ;LOCATION 94
U 02A1, 0003,6C ;5050          WORD[036C]
                ;5051
U 02A2, 0002,E5 ;5052          WORD[02E5]              ;LOCATION 98
U 02A3, 0002,E8 ;5053          WORD[02E8]
                ;5054
U 02A4, 0002,E9 ;5055          WORD[02E9]              ;LOCATION 9C
U 02A5, 0003,88 ;5056          WORD[0388]
                ;5057
U 02A6, 0003,78 ;5058          WORD[0378]              ;LOCATION 0A0
U 02A7, 0003,80 ;5059          WORD[0380]
                ;5060
U 02A8, 0002,EC ;5061          WORD[02EC]              ;LOCATION 0A4
U 02A9, 0002,ED ;5062          WORD[02ED]
                ;5063
U 02AA, 0002,F0 ;5064          WORD[02F0]              ;LOCATION 0A8
U 02AB, 0002,F1 ;5065          WORD[02F1]
                ;5066
U 02AC, 0002,F4 ;5067          WORD[02F4]              ;LOCATION 0AC
U 02AD, 0002,F5 ;5068          WORD[02F5]
                ;5069
U 02AE, 0002,F8 ;5070          WORD[02F8]              ;LOCATION 0B0
U 02AF, 0002,FA ;5071          WORD[02FA]
                ;5072
U 02B0, 0003,28 ;5073          WORD[0328]              ;LOCATION 0B4
U 02B1, 0003,2A ;5074          WORD[032A]
                ;5075
U 02B2, 0003,2C ;5076          WORD[032C]              ;LOCATION 0B8
U 02B3, 0003,2D ;5077          WORD[032D]
                ;5078
U 02B4, 0003,30 ;5079          WORD[0330]              ;LOCATION 0BC
```

```
U 02B5, 0003,32 ;5080          WORD[0332]
                ;5081
U 02B6, 0003,34 ;5082          WORD[0334]              ;LOCATION 0C0
U 02B7, 0003,36 ;5083          WORD[0336]
                ;5084
U 02B8, 0003,38 ;5085          WORD[0338]              ;LOCATION 0C4
U 02B9, 0003,3A ;5086          WORD[033A]
                ;5087
U 02BA, 0003,3C ;5088          WORD[033C]              ;LOCATION 0C8
U 02BB, 0003,3E ;5089          WORD[033E]
                ;5090
U 02BC, 0003,40 ;5091          WORD[0340]              ;LOCATION 0CC
U 02BD, 0000,93 ;5092          WORD[0093]
                ;5093
U 02BE, 0003,48 ;5094          WORD[0348]              ;LOCATION 0D0
U 02BF, 0003,98 ;5095          WORD[0398]
                ;5096
U 02C0, 0003,A0 ;5097          WORD[03A0]              ;LOCATION 0D4
U 02C1, 0002,A7 ;5098          WORD[02A7]
                ;5099
U 02C2, 0003,42 ;5100          WORD[0342]              ;LOCATION 0D8
U 02C3, 0000,59 ;5101          WORD[0059]
                ;5102
U 02C4, 0000,83 ;5103          WORD[0083]              ;LOCATION 0DC
U 02C5, 0000,4F ;5104          WORD[004F]
                ;5105
U 02C6, 0003,45 ;5106          WORD[0345]              ;LOCATION 0E0
U 02C7, 0001,68 ;5107          WORD[0168]
                ;5108
U 02C8, 0003,68 ;5109          WORD[0368]              ;LOCATION 0E4
U 02C9, 0000,7E ;5110          WORD[007E]
                ;5111
U 02CA, 0003,4A ;5112          WORD[034A]              ;LOCATION 0E8
U 02CB, 0003,AA ;5113          WORD[03AA]
                ;5114
U 02CC, 0003,B2 ;5115          WORD[03B2]              ;LOCATION 0EC
U 02CD, 0000,4F ;5116          WORD[004F]
                ;5117
U 02CE, 0003,4A ;5118          WORD[034A]              ;LOCATION 0F0
U 02CF, 0000,93 ;5119          WORD[0093]
                ;5120
U 02D0, 0003,98 ;5121          WORD[0398]              ;LOCATION 0F4
U 02D1, 0003,48 ;5122          WORD[0348]
                ;5123
U 02D2, 0002,DC ;5124          WORD[02DC]              ;LOCATION 0F8
U 02D3, 0000,65 ;5125          WORD[0065]
                ;5126
U 02D4, 0002,DD ;5127          WORD[02DD]              ;LOCATION 0FC
U 02D5, 0001,68 ;5128          WORD[0168]
                ;5129
U 02D6, 0002,E4 ;5130          WORD[02E4]              ;LOCATION 100
U 02D7, 0003,7A ;5131          WORD[037A]
                ;5132
U 02D8, 0003,50 ;5133          WORD[0350]              ;LOCATION 104
U 02D9, 0003,AA ;5134          WORD[03AA]
```

```
                ;5135
U 02DA, 0003,55 ;5136           WORD[0355]              ;LOCATION 108
U 02DB, 0003,88 ;5137           WORD[0388]
                ;5138
U 02DC, 0003,54 ;5139           WORD[0354]              ;LOCATION 10C
U 02DD, 0002,F1 ;5140           WORD[02F1]
                ;5141
U 02DE, 0003,58 ;5142           WORD[0358]              ;LOCATION 110
U 02DF, 0003,2A ;5143           WORD[032A]
                ;5144
U 02E0, 0003,59 ;5145           WORD[0359]              ;LOCATION 114
U 02E1, 0003,36 ;5146           WORD[0336]
                ;5147
U 02E2, 0002,DD ;5148           WORD[02DD]              ;LOCATION 118
U 02E3, 0000,93 ;5149           WORD[0093]
                ;5150
U 02E4, 0003,47 ;5151           WORD[0347]              ;LOCATION 11C
U 02E5, 0002,DC ;5152           WORD[02DC]
                ;5153
U 02E6, 0003,47 ;5154           WORD[0347]              ;LOCATION 120
U 02E7, 0002,DD ;5155           WORD[02DD]
                ;5156
U 02E8, 0003,A8 ;5157           WORD[03A8]              ;LOCATION 124
U 02E9, 0001,22 ;5158           WORD[0122]
                ;5159
U 02EA, 0003,B0 ;5160           WORD[03B0]              ;LOCATION 128
U 02EB, 0000,11 ;5161           WORD[0011]
                ;5162
U 02EC, 0003,92 ;5163           WORD[0392]              ;LOCATION 12C
U 02ED, 0001,95 ;5164           WORD[0195]
                ;5165
U 02EE, 0002,E5 ;5166           WORD[02E5]              ;LOCATION 130
U 02EF, 0003,51 ;5167           WORD[0351]
                ;5168
U 02F0, 0002,EC ;5169           WORD[02EC]              ;LOCATION 134
U 02F1, 0001,8D ;5170           WORD[018D]
                ;5171
U 02F2, 0002,F8 ;5172           WORD[02F8]              ;LOCATION 138
U 02F3, 0003,5C ;5173           WORD[035C]
                ;5174
U 02F4, 0003,30 ;5175           WORD[0330]              ;LOCATION 13C
U 02F5, 0003,60 ;5176           WORD[0360]
                ;5177
U 02F6, 0003,3C ;5178           WORD[033C]              ;LOCATION 140
U 02F7, 0003,62 ;5179           WORD[0362]
                ;5180
U 02F8, 0002,A7 ;5181           WORD[02A7]              ;LOCATION 144
U 02F9, 0000,4F ;5182           WORD[004F]
                ;5183
U 02FA, 0003,4A ;5184           WORD[034A]              ;LOCATION 148
U 02FB, 0003,B2 ;5185           WORD[03B2]
                ;5186
U 02FC, 0003,A8 ;5187           WORD[03A8]              ;LOCATION 14C
U 02FD, 0003,46 ;5188           WORD[0346]
                ;5189
```

```
U 02FE, 0003,B0 ;5190          WORD[03B0]          ;LOCATION 150
U 02FF, 0003,5E ;5191          WORD[035E]
                ;5192
U 0300, 0003,7A ;5193          WORD[037A]          ;LOCATION 154
U 0301, 0002,C8 ;5194          WORD[02C8]
                ;5195
U 0302, 0003,92 ;5196          WORD[0392]          ;LOCATION 158
U 0303, 0003,64 ;5197          WORD[0364]
                ;5198
U 0304, 0002,E5 ;5199          WORD[02E5]          ;LOCATION 15C
U 0305, 0003,6D ;5200          WORD[036D]
                ;5201
U 0306, 0003,88 ;5202          WORD[0388]          ;LOCATION 160
U 0307, 0000,43 ;5203          WORD[0043]
                ;5204
U 0308, 0002,EC ;5205          WORD[02EC]          ;LOCATION 164
U 0309, 0003,74 ;5206          WORD[0374]
                ;5207
U 030A, 0002,F1 ;5208          WORD[02F1]          ;LOCATION 168
U 030B, 0003,75 ;5209          WORD[0375]
                ;5210
U 030C, 0002,F8 ;5211          WORD[02F8]          ;LOCATION 16C
U 030D, 0003,7C ;5212          WORD[037C]
                ;5213
U 030E, 0003,2A ;5214          WORD[032A]          ;LOCATION 170
U 030F, 0003,7E ;5215          WORD[037E]
                ;5216
U 0310, 0003,30 ;5217          WORD[0330]          ;LOCATION 174
U 0311, 0003,84 ;5218          WORD[0384]
                ;5219
U 0312, 0003,36 ;5220          WORD[0336]          ;LOCATION 178
U 0313, 0002,72 ;5221          WORD[0272]
                ;5222
U 0314, 0003,3C ;5223          WORD[033C]          ;LOCATION 17C
U 0315, 0003,8E ;5224          WORD[038E]
                ;5225
U 0316, 0000,93 ;5226          WORD[0093]          ;LOCATION 180
U 0317, 0002,54 ;5227          WORD[0254]
                ;5228
U 0318, 0001,68 ;5229          WORD[0168]          ;LOCATION 184
U 0319, 0002,E5 ;5230          WORD[02E5]
                ;5231
U 031A, 0000,1F ;5232          WORD[001F]          ;LOCATION 188
U 031B, 0002,A7 ;5233          WORD[02A7]
                ;5234
U 031C, 0003,42 ;5235          WORD[0342]          ;LOCATION 18C
U 031D, 0001,68 ;5236          WORD[0168]
                ;5237
U 031E, 0002,40 ;5238          WORD[0240]          ;LOCATION 190
U 031F, 0003,94 ;5239          WORD[0394]
                ;5240
U 0320, 0001,7C ;5241          WORD[017C]          ;LOCATION 194
U 0321, 0003,96 ;5242          WORD[0396]
                ;5243
U 0322, 0000,B9 ;5244          WORD[00B9]          ;LOCATION 198
```

```
U 0323, 0003,96 ;5245           WORD[0396]
                ;5246
U 0324, 0003,9C ;5247           WORD[039C]              ;LOCATION 19C
U 0325, 0003,9C ;5248           WORD[039C]
                ;5249
U 0326, 0003,B8 ;5250           WORD[03B8]              ;LOCATION 1A0
U 0327, 0003,21 ;5251           WORD[0321]
                ;5252
U 0328, 0003,A6 ;5253           WORD[03A6]              ;LOCATION 1A4
U 0329, 0003,B8 ;5254           WORD[03B8]
                ;5255
U 032A, 0002,1E ;5256           WORD[021E]              ;LOCATION 1A8
U 032B, 0003,A4 ;5257           WORD[03A4]
                ;5258
U 032C, 0001,44 ;5259           WORD[0144]              ;LOCATION 1AC
U 032D, 0003,20 ;5260           WORD[0320]
                ;5261
U 032E, 0000,21 ;5262           WORD[0021]              ;LOCATION 1B0
U 032F, 0003,24 ;5263           WORD[0324]
                ;5264
U 0330, 0001,44 ;5265           WORD[0144]              ;LOCATION 1B4
U 0331, 0002,09 ;5266           WORD[0209]
                ;5267
U 0332, 0003,9E ;5268           WORD[039E]              ;LOCATION 1B8
U 0333, 0000,18 ;5269           WORD[0018]
                ;5270
U 0334, 0003,42 ;5271           WORD[0342]              ;LOCATION 1BC
U 0335, 0001,68 ;5272           WORD[0168]
                ;5273
U 0336, 0002,40 ;5274           WORD[0240]              ;LOCATION 1C0
U 0337, 0003,AC ;5275           WORD[03AC]
                ;5276
U 0338, 0000,93 ;5277           WORD[0093]              ;LOCATION 1C4
U 0339, 0000,94 ;5278           WORD[0094]
                ;5279
U 033A, 0001,7C ;5280           WORD[017C]              ;LOCATION 1C8
U 033B, 0000,B9 ;5281           WORD[00B9]
                ;5282
U 033C, 0003,AE ;5283           WORD[03AE]              ;LOCATION 1CC
U 033D, 0001,68 ;5284           WORD[0168]
                ;5285
U 033E, 0001,44 ;5286           WORD[0144]              ;LOCATION 1D0
U 033F, 0002,1E ;5287           WORD[021E]
                ;5288
U 0340, 0003,BD ;5289           WORD[03BD]              ;LOCATION 1D4
U 0341, 0000,15 ;5290           WORD[0015]
                ;5291
U 0342, 0003,C0 ;5292           WORD[03C0]              ;LOCATION 1D8
U 0343, 0002,09 ;5293           WORD[0209]
                ;5294
U 0344, 0003,36 ;5295           WORD[0336]              ;LOCATION 1DC
U 0345, 0000,00 ;5296           WORD[0000]
                ;5297
U 0346, 0001,68 ;5298           WORD[0168]              ;LOCATION 1E0
U 0347, 0002,E5 ;5299           WORD[02E5]
```

```
                ;5300
U 0348, 0003,37 ;5301           WORD[0337]              ;LOCATION 1E4
U 0349, 0001,68 ;5302           WORD[0168]
                ;5303
U 034A, 0001,3C ;5304           WORD[013C]              ;LOCATION 1E8
U 034B, 0003,92 ;5305           WORD[0392]
                ;5306
U 034C, 0000,3D ;5307           WORD[003D]              ;LOCATION 1EC
U 034D, 0003,73 ;5308           WORD[0373]
                ;5309
U 034E, 0003,B0 ;5310           WORD[03B0]              ;LOCATION 1F0
U 034F, 0001,68 ;5311           WORD[0168]
                ;5312
U 0350, 0001,CF ;5313           WORD[01CF]              ;LOCATION 1F4
U 0351, 0001,45 ;5314           WORD[0145]
                ;5315
U 0352, 0001,B7 ;5316           WORD[01B7]              ;LOCATION 1F8
U 0353, 0000,C8 ;5317           WORD[00C8]
                ;5318
U 0354, 0003,6E ;5319           WORD[036E]              ;LOCATION 1FC
U 0355, 0002,F1 ;5320           WORD[02F1]
                ;5321
U 0356, 0002,F7 ;5322           WORD[02F7]              ;LOCATION 200
U 0357, 0003,7A ;5323           WORD[037A]
                ;5324
U 0358, 0002,08 ;5325           WORD[0208]              ;LOCATION 204
U 0359, 0001,9F ;5326           WORD[019F]
                ;5327
U 035A, 0001,68 ;5328           WORD[0168]              ;LOCATION 208
U 035B, 0001,45 ;5329           WORD[0145]
                ;5330
U 035C, 0003,B0 ;5331           WORD[03B0]              ;LOCATION 20C
U 035D, 0001,5C ;5332           WORD[015C]
                ;5333
U 035E, 0003,92 ;5334           WORD[0392]              ;LOCATION 210
U 035F, 0003,B0 ;5335           WORD[03B0]
                ;5336
U 0360, 0001,68 ;5337           WORD[0168]              ;LOCATION 214
U 0361, 0001,FB ;5338           WORD[01FB]
                ;5339
U 0362, 0002,F7 ;5340           WORD[02F7]              ;LOCATION 218
U 0363, 0002,2B ;5341           WORD[022B]
                ;5342
U 0364, 0000,06 ;5343           WORD[0006]              ;LOCATION 21C
U 0365, 0000,93 ;5344           WORD[0093]
                ;5345
U 0366, 0003,47 ;5346           WORD[0347]              ;LOCATION 220
U 0367, 0001,24 ;5347           WORD[0124]
                ;5348
U 0368, 0000,93 ;5349           WORD[C093]              ;LOCATION 224
U 0369, 0003,47 ;5350           WORD[0347]
                ;5351
U 036A, 0003,42 ;5352           WORD[0342]              ;LOCATION 228
U 036B, 0000,93 ;5353           WORD[0093]
                ;5354
```

```
U 036C, 0003,47 ;5355          WORD[0347]          ;LOCATION 22C
U 036D, 0001,F1 ;5356          WORD[01F1]
                ;5357
U 036E, 0000,93 ;5358          WORD[0093]          ;LOCATION 230
U 036F, 0003,47 ;5359          WORD[0347]
                ;5360
U 0370, 0000,91 ;5361          WORD[0091]          ;LOCATION 234
U 0371, 0003,7A ;5362          WORD[037A]
                ;5363
U 0372, 0003,A8 ;5364          WORD[03A8]          ;LOCATION 238
U 0373, 0001,26 ;5365          WORD[0126]
                ;5366
U 0374, 0000,94 ;5367          WORD[0094]          ;LOCATION 23C
U 0375, 0002,40 ;5368          WORD[0240]
                ;5369
U 0376, 0001,68 ;5370          WORD[0168]          ;LOCATION 240
U 0377, 0002,80 ;5371          WORD[0280]
                ;5372
U 0378, 0000,93 ;5373          WORD[0093]          ;LOCATION 244
U 0379, 0000,94 ;5374          WORD[0094]
                ;5375
U 037A, 0000,F4 ;5376          WORD[00F4]          ;LOCATION 248
U 037B, 0000,93 ;5377          WORD[0093]
                ;5378
U 037C, 0000,94 ;5379          WORD[0094]          ;LOCATION 24C
U 037D, 0002,3B ;5380          WORD[023B]
                ;5381
U 037E, 0000,93 ;5382          WORD[0093]          ;LOCATION 250
U 037F, 0003,92 ;5383          WORD[0392]
                ;5384
U 0380, 0001,CE ;5385          WORD[01CE]          ;LOCATION 254
U 0381, 0000,93 ;5386          WORD[0093]
                ;5387
U 0382, 0003,7A ;5388          WORD[037A]          ;LOCATION 258
U 0383, 0000,94 ;5389          WORD[0094]
                ;5390
U 0384, 0001,74 ;5391          WORD[0174]          ;LOCATION 25C
U 0385, 0000,BC ;5392          WORD[00BC]
                ;5393
U 0386, 0003,C2 ;5394          WORD[03C2]          ;LOCATION 260
U 0387, 0002,09 ;5395          WORD[0209]
                ;5396
U 0388, 0003,C3 ;5397          WORD[03C3]          ;LOCATION 264
U 0389, 0002,09 ;5398          WORD[0209]
                ;5399
U 038A, 0001,44 ;5400          WORD[0144]          ;LOCATION 268
U 038B, 0001,23 ;5401          WORD[0123]
                ;5402
U 038C, 0003,00 ;5403          WORD[0300]          ;LOCATION 26C
U 038D, 0003,25 ;5404          WORD[0325]
                ;5405
U 038E, 0003,01 ;5406          WORD[0301]          ;LOCATION 270
U 038F, 0003,21 ;5407          WORD[0321]
                ;5408
U 0390, 0003,00 ;5409          WORD[0300]          ;LOCATION 274
```

```
U 0391, 0003,C4 ;5410           WORD[03C4]
                ;5411
U 0392, 0002,09 ;5412           WORD[0209]              ;LOCATION 278
U 0393, 0001,44 ;5413           WORD[0144]
                ;5414
U 0394, 0001,23 ;5415           WORD[0123]              ;LOCATION 27C
U 0395, 0000,F2 ;5416           WORD[00F2]
                ;5417
U 0396, 0003,42 ;5418           WORD[0342]              ;LOCATION 280
U 0397, 0003,C5 ;5419           WORD[03C5]
                ;5420
U 0398, 0000,93 ;5421           WORD[0093]              ;LOCATION 284
U 0399, 0000,94 ;5422           WORD[0094]
                ;5423
U 039A, 0003,C6 ;5424           WORD[03C6]              ;LOCATION 288
U 039B, 0002,10 ;5425           WORD[0210]
                ;5426
U 039C, 0002,32 ;5427           WORD[0232]              ;LOCATION 28C
U 039D, 0000,42 ;5428           WORD[0042]
                ;5429
U 039E, 0001,22 ;5430           WORD[0122]              ;LOCATION 290
U 039F, 0000,C8 ;5431           WORD[00C8]
                ;5432
U 03A0, 0003,C7 ;5433           WORD[03C7]              ;LOCATION 294
U 03A1, 0002,12 ;5434           WORD[0212]
                ;5435
U 03A2, 0001,21 ;5436           WORD[0121]              ;LOCATION 298
U 03A3, 0003,44 ;5437           WORD[0344]
                ;5438
U 03A4, 0000,88 ;5439           WORD[0088]              ;LOCATION 29C
U 03A5, 0002,00 ;5440           WORD[0200]
                ;5441
U 03A6, 0001,22 ;5442           WORD[0122]              ;LOCATION 2A0
U 03A7, 0003,CA ;5443           WORD[03CA]
                ;5444
U 03A8, 0003,D1 ;5445           WORD[03D1]              ;LOCATION 2A4
U 03A9, 0000,93 ;5446           WORD[0093]
                ;5447
U 03AA, 0000,94 ;5448           WORD[0094]              ;LOCATION 2A8
U 03AB, 0003,42 ;5449           WORD[0342]
                ;5450
U 03AC, 0000,88 ;5451           WORD[0088]              ;LOCATION 2AC
U 03AD, 0000,89 ;5452           WORD[0089]
                ;5453
U 03AE, 0000,8B ;5454           WORD[008B]              ;LOCATION 2B0
U 03AF, 0000,B9 ;5455           WORD[00B9]
                ;5456
U 03B0, 0001,7C ;5457           WORD[017C]              ;LOCATION 2B4
U 03B1, 0000,21 ;5458           WORD[0021]
                ;5459
U 03B2, 0000,18 ;5460           WORD[0018]              ;LOCATION 2B8
U 03B3, 0003,C8 ;5461           WORD[03C8]
                ;5462
U 03B4, 0000,00 ;5463           WORD[0000]              ;LOCATION 2BC
U 03B5, 0000,00 ;5464           WORD[0000]
```

```
                    ;5465
U 03B6, 0000,00 ;5466          WORD[0000]          ;LOCATION 2C0
U 03B7, 0000,00 ;5467          WORD[0000]
                    ;5468
U 03B8, 0000,00 ;5469          WORD[0000]          ;LOCATION 2C4
U 03B9, 0000,00 ;5470          WORD[0000]
                    ;5471
U 03BA, 0000,00 ;5472          WORD[0000]          ;LOCATION 2C8
U 03BB, 0000,00 ;5473          WORD[0000]
                    ;5474
U 03BC, 0003,CD ;5475          WORD[03CD]          ;LOCATION 2CC
U 03BD, 0003,CE ;5476          WORD[03CE]
                    ;5477
U 03BE, 0003,CF ;5478          WORD[03CF]          ;LOCATION 2D0
U 03BF, 0003,D0 ;5479          WORD[03D0]
                    ;5480
U 03C0, 0001,68 ;5481          WORD[0168]          ;LOCATION 2D4
U 03C1, 0002,40 ;5482          WORD[0240]
                    ;5483
U 03C2, 0000,93 ;5484          WORD[0093]          ;LOCATION 2D8
U 03C3, 0000,15 ;5485          WORD[0015]
                    ;5486
U 03C4, 0002,76 ;5487          WORD[0276]          ;LOCATION 2DC
U 03C5, 0003,A8 ;5488          WORD[03A8]
                    ;5489
U 03C6, 0003,98 ;5490          WORD[0398]          ;LOCATION 2E0
U 03C7, 0003,A0 ;5491          WORD[03A0]
                    ;5492
U 03C8, 0001,68 ;5493          WORD[0168]          ;LOCATION 2E4
U 03C9, 0003,68 ;5494          WORD[0368]
                    ;5495
U 03CA, 0003,70 ;5496          WORD[0370]          ;LOCATION 2E8
U 03CB, 0003,B0 ;5497          WORD[03B0]
                    ;5498
U 03CC, 0003,6A ;5499          WORD[036A]          ;LOCATION 2EC
U 03CD, 0003,72 ;5500          WORD[0372]
                    ;5501
U 03CE, 0003,7A ;5502          WORD[037A]          ;LOCATION 2F0
U 03CF, 0003,82 ;5503          WORD[0382]
                    ;5504
U 03D0, 0003,8A ;5505          WORD[038A]          ;LOCATION 2F4
U 03D1, 0003,92 ;5506          WORD[0392]
                    ;5507
U 03D2, 0003,9A ;5508          WORD[039A]          ;LOCATION 2F8
U 03D3, 0003,A2 ;5509          WORD[03A2]
                    ;5510
U 03D4, 0003,AA ;5511          WORD[03AA]          ;LOCATION 2FC
U 03D5, 0003,B2 ;5512          WORD[03B2]
                    ;5513
U 03D6, 0003,6C ;5514          WORD[036C]          ;LOCATION 300
U 03D7, 0002,E5 ;5515          WORD[02E5]
                    ;5516
U 03D8, 0002,E8 ;5517          WORD[02E8]          ;LOCATION 304
U 03D9, 0002,E9 ;5518          WORD[02E9]
                    ;5519
```

```
U 03DA, 0003,88 ;5520          WORD[0388]          ;LOCATION 308
U 03DB, 0003,78 ;5521          WORD[0378]
                ;5522
U 03DC, 0003,80 ;5523          WORD[0380]          ;LOCATION 30C
U 03DD, 0002,EC ;5524          WORD[02EC]
                ;5525
U 03DE, 0002,ED ;5526          WORD[02ED]          ;LOCATION 310
U 03DF, 0002,F0 ;5527          WORD[02F0]
                ;5528
U 03E0, 0002,F1 ;5529          WORD[02F1]          ;LOCATION 314
U 03E1, 0002,F4 ;5530          WORD[02F4]
                ;5531
U 03E2, 0002,F5 ;5532          WORD[02F5]          ;LOCATION 318
U 03E3, 0002,F8 ;5533          WORD[02F8]
                ;5534
U 03E4, 0002,FA ;5535          WORD[02FA]          ;LOCATION 31C
U 03E5, 0003,28 ;5536          WORD[0328]
                ;5537
U 03E6, 0003,2A ;5538          WORD[032A]          ;LOCATION 320
U 03E7, 0003,2C ;5539          WORD[032C]
                ;5540
U 03E8, 0003,2D ;5541          WORD[032D]          ;LOCATION 324
U 03E9, 0003,30 ;5542          WORD[0330]
                ;5543
U 03EA, 0003,32 ;5544          WORD[0332]          ;LOCATION 328
U 03EB, 0003,34 ;5545          WORD[0334]
                ;5546
U 03EC, 0003,36 ;5547          WORD[0336]          ;LOCATION 32C
U 03ED, 0003,38 ;5548          WORD[0338]
                ;5549
U 03EE, 0003,3A ;5550          WORD[033A]          ;LOCATION 330
U 03EF, 0003,3C ;5551          WORD[033C]
                ;5552
U 03F0, 0003,3E ;5553          WORD[033E]          ;LOCATION 334
U 03F1, 0003,40 ;5554          WORD[0340]
                ;5555
U 03F2, 0000,74 ;5556          WORD[0074]          ;LOCATION 338
U 03F3, 0000,65 ;5557          WORD[0065]
                ;5558
U 03F4, 0001,68 ;5559          WORD[0168]          ;LOCATION 33C
U 03F5, 0002,DD ;5560          WORD[02DD]
                ;5561
U 03F6, 0003,A8 ;5562          WORD[03A8]          ;LOCATION 340
U 03F7, 0001,22 ;5563          WORD[0122]
                ;5564
U 03F8, 0003,7A ;5565          WORD[037A]          ;LOCATION 344
U 03F9, 0002,E4 ;5566          WORD[02E4]
                ;5567
U 03FA, 0003,B0 ;5568          WORD[03B0]          ;LOCATION 348
U 03FB, 0000,11 ;5569          WORD[0011]
                ;5570
U 03FC, 0003,AA ;5571          WORD[03AA]          ;LOCATION 34C
U 03FD, 0003,50 ;5572          WORD[0350]
                ;5573
U 03FE, 0003,92 ;5574          WORD[0392]          ;LOCATION 350
```

```
U 03FF, 0001,95 ;5575          WORD[0195]
                ;5576
U 0400, 0003,88 ;5577          WORD[0388]              ;LOCATION 354
U 0401, 0003,55 ;5578          WORD[0355]
                ;5579
U 0402, 0002,E5 ;5580          WORD[02E5]              ;LOCATION 358
U 0403, 0003,51 ;5581          WORD[0351]
                ;5582
U 0404, 0002,F1 ;5583          WORD[02F1]              ;LOCATION 35C
U 0405, 0003,54 ;5584          WORD[0354]
                ;5585
U 0406, 0002,EC ;5586          WORD[02EC]              ;LOCATION 360
U 0407, 0001,8D ;5587          WORD[018D]
                ;5588
U 0408, 0003,2A ;5589          WORD[032A]              ;LOCATION 364
U 0409, 0003,58 ;5590          WORD[0358]
                ;5591
U 040A, 0002,F8 ;5592          WORD[02F8]              ;LOCATION 368
U 040B, 0003,5C ;5593          WORD[035C]
                ;5594
U 040C, 0003,36 ;5595          WORD[0336]              ;LOCATION 36C
U 040D, 0003,59 ;5596          WORD[0359]
                ;5597
U 040E, 0003,30 ;5598          WORD[0330]              ;LOCATION 370
U 040F, 0003,60 ;5599          WORD[0360]
                ;5600
U 0410, 0003,3C ;5601          WORD[033C]              ;LOCATION 374
U 0411, 0003,62 ;5602          WORD[0362]
                ;5603
U 0412, 0003,42 ;5604          WORD[0342]              ;LOCATION 378
U 0413, 0001,68 ;5605          WORD[0168]
                ;5606
U 0414, 0003,45 ;5607          WORD[0345]              ;LOCATION 37C
U 0415, 0003,68 ;5608          WORD[0368]
                ;5609
U 0416, 0003,AA ;5610          WORD[03AA]              ;LOCATION 380
U 0417, 0003,4A ;5611          WORD[034A]
                ;5612
U 0418, 0003,B2 ;5613          WORD[03B2]              ;LOCATION 384
U 0419, 0003,A8 ;5614          WORD[03A8]
                ;5615
U 041A, 0003,46 ;5616          WORD[0346]              ;LOCATION 388
U 041B, 0003,B0 ;5617          WORD[03B0]
                ;5618
U 041C, 0003,5E ;5619          WORD[035E]              ;LOCATION 38C
U 041D, 0003,7A ;5620          WORD[037A]
                ;5621
U 041E, 0002,C8 ;5622          WORD[02C8]              ;LOCATION 390
U 041F, 0003,92 ;5623          WORD[0392]
                ;5624
U 0420, 0003,64 ;5625          WORD[0364]              ;LOCATION 394
U 0421, 0002,E5 ;5626          WORD[02E5]
                ;5627
U 0422, 0003,6D ;5628          WORD[036D]              ;LOCATION 398
U 0423, 0003,88 ;5629          WORD[0388]
```

```
                    ;5630
U 0424, 0000,43 ;5631       WORD[0043]          ;LOCATION 39C
U 0425, 0002,EC ;5632       WORD[02EC]
                    ;5633
U 0426, 0003,74 ;5634       WORD[0374]          ;LOCATION 3A0
U 0427, 0002,F1 ;5635       WORD[02F1]
                    ;5636
U 0428, 0003,75 ;5637       WORD[0375]          ;LOCATION 3A4
U 0429, 0002,F8 ;5638       WORD[02F8]
                    ;5639
U 042A, 0003,7C ;5640       WORD[037C]          ;LOCATION 3A8
U 042B, 0003,2A ;5641       WORD[032A]
                    ;5642
U 042C, 0003,7E ;5643       WORD[037E]          ;LOCATION 3AC
U 042D, 0003,30 ;5644       WORD[0330]
                    ;5645
U 042E, 0003,84 ;5646       WORD[0384]          ;LOCATION 3B0
U 042F, 0003,36 ;5647       WORD[0336]
                    ;5648
U 0430, 0002,72 ;5649       WORD[0272]          ;LOCATION 3B4
U 0431, 0003,3C ;5650       WORD[033C]
                    ;5651
U 0432, 0003,8E ;5652       WORD[038E]          ;LOCATION 3B8
U 0433, 0000,83 ;5653       WORD[0083]
                    ;5654
U 0434, 0002,B0 ;5655       WORD[02B0]          ;LOCATION 3BC
U 0435, 0000,4F ;5656       WORD[004F]
                    ;5657
U 0436, 0003,BF ;5658       WORD[03BF]          ;LOCATION 3C0
U 0437, 0000,93 ;5659       WORD[0093]
                    ;5660
U 0438, 0002,D0 ;5661       WORD[02D0]          ;LOCATION 3C4
U 0439, 0001,42 ;5662       WORD[0142]
                    ;5663
U 043A, 0001,65 ;5664       WORD[0165]          ;LOCATION 3C8
U 043B, 0001,43 ;5665       WORD[0143]
                    ;5666
U 043C, 0001,59 ;5667       WORD[0159]          ;LOCATION 3CC
U 043D, 0003,42 ;5668       WORD[0342]
                    ;5669
U 043E, 0000,93 ;5670       WORD[0093]          ;LOCATION 3D0
U 043F, 0003,A8 ;5671       WORD[03A8]
                    ;5672
U 0440, 0001,68 ;5673       WORD[0168]          ;LOCATION 3D4
U 0441, 0003,B0 ;5674       WORD[03B0]
                    ;5675
U 0442, 0003,C1 ;5676       WORD[03C1]          ;LOCATION 3D8
U 0443, 0001,68 ;5677       WORD[0168]
                    ;5678
U 0444, 0003,7A ;5679       WORD[037A]          ;LOCATION 3DC
U 0445, 0000,93 ;5680       WORD[0093]
                    ;5681
U 0446, 0001,02 ;5682       WORD[0102]          ;LOCATION 3E0
U 0447, 0003,82 ;5683       WORD[0382]
                    ;5684
```

```
U 0448, 0003,8A ;5685          WORD[038A]          ;LOCATION 3E4
U 0449, 0002,E4 ;5686          WORD[02E4]
                ;5687
U 044A, 0003,42 ;5688          WORD[0342]          ;LOCATION 3E8
U 044B, 0001,D8 ;5689          WORD[01D8]
                ;5690
U 044C, 0000,B9 ;5691          WORD[00B9]          ;LOCATION 3EC
U 044D, 0000,06 ;5692          WORD[0006]
                ;5693
U 044E, 0003,A8 ;5694          WORD[03A8]          ;LOCATION 3F0
U 044F, 0003,B0 ;5695          WORD[03B0]
                ;5696
U 0450, 0001,42 ;5697          WORD[0142]          ;LOCATION 3F4
U 0451, 0001,43 ;5698          WORD[0143]
                ;5699
U 0452, 0000,83 ;5700          WORD[0083]          ;LOCATION 3F8
U 0453, 0001,57 ;5701          WORD[0157]
                ;5702
U 0454, 0002,D0 ;5703          WORD[02D0]          ;LOCATION 3FC
U 0455, 0000,9B ;5704          WORD[009B]
                ;5705
U 0456, 0003,98 ;5706          WORD[0398]          ;LOCATION 400
U 0457, 0003,A0 ;5707          WORD[03A0]
                ;5708
U 0458, 0000,65 ;5709          WORD[0065]          ;LOCATION 404
U 0459, 0003,A8 ;5710          WORD[03A8]
                ;5711
U 045A, 0000,93 ;5712          WORD[0093]          ;LOCATION 408
U 045B, 0000,94 ;5713          WORD[0094]
                ;5714
U 045C, 0001,F1 ;5715          WORD[01F1]          ;LOCATION 40C
U 045D, 0001,50 ;5716          WORD[0150]
                ;5717
U 045E, 0000,83 ;5718          WORD[0083]          ;LOCATION 410
U 045F, 0001,99 ;5719          WORD[0199]
                ;5720
U 0460, 0003,92 ;5721          WORD[0392]          ;LOCATION 414
U 0461, 0003,B2 ;5722          WORD[03B2]
                ;5723
U 0462, 0003,6C ;5724          WORD[036C]          ;LOCATION 418
U 0463, 0003,50 ;5725          WORD[0350]
                ;5726
U 0464, 0003,64 ;5727          WORD[0364]          ;LOCATION 41C
U 0465, 0001,8C ;5728          WORD[018C]
                ;5729
U 0466, 0001,68 ;5730          WORD[0168]          ;LOCATION 420
U 0467, 0002,32 ;5731          WORD[0232]
                ;5732
U 0468, 0003,36 ;5733          WORD[0336]          ;LOCATION 424
U 0469, 0003,38 ;5734          WORD[0338]
                ;5735
U 046A, 0003,3A ;5736          WORD[033A]          ;LOCATION 428
U 046B, 0003,59 ;5737          WORD[0359]
                ;5738
U 046C, 0000,15 ;5739          WORD[0015]          ;LOCATION 42C
```

```
U 046D, 0003,30 ;5740          WORD[0330]
                ;5741
U 046E, 0003,32 ;5742          WORD[0332]              ;LOCATION 430
U 046F, 0003,34 ;5743          WORD[0334]
                ;5744
U 0470, 0003,58 ;5745          WORD[0358]              ;LOCATION 434
U 0471, 0000,08 ;5746          WORD[0008]
                ;5747
U 0472, 0003,7A ;5748          WORD[037A]              ;LOCATION 438
U 0473, 0003,82 ;5749          WORD[0382]
                ;5750
U 0474, 0003,8A ;5751          WORD[038A]              ;LOCATION 43C
U 0475, 0002,E4 ;5752          WORD[02E4]
                ;5753
U 0476, 0000,F0 ;5754          WORD[00F0]              ;LOCATION 440
U 0477, 0003,42 ;5755          WORD[0342]
                ;5756
U 0478, 0003,AA ;5757          WORD[03AA]              ;LOCATION 444
U 0479, 0003,B2 ;5758          WORD[03B2]
                ;5759
U 047A, 0003,6C ;5760          WORD[036C]              ;LOCATION 448
U 047B, 0003,50 ;5761          WORD[0350]
                ;5762
U 047C, 0000,7E ;5763          WORD[007E]              ;LOCATION 44C
U 047D, 0000,B9 ;5764          WORD[00B9]
                ;5765
U 047E, 0001,7C ;5766          WORD[017C]              ;LOCATION 450
U 047F, 0000,21 ;5767          WORD[0021]
                ;5768
U 0480, 0000,18 ;5769          WORD[0018]              ;LOCATION 454
U 0481, 0000,B2 ;5770          WORD[00B2]
                ;5771
U 0482, 0001,B0 ;5772          WORD[01B0]              ;LOCATION 458
U 0483, 0002,34 ;5773          WORD[0234]
                ;5774
U 0484, 0002,24 ;5775          WORD[0224]              ;LOCATION 45C
U 0485, 0000,98 ;5776          WORD[0098]
                ;5777
U 0486, 0000,9A ;5778          WORD[009A]              ;LOCATION 460
U 0487, 0003,E7 ;5779          WORD[03E7]
                ;5780
U 0488, 0001,89 ;5781          WORD[0189]              ;LOCATION 464
U 0489, 0000,93 ;5782          WORD[0093]
                ;5783
U 048A, 0000,A0 ;5784          WORD[00A0]              ;LOCATION 468
U 048B, 0002,32 ;5785          WORD[0232]
                ;5786
U 048C, 0000,B9 ;5787          WORD[00B9]              ;LOCATION 46C
U 048D, 0001,7C ;5788          WORD[017C]
                ;5789
U 048E, 0000,03 ;5790          WORD[0003]              ;LOCATION 470
U 048F, 0002,EC ;5791          WORD[02EC]
                ;5792
U 0490, 0002,ED ;5793          WORD[02ED]              ;LOCATION 474
U 0491, 0000,31 ;5794          WORD[0031]
```

```
                ;5795
U 0492, 0003,2A ;5796           WORD[032A]              ;LOCATION 478
U 0493, 0003,2C ;5797           WORD[032C]
                ;5798
U 0494, 0002,62 ;5799           WORD[0262]              ;LOCATION 47C
U 0495, 0003,AA ;5800           WORD[03AA]
                ;5801
U 0496, 0003,B2 ;5802           WORD[03B2]              ;LOCATION 480
U 0497, 0000,AF ;5803           WORD[00AF]
                ;5804
U 0498, 0000,93 ;5805           WORD[0093]              ;LOCATION 484
U 0499, 0000,94 ;5806           WORD[0094]
                ;5807
U 049A, 0003,C7 ;5808           WORD[03C7]              ;LOCATION 488
U 049B, 0003,CE ;5809           WORD[03CE]
                ;5810
U 049C, 0000,40 ;5811           WORD[0040]              ;LOCATION 48C
U 049D, 0000,41 ;5812           WORD[0041]
                ;5813
U 049E, 0018,42 ;5814           WORD[1842]              ;LOCATION 490
U 049F, 0018,43 ;5815           WORD[1843]
                ;5816
U 04A0, 0030,44 ;5817           WORD[3044]              ;LOCATION 494
U 04A1, 0030,45 ;5818           WORD[3045]
                ;5819
U 04A2, 0028,46 ;5820           WORD[2846]              ;LOCATION 498
U 04A3, 0028,47 ;5821           WORD[2847]
                ;5822
U 04A4, 0040,48 ;5823           WORD[4048]              ;LOCATION 49C
U 04A5, 0048,49 ;5824           WORD[4849]
                ;5825
U 04A6, 0050,4A ;5826           WORD[504A]              ;LOCATION 4A0
U 04A7, 0058,4B ;5827           WORD[584B]
                ;5828
U 04A8, 0080,4C ;5829           WORD[804C]              ;LOCATION 4A4
U 04A9, 0088,4D ;5830           WORD[884D]
                ;5831
U 04AA, 0090,4E ;5832           WORD[904E]              ;LOCATION 4A8
U 04AB, 0000,4F ;5833           WORD[004F]
                ;5834
U 04AC, 0010,51 ;5835           WORD[1051]              ;LOCATION 4AC
U 04AD, 0038,54 ;5836           WORD[3854]
                ;5837
U 04AE, 0020,55 ;5838           WORD[2055]              ;LOCATION 4B0
U 04AF, 0068,56 ;5839           WORD[6856]
                ;5840
U 04B0, 0001,60 ;5841           WORD[0160]              ;LOCATION 4B4
U 04B1, 0001,61 ;5842           WORD[0161]
                ;5843
U 04B2, 0019,62 ;5844           WORD[1962]              ;LOCATION 4B8
U 04B3, 0019,63 ;5845           WORD[1963]
                ;5846
U 04B4, 0031,64 ;5847           WORD[3164]              ;LOCATION 4BC
U 04B5, 0031,65 ;5848           WORD[3165]
                ;5849
```

```
U 04B6, 0029,66 ;5850          WORD[2966]              ;LOCATION 4C0
U 04B7, 0029,67 ;5851          WORD[2967]
                ;5852
U 04B8, 0041,68 ;5853          WORD[4168]              ;LOCATION 4C4
U 04B9, 0049,69 ;5854          WORD[4969]
                ;5855
U 04BA, 0051,6A ;5856          WORD[516A]              ;LOCATION 4C8
U 04BB, 0059,6B ;5857          WORD[596B]
                ;5858
U 04BC, 0081,6C ;5859          WORD[816C]              ;LOCATION 4CC
U 04BD, 0089,6D ;5860          WORD[896D]
                ;5861
U 04BE, 0091,6E ;5862          WORD[916E]              ;LOCATION 4D0
U 04BF, 0001,6F ;5863          WORD[016F]
                ;5864
U 04C0, 0011,71 ;5865          WORD[1171]              ;LOCATION 4D4
U 04C1, 0039,74 ;5866          WORD[3974]
                ;5867
U 04C2, 0021,75 ;5868          WORD[2175]              ;LOCATION 4D8
U 04C3, 0061,76 ;5869          WORD[6176]
                ;5870
U 04C4, 00C0,C4 ;5871          WORD[0C0C4]             ;LOCATION 4DC
U 04C5, 00C0,C5 ;5872          WORD[0C0C5]
                ;5873
U 04C6, 00D0,C6 ;5874          WORD[0D0C6]             ;LOCATION 4E0
U 04C7, 00D0,C7 ;5875          WORD[0D0C7]
                ;5876
U 04C8, 0079,32 ;5877          WORD[7932]              ;LOCATION 4E4
U 04C9, 0062,33 ;5878          WORD[6233]
                ;5879
U 04CA, 0002,40 ;5880          WORD[0240]              ;LOCATION 4E8
U 04CB, 0002,41 ;5881          WORD[0241]
                ;5882
U 04CC, 001A,42 ;5883          WORD[1A42]              ;LOCATION 4EC
U 04CD, 001A,43 ;5884          WORD[1A43]
                ;5885
U 04CE, 0032,44 ;5886          WORD[3244]              ;LOCATION 4F0
U 04CF, 0032,45 ;5887          WORD[3245]
                ;5888
U 04D0, 002A,46 ;5889          WORD[2A46]              ;LOCATION 4F4
U 04D1, 002A,47 ;5890          WORD[2A47]
                ;5891
U 04D2, 0042,48 ;5892          WORD[4248]              ;LOCATION 4F8
U 04D3, 004A,49 ;5893          WORD[4A49]
                ;5894
U 04D4, 0052,4A ;5895          WORD[524A]              ;LOCATION 4FC
U 04D5, 005A,4B ;5896          WORD[5A4B]
                ;5897
U 04D6, 0082,4C ;5898          WORD[824C]              ;LOCATION 500
U 04D7, 008A,4D ;5899          WORD[8A4D]
                ;5900
U 04D8, 0092,4E ;5901          WORD[924E]              ;LOCATION 504
U 04D9, 0002,4F ;5902          WORD[024F]
                ;5903
U 04DA, 0012,51 ;5904          WORD[1251]              ;LOCATION 508
```

```
U 04DB, 003A,54 ;5905           WORD[3A54]
                ;5906
U 04DC, 0022,55 ;5907           WORD[2255]              ;LOCATION 50C
U 04DD, 007A,56 ;5908           WORD[7A56]
                ;5909
U 04DE, 0003,60 ;5910           WORD[0360]              ;LOCATION 510
U 04DF, 0003,61 ;5911           WORD[0361]
                ;5912
U 04E0, 001B,62 ;5913           WORD[1B62]              ;LOCATION 514
U 04E1, 001B,63 ;5914           WORD[1B63]
                ;5915
U 04E2, 0033,64 ;5916           WORD[3364]              ;LOCATION 518
U 04E3, 0033,65 ;5917           WORD[3365]
                ;5918
U 04E4, 002B,66 ;5919           WORD[2B66]              ;LOCATION 51C
U 04E5, 002B,67 ;5920           WORD[2B67]
                ;5921
U 04E6, 0043,68 ;5922           WORD[4368]              ;LOCATION 520
U 04E7, 004B,69 ;5923           WORD[4B69]
                ;5924
U 04E8, 0053,6A ;5925           WORD[536A]              ;LOCATION 524
U 04E9, 005B,6B ;5926           WORD[5B6B]
                ;5927
U 04EA, 0083,6C ;5928           WORD[836C]              ;LOCATION 528
U 04EB, 008B,6D ;5929           WORD[8B6D]
                ;5930
U 04EC, 0093,6E ;5931           WORD[936E]              ;LOCATION 52C
U 04ED, 0003,6F ;5932           WORD[036F]
                ;5933
U 04EE, 0013,71 ;5934           WORD[1371]              ;LOCATION 530
U 04EF, 003B,74 ;5935           WORD[3B74]
                ;5936
U 04F0, 0023,75 ;5937           WORD[2375]              ;LOCATION 534
U 04F1, 0073,76 ;5938           WORD[7376]
                ;5939
U 04F2, 0078,98 ;5940           WORD[7898]              ;LOCATION 538
U 04F3, 0070,99 ;5941           WORD[7099]
                ;5942
U 04F4, 0063,F6 ;5943           WORD[63F6]              ;LOCATION 53C
U 04F5, 006B,F7 ;5944           WORD[6BF7]
                ;5945
U 04F6, 0003,43 ;5946           WORD[0343]              ;LOCATION 540
U 04F7, 0003,AA ;5947           WORD[03AA]
                ;5948
U 04F8, 0003,B2 ;5949           WORD[03B2]              ;LOCATION 544
U 04F9, 0003,6C ;5950           WORD[036C]
                ;5951
U 04FA, 0003,50 ;5952           WORD[0350]              ;LOCATION 548
U 04FB, 0000,00 ;5953           WORD[0000]
                ;5954
U 04FC, 0000,00 ;5955           WORD[0000]              ;LOCATION 54C
U 04FD, 0000,00 ;5956           WORD[0000]
                ;5957
U 04FE, 0000,00 ;5958           WORD[0000]              ;LOCATION 550
U 04FF, 0000,00 ;5959           WORD[0000]
```

```
                  ;5960
U 0500, 0000,00 ;5961          WORD[0000]              ;LOCATION 554
U 0501, 0000,00 ;5962          WORD[0000]
                  ;5963
U 0502, 0000,00 ;5964          WORD[0000]              ;LOCATION 558
U 0503, 0000,00 ;5965          WORD[0000]
                  ;5966
U 0504, 0000,00 ;5967          WORD[0000]              ;LOCATION 55C
U 0505, 0000,00 ;5968          WORD[0000]
                  ;5969
                  ;5970
                  ;5971
```

```
;5972    .page
;5973    ;mml            Contents        Meaning
;5974
;5975    ;0              010             T1:     Word to Assert ACC SYNC
;5976    ;2              001             T3:     Expected data when 001 forced
;5977    ;4              00E                     Expected data when 002 forced
;5978    ;6              040                     Expected data when 008 forced
;5979    ;8              0AB                     Expected data when 020 forced
;5980    ;A              014                     Expected data when 040 forced
;5981    ;C              082                     Expected data when 080 forced
;5982    ;E              008                     Expected data when 100 forced
;5983    ;10             0AC                     Expected data when 024 forced
;5984    ;12             03C                     Expected data when 11 forced
;5985    ;14             00E                     Last Address forced
;5986    ;16             000                     Address before 001
;5987    ;18             001                     Address expected when 000 forced
;5988    ;1A             034                     Address before 002
;5989    ;1C             009                     Address expected when 031 forced
;5990    ;1E             020                     Address before 024
;5991    ;20             0AB                     Address expected when 0AB forced
;5992    ;22             038                     Address before 008
;5993    ;24             022                     Address expected when 039 forced
;5994    ;26             2AD                     Address before 010
;5995    ;28             030                     Address expected when 2AD forced
;5996    ;2A             09B                     Address before 020
;5997    ;2C             024                     Address expected when 04B forced
;5998    ;2E             022                     Address before 040
;5999    ;30             02C                     Address expected when 02C forced
;6000    ;32             07F                     Address before 080
;6001    ;34             081                     Address expected when 07F forced
;6002    ;36             00E                     Address before 100
;6003    ;38             038                     Address expected when 00E forced
;6004    ;3A             007             T4:     Expected data when BPW forced.
;6005    ;3C             2B2                     Address of #1 bad parity word
;6006    ;3E             007                     Expected data for #2 BPW
;6007    ;40             2B4                     Address of #2 BPW
;6008    ;42             007                     Expected data for #3 BPW
;6009    ;44             2B6                     Address of #3 BPW
;6010    ;46             007                     Expected data for #4 BPW
;6011    ;48             2B8                     Address of #4 BPW
;6012    ;4A             007                     Expected data for #5 BPW
;6013    ;4C             2BC                     Address of #5 BPW
;6014    ;4E             007                     Expected data for #6 BPW
;6015    ;50             2BD                     Address of #6 BPW
;6016    ;52             007                     Expected data for #7 BPW
;6017    ;54             2C0                     Address of #7 BPW
;6018    ;56             007                     Expected data for #8 BPW
;6019    ;58             2C4                     Address of #8 BPW
;6020    ;5A             005                     Expected data for #9 BPW
;6021    ;5C             2C5                     Address of #9 BPW
;6022    ;5E             003                     Expected data for #10 BPW
;6023    ;60             2C9                     Address of #10 BPW
;6024    ;62             299             T6:     Address of FPA SYNC word
;6025    ;64             298             T7:     Address of word to speed up FPA clock
;6026    ;66             299                     Address of FPA SYNC word
```

```
;6027  ;68   302        T8:   Expected data
;6028  ;6A   390              Addresss to be forced
;6029  ;6C   300              Expected data
;6030  ;6E   2A7        TA:   Address of the double load routine
;6031  ;70   059              Address of STORE first float
;6032  ;72   06E              Address of STORE second float
;6033  ;74   3A8              MOV WR[1]
;6034  ;76   398              STORE first WR[1]
;6035  ;78   3A0              STORE second WR[1]
;6036  ;7A   168              MOV WR[2]
;6037  ;7C   368              STORE first float WR[2]
;6038  ;7E   370              STORE second float WR[2]
;6039  ;80   3B0              MOV WR[3]
;6040  ;82   36A              STORE first float WR[3]
;6041  ;84   372              STORE second float WR[3]
;6042  ;86   37A              MOV WR[4]
;6043  ;88   382              STORE first float WR[4]
;6044  ;8A   38A              STORE second float WR[4]
;6045  ;8C   392              MOV WR[5]
;6046  ;8E   39A              STORE first float WR[5]
;6047  ;90   3A2              STORE second float WR[5]
;6048  ;92   3AA              MOV WR[6]
;6049  ;94   3B2              STORE first float WR[6]
;6050  ;96   36C              STORE second float WR[6]
;6051  ;98   2E5              MOV WR[7]
;6052  ;9A   2E8              STORE first float WR[7]
;6053  ;9C   2E9              STORE second float WR[7]
;6054  ;9E   388              MOV WR[8]
;6055  ;A0   378              STORE first float WR[8]
;6056  ;A2   380              STORE second float WR[8]
;6057  ;A4   2EC              MOV WR[9]
;6058  ;A6   2ED              STORE first float WR[9]
;6059  ;A8   2F0              STORE second float WR[9]
;6060  ;AA   2F1              MOV WR[A]
;6061  ;AC   2F4              STORE first float WR[A]
;6062  ;AE   2F5              STORE second float WR[A]
;6063  ;B0   2F8              MOV WR[B]
;6064  ;B2   2FA              STORE first float WR[B]
;6065  ;B4   328              STORE second float WR[B]
;6066  ;B6   32A              MOV WR[C]
;6067  ;B8   32C              STORE first float WR[C]
;6068  ;BA   32D              STORE second float WR[C]
;6069  ;BC   330              MOV WR[D]
;6070  ;BE   332              STORE first float WR[D]
;6071  ;C0   334              STORE second float WR[D]
;6072  ;C2   336              MOV WR[E]
;6073  ;C4   338              STORE first float WR[E]
;6074  ;C6   33A              STORE second float WR[E]
;6075  ;C8   33C              MOV WR[F]
;6076  ;CA   33E              STORE first float WR[F]
;6077  ;CC   340              STORE second float WR[F]
;6078  ;CE   093              MOV WR[0] TO Q
;6079  ;D0   348              MOV Q TO WR[1]
;6080  ;D2   398              STORE first float WR[1]
;6081  ;D4   3A0              STORE second float WR[1]
```

```
;6082   ;D6          2A7          TB:     LOAD routine
;6083   ;D8          342                  ET0 left
;6084   ;DA          059                  ET0 store
;6085   ;DC          083                  ET0 right
;6086   ;DE          04F                  ET2 left
;6087   ;E0          345                  ET2 right
;6088   ;E2          168                  ET2 mov
;6089   ;E4          368                  ET2 store
;6090   ;E6          07E                  ET6 left
;6091   ;E8          34A                  ET6 right
;6092   ;EA          3AA                  ET6 mov
;6093   ;EC          3B2                  ET6 store
;6094   ;EE          04F                  EQ LEFT
;6095   ;F0          34A                  EQ RIGHT
;6096   ;F2          093                  ET0 TO EQ
;6097   ;F4          398                  STORE ET1
;6098   ;F6          348                  EQ TO ET1
;6099   ;F8          2DC          TC:     LOAD Routine
;6100   ;FA          065                  ET0 STORE
;6101   ;FC          2DD                  ET2 STORE
;6102   ;FE          168                  ET2 MOVE
;6103   ;100         2E4                  ET4 STORE
;6104   ;102         37A                  ET4 MOVE
;6105   ;104         350                  ET6 STORE
;6106   ;106         3AA                  ET6 MOVE
;6107   ;108         355                  ET8 STORE
;6108   ;10A         388                  ET8 MOVE
;6109   ;10C         354                  ETA STORE
;6110   ;10E         2F1                  ETA MOVE
;6111   ;110         358                  ETC STORE
;6112   ;112         32A                  ETC MOVE
;6113   ;114         359                  ETE STORE
;6114   ;116         336                  ETE MOVE
;6115   ;118         2DD                  EQ STORE
;6116   ;11A         093                  EQ MOVE
;6117   ;11C         347                  EQ MOVE FROM Q
;6118   ;11E         2DC          TD:     LOAD FT0
;6119   ;120         347                  MOV FQ TO FT2
;6120   ;122         2DD                  STORE FT2
;6121   ;124         3A8                  MOV FT0 TO FT1
;6122   ;126         122                  MOV FT1 TO FQ
;6123   ;128         3B0                  MOV FT0 TO FT3
;6124   ;12A         011                  MOV FT3 TO FQ
;6125   ;12C         392                  MOV FT0 TO FT5
;6126   ;12E         195                  MOV FT5 TO FQ
;6127   ;130         2E5                  MOV FT0 TO FT7
;6128   ;132         351                  MOV FT7 TO FQ
;6129   ;134         2EC                  MOV FT0 TO FT9
;6130   ;136         18D                  MOV FT9 TO FQ
;6131   ;138         2F8                  MOV FT0 TO FTB
;6132   ;13A         35C                  MOV FTB TO FQ
;6133   ;13C         330                  MOV FT0 TO FTD
;6134   ;13E         360                  MOV FTD TO FQ
;6135   ;140         33C                  MOV FT0 TO FTF
;6136   ;142         362                  MOV FTF TO FQ
```

```
;6137  ;144  2A7          TE:    LOAD
;6138  ;146  04F                 LEFT
;6139  ;148  34A                 RIGHT
;6140  ;14A  3B2                 STORE ET6
;6141  ;14C  3A8                 MOV TO 1
;6142  ;14E  346                 MOV FROM 1
;6143  ;150  3B0                 MOV TO 3
;6144  ;152  35E                 MOV FROM 3
;6145  ;154  37A                 MOV TO 4
;6146  ;156  2C8                 MOV FROM 4
;6147  ;158  392                 MOV TO 5
;6148  ;15A  364                 MOV FROM 5
;6149  ;15C  2E5                 MOV TO 7
;6150  ;15E  36D                 MOV FROM 7
;6151  ;160  388                 MOV TO 8
;6152  ;162  043                 MOV FROM 8
;6153  ;164  2EC                 MOV TO 9
;6154  ;166  374                 MOV FROM 9
;6155  ;168  2F1                 MOV TO A
;6156  ;16A  375                 MOV FROM A
;6157  ;16C  2F8                 MOV TO B
;6158  ;16E  37C                 MOV FROM B
;6159  ;170  32A                 MOV TO C
;6160  ;172  37E                 MOV FROM C
;6161  ;174  330                 MOV TO D
;6162  ;176  384                 MOV FROM D
;6163  ;178  336                 MOV TO E
;6164  ;17A  272                 MOV FROM E
;6165  ;17C  33C                 MOV TO F
;6166  ;17E  38E                 MOV FROM F
;6167  ;180  93           T13:   MOV FWR0 TO FQ, EWR0 TO FQ
;6168  ;182  254                 AND FWR0 WITH FQ TO FWR2
;6169  ;184  168          T14:   MOV FWR0 TO FWR2
;6170  ;186  2E5                 MOV FWR0 TO FWR[INT.MASK]
;6171  ;188  01F                 AND NOT FWR[INT.MASK] WITH FWR2
;6172  ;18A  2A7          T2A:   LOAD ET0
;6173  ;18C  342                 SHIFT LEFT ET0
;6174  ;18E  168                 MOV ET0 TO ET2
;6175  ;190  240                 ADD ET0 TO ET2
;6176  ;192  394                 BRANCH ON EXP COUT AND GRAND
;6177  ;194  17C          T2B:   SET SIGN OUT TO 1
;6178  ;196  396                 BRANCH ON SIGN OUT AND HUGE
;6179  ;198  0B9                 SET SIGN OUT TO 0
;6180  ;19A  396                 BRANCH ON SIGN OUT AND HUGE
;6181  ;19C  39C          T2C:   BRANCH ON CPU DATA AVAIL AND SINGLE
;6182  ;19E  39C          T2D:   BRANCH ON CPU DATA AVAIL AND SINGLE
;6183  ;1A0  3B8                 BRANCH ON CPU DATA AVAIL AND ADD+SUB
;6184  ;1A2  321                 BRANCH ON CPU RCV DATA
;6185  ;1A4  3A6                 BRANCH ON CPU RCV DATA (OP SYNC)
;6186  ;1A6  3B8          T2E:   Branch on CPU DATA AVAIL AND ADD+SUB
;6187  ;1A8  21E          T2F:   ADD FT0 TO FT2
;6188  ;1AA  3A4                 Branch on FCOUT AND EXT FUNC
;6189  ;1AC  144                 CLR FT0
;6190  ;1AE  320          T30:   Branch on OP1 SIGN and EMOD asserted
;6191  ;1B0  021                 Clock OP1 SIGN
```

```
;6192  ;1B2  324          T31:  Branch on F55 F3 and SINGLE
;6193  ;1B4  144                CLR FWR[FT0]
;6194  ;1B6  209                MOV FWR[FT0] TO FWR[FT0]
;6195  ;1B8  39E          T32:  Branch on OP2 SIGN and ADD+SUB asserted
;6196  ;1BA  018                CLOCK OP2 SIGN
;6197  ;1BC  342          T33:  Shift ET0 LEFT
;6198  ;1BE  168                MOV ET0 TO ET2
;6199  ;1C0  240                ADD ET2 TO ET0
;6200  ;1C2  3AC                BRANCH on EXP COUT and EXP15 F3
;6201  ;1C4  093          T34:  MOV ET0 TO EQ
;6202  ;1C6  094                MOV EQ TO ET0 AND CLOCK OP2=0
;6203  ;1C8  17C                CLOCK SIGN OUT WITH ONE
;6204  ;1CA  0B9                CLOCK SIGN OUT WITH ZERO
;6205  ;1CC  3AE                BRANCH ON SIGN OUT AND OP2=0
;6206  ;1CE  168          T35:  MOV FT0 TO FT2
;6207  ;1D0  144                CLR FT0
;6208  ;1D2  21E                ADD FT0 TO FT2
;6209  ;1D4  3BD                Branch on FRAC55 F3
;6210  ;1D6  015          T36:  shift FQ left and IN[ONE]
;6211  ;1D8  3C0                branch on FRAC47 F3 and 00Q0
;6212  ;1DA  209                MOV FT0 TO FT0
;6213  ;1DC  336                MOV FWR[FT0] TO FWR[D.RND]
;6214  ;1DE
;6215  ;1E0  168          T15:  MOV FWR0 TO FWR2
;6216  ;1E2  2E5                MOV FWR0 TO FWR[INT.MASK]
;6217  ;1E4  337                XOR FWR[INT.MASK] WITH FWR2
;6218  ;1E6  168          T16:  MOV FWR0 TO FWR2
;6219  ;1E8  13C                XNOR FWR[FT2] WITH FWR[FT2]
;6220  ;1EA  392                MOV FWR[FT0] TO FWR[FT5]
;6221  ;1EC  03D                MOV FWR[FT6] TO FWR[FT2]
;6222  ;1EE  373                XNOR ZERO WITH FWR[FT5] TO FWR[FT6]
;6223  ;1F0  3B0          T17:  MOV FWR[0] TO FWR[3]
;6224  ;1F2  168                MOV FWR[FT0] TO FWR[FT2]
;6225  ;1F4  1CF                ADD FWR[FT3] PLUS FWR[FT1] TO FQ
;6226  ;1F6  145                CLR FWR[FT1]
;6227  ;1F8  1B7                ADD FQ TO FWR[FT0]
;6228  ;1FA  0C8                MOV FWR[FT0] TO FQ
;6229  ;1FC  36E                CLR FQ
;6230  ;1FE  2F1                MOV EWR[ET0] TO EWR[ZERO]
;6231  ;200  2F7          T18:  CLR FWR[FT3]
;6232  ;202  37A                MOV FWR[FT0] TO FWR[FT4]
;6233  ;204  208                SUB FWR[FT3] FROM FWR[FT1] TO FQ
;6234  ;206  19F                MOV FWR[FT4] TO FWR[FT0]
;6235  ;208  168                MOV FWR[FT0] TO FWR[FT2]
;6236  ;20A  145                CLR FWR[FT1]
;6237  ;20C  3B0                MOV FWR[FT0] TO FWR[FT3]
;6238  ;20E  15C                SUB EWR[ET5] FROM EWR[ET0]
;6239  ;210  392                MOV EWR[ET0] TO EWR[ET5]
;6240  ;212  3B0          T19:  MOV FWR[FT0] TO FWR[FT3]
;6241  ;214  168                MOV FWR[FT0] TO FWR[FT2]
;6242  ;216  1FB                SUB FWR[FT1] FROM FWR[FT3] TO FQ
;6243  ;218  2F7                CLR FWR[FT3]
;6244  ;21A  22B                ADD EWR[ET2] TO -EWR[ET0]
;6245  ;21C  006          T1A:  SHF LEFT FWR[FT0] IN[ONE]
;6246  ;21E  093                MOV FWR0 TO FQ, EWR0 TO EQ
```

```
;6247  ;220  347              MOV FQ TO FWR[FT2]
;6248  ;222  124        T1B:  SHF RIGHT FWR[FT0] IN[MUL.SHF]
;6249  ;224  093              MOV FWR0 TO FQ, EWR0 TO EQ
;6250  ;226  347              MOV FQ TO FWR[FT2]
;6251  ;228  342        T1C:  SHF LEFT FWR[FT0] AND FQ IN[ZERO]
;6252  ;22A  093              MOV FWR0 TO FQ, EWR0 TO EQ
;6253  ;22C  347              MOV FQ TO FWR[FT2]
;6254  ;22E  1F1        T1D:  SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]   H.SUM.PREAL.LP
;6255  ;230  093              MOV FWR0 TO FQ, EWR0 TO EQ
;6256  ;232  347              MOV FQ TO FWR[FT2]
;6257  ;234  091        T1E:  OR EWR[ET1] WITH EWR[ET4]               D.PROD.NOT.0.RET
;6258  ;236  37A              MOV EWR[ET0] TO EWR[ET4]
;6259  ;238  3A8              MOV EWR[ET0] TO EWR[ET1]
;6260  ;23A  126              MOV EWR[ET4] TO EQ                    MUL.RT
;6261  ;23C  094              MOV EQ TO EWR[ET0]
;6262  ;23E  240        T1F:  ADD EWR[ET2] TO EWR[ET0]
;6263  ;240  168              MOV FWR[FT0] TO FWR[FT2]
;6264  ;242  280        T20:  INC EQ                                  DIVL.NEG1
;6265  ;244  093              MOV EWR[ET0] TO EQ
;6266  ;246  094              MOV EQ TO EWR[ET0]
;6267  ;248  0F4        T21:  DEC EQ                                  END.RND2
;6268  ;24A  093              MOV EWR[ET0] TO EQ
;6269  ;24C  094              MOV EQ TO EWR[ET0]
;6270  ;24E  23B        T22:  SUB EWR[ET5] FROM EQ TO EWR[ET0]
;6271  ;250  093              MOV EWR[0] TO EQ
;6272  ;252  392              MOV EWR[ET0] TO EWR[ET5]
;6273  ;254  1CE        T23:  SUB EQ FROM EWR[ET4] TO EWR[ET1]        S.D.OPT2.GT
;6274  ;256  093              MOV EWR[ET0] TO EQ
;6275  ;258  37A              MOV EWR[ET0] TO EWR[ET4]
;6276  ;25A  094              MOV EQ TO EWR[ET0]
;6277  ;25C  174              MOV EWR[ET1] TO EQ                    EMODG.ADJ.STORE.2
;6278  ;25E  0BC        T.24  CLR EWR[ET0]                          UN.FLO3
;6279  ;260  3C2        T37:  Branch on F47-16 EQ 0
;6280  ;262  209              MOV FT0 TO FT0
;6281  ;264  3C3        T38:  BRANCH ON F55-0 EQ 0
;6282  ;266  209              MOV FT0 TO FT0
;6283  ;268  144              CLR FT0
;6284  ;26A  123              SHIFT LEFT FWR[FT0] IN[ZERO]
;6285  ;26C  300        T39:  EXPECTED DATA
;6286  ;26E  325              NULL BRANCH
;6287  ;270  301              EXPECTED DATA ON CPU RCV DATA BRANCH
;6288  ;272  321              BRANCH ON CPU RCV DATA
;6289  ;274  300              EXPECTED DATA ON CPU RCV DATA BRANCH
;6290  ;276  3C4        T3A:  BRANHC ON F55-7 EQ 0
;6291  ;278  209              MOV FT0 TO FT0
;6292  ;27A  144              CLR FT0
;6293  ;27C  123              SHIFT LEFT FWR[FT0] IN[ZERO]
;6294  ;27E  0F2        T3B:  MOV ET0 TO ET0
;6295  ;280  342              SHIFT LEFT ET0
;6296  ;282  3C5              BRANCH ON EXPONENT=0 AND EXP15 F3
;6297  ;284  093        T3C:  MOV ET0 TO EQ
;6298  ;286  094              MOV EQ TO ET0 AND CLOCK OP2=0
;6299  ;288  3C6              BRANCH ON OP2=0 AND OP1=0
;6300  ;28A  210        T3D:  CALL #1 TO SUBROUTINE
;6301  ;28C  232              CALL #2 TO SUBROUTINE
```

```
;6302  ;28E   042          CALL #3 TO SUBROUTINE
;6303  ;290   122          CALL #4 TO SUBROUTINE
;6304  ;292   0C8          CALL #5 TO SUBROUTINE
;6305  ;294   3C7          RETURN
;6306  ;296   212          RETURN ADDRESS OF #1
;6307  ;298   121          RETURN ADDRESS OF #2
;6308  ;29A   344          RETURN ADDRESS OF #3
;6309  ;29C   088          RETURN ADDRESS OF #4
;6310  ;29E   200          RETURN ADDRESS OF #5
;6311  ;2A0   122    T3E:  CALL #4 TO SUBROUTINE
;6312  ;2A2   3CA          RETURN FROM SUBROUTINE(OP1+OP2)/=0
;6313  ;2A4   3D1          RETURN FROM SUBROUTINE(EXP15 F3)
;6314  ;2A6   093          MOV ET0 TO EQ
;6315  ;2A8   094          MOV EQ TO ET0 AND CLOCK OP2=0, OP1=0
;6316  ;2AA   342          SHIFT LEFT ET0
;6317  ;2AC   088          RETURN NUA
;6318  ;2AE   089          RETURN +1 NUA
;6319  ;2B0   08B          RETURN WHEN EXP15 F3 IS SET
;6320  ;2B2   0B9    T3F:  CLOCK SIGN OUT WITH 0
;6321  ;2B4   17C          CLOCK SIGN OUT WITH 1
;6322  ;2B6   021          CLOCK OP1 SIGN
;6323  ;2B8   018          CLOCK OP2 SIGN
;6324  ;2BA   3C8          BRANCH ON SUMPATH
;6325  ;2BC
;6326  ;2BE
;6327  ;2C0
;6328  ;2C2
;6329  ;2C4
;6330  ;2C6
;6331  ;2C8
;6332  ;2CA
;6333  ;2CC   3CD    T40:  EXT BRANCH ON INSTR BITS
;6334  ;2CE   3CE          EXT BRANCH ON THE SIZE BITS
;6335  ;2D0   3CF          EXT BRANCH ON DOUB OPER
;6336  ;2D2   3D0          EXT BRANCH ON INSTR BITS
;6337  ;2D4   168          MOV ET0 TO ET2
;6338  ;2D6   240          ADD ET0 TO ET2
;6339  ;2D8   093          MOV FWR[FT0] TO FQ
;6340  ;2DA   015          SHIFT FQ LEFT AND IN[ONE]
;6341  ;2DC   276    T9:   CLR FWR[FT0], CLR EWR[ET0]          H.EXCEP.TST
;6342  ;2DE   3A8    TF:   MOV WR[1]
;6343  ;2E0   398          STORE first WR[1]
;6344  ;2E2   3A0          STORE second WR[1]
;6345  ;2E4   168          MOV WR[2]
;6346  ;2E6   368          STORE first float WR[2]
;6347  ;2E8   370          STORE second float WR[2]
;6348  ;2EA   3B0          MOV WR[3]
;6349  ;2EC   36A          STORE first float WR[3]
;6350  ;2EE   372          STORE second float WR[3]
;6351  ;2F0   37A          MOV WR[4]
;6352  ;2F2   382          STORE first float WR[4]
;6353  ;2F4   38A          STORE second float WR[4]
;6354  ;2F6   392          MOV WR[5]
;6355  ;2F8   39A          STORE first float WR[5]
;6356  ;2FA   3A2          STORE second float WR[5]
```

```
;6357  ;2FC  3AA                    MOV WR[6]
;6358  ;2FE  3B2                    STORE first float WR[6]
;6359  ;300  36C                    STORE second float WR[6]
;6360  ;302  2E5                    MOV WR[7]
;6361  ;304  2E8                    STORE first float WR[7]
;6362  ;306  2E9                    STORE second float WR[7]
;6363  ;308  388                    MOV WR[8]
;6364  ;30A  378                    STORE first float WR[8]
;6365  ;30C  380                    STORE second float WR[8]
;6366  ;30E  2EC                    MOV WR[9]
;6367  ;310  2ED                    STORE first float WR[9]
;6368  ;312  2F0                    STORE second float WR[9]
;6369  ;314  2F1                    MOV WR[A]
;6370  ;316  2F4                    STORE first float WR[A]
;6371  ;318  2F5                    STORE second float WR[A]
;6372  ;31A  2F8                    MOV WR[B]
;6373  ;31C  2FA                    STORE first float WR[B]
;6374  ;31E  328                    STORE second float WR[B]
;6375  ;320  32A                    MOV WR[C]
;6376  ;322  32C                    STORE first float WR[C]
;6377  ;324  32D                    STORE second float WR[C]
;6378  ;326  330                    MOV WR[D]
;6379  ;328  332                    STORE first float WR[D]
;6380  ;32A  334                    STORE second float WR[D]
;6381  ;32C  336                    MOV WR[E]
;6382  ;32E  338                    STORE first float WR[E]
;6383  ;330  33A                    STORE second float WR[E]
;6384  ;332  33C                    MOV WR[F]
;6385  ;334  33E                    STORE first float WR[F]
;6386  ;336  340                    STORE second float WR[F]
;6387  ;338  074          T.10:     MOV FQ TO FWR[FT0]                    FET.G.DONE.2
;6388  ;33A  065                    STORE FWR[FT0]
;6389  ;33C  168                    MOV FT0 TO FWR[FT2]
;6390  ;33E  2DD                    STORE FWR[FT3] SEC[HUGE]
;6391  ;340  3A8                    MOV FT0 TO FWR[FT1]
;6392  ;342  122                    MOV FWR[FT1] TO FQ
;6393  ;344  37A                    MOV FT0 TO FWR[FT4]
;6394  ;346  2E4                    STORE FWR[FT5] SEC[HUGE]
;6395  ;348  3B0                    MOV FT0 TO FWR[FT3]
;6396  ;34A  011                    MOV FT3 TO FQ
;6397  ;34C  3AA                    MOV FT0 TO FT6
;6398  ;34E  350                    STORE FT7 SEC[HUGE]
;6399  ;350  392                    MOV FT0 TO FT5
;6400  ;352  195                    MOV FT5 TO FQ
;6401  ;354  388                    MOV FT0 TO FT8
;6402  ;356  355                    STORE FT9 SEC[HUGE]
;6403  ;358  2E5                    MOV FT0 TO FT7
;6404  ;35A  351                    MOV FT7 TO FQ
;6405  ;35C  2F1                    MOV FT0 TO FTA
;6406  ;35E  354                    STORE FTB SEC[HUGE]
;6407  ;360  2EC                    MOV FT0 TO FT9
;6408  ;362  18D                    MOV FT9 TO FQ
;6409  ;364  32A                    MOV FT0 TO FTC
;6410  ;366  358                    STORE FTD SEC[HUGE]
;6411  ;368  2F8                    MOV FT0 TO FTB
```

```
;6412  ;36A  35C                    MOV FTB TO FQ
;6413  ;36C  336                    MOV FT0 TO FTE
;6414  ;36E  359                    STORE FTF SEC[HUGE]
;6415  ;370  330                    MOV FT0 TO FTD
;6416  ;372  360                    MOV FTD TO FQ
;6417  ;374  33C                    MOV FT0 TO FTF
;6418  ;376  362                    MOV FTF TO FQ
;6419  ;378  342              T11:  SHIFT LEFT ET0
;6420  ;37A  168                    MOV ET0 TO ET2
;6421  ;37C  345                    SHIFT RIGHT ET2
;6422  ;37E  368                    STORE ET2
;6423  ;380  3AA                    MOV ET0 TO ET6
;6424  ;382  34A                    SHIFT ET6 RIGHT
;6425  ;384  3B2                    STORE ET6
;6426  ;386  3A8                    MOV ET0 TO ET1
;6427  ;388  346                    MOV ET1 TO ET6
;6428  ;38A  3B0                    MOV ET0 TO ET3
;6429  ;38C  35E                    MOV ET3 TO ET6
;6430  ;38E  37A                    MOV ET0 TO ET4
;6431  ;390  2C8                    MOV ET4 TO ET6
;6432  ;392  392                    MOV ET0 TO ET5
;6433  ;394  364                    MOV ET5 TO ET6
;6434  ;396  2E5                    MOV ET0 TO ET7
;6435  ;398  36D                    MOV ET7 TO ET6
;6436  ;39A  388                    MOV ET0 TO ET8
;6437  ;39C  043                    MOV ET8 TO ET6
;6438  ;39E  2EC                    MOV ET0 TO ET9
;6439  ;3A0  374                    MOV ET9 TO ET6
;6440  ;3A2  2F1                    MOV ET0 TO ETA
;6441  ;3A4  375                    MOV ETA TO ET6
;6442  ;3A6  2F8                    MOV ET0 TO ETB
;6443  ;3A8  37C                    MOV ETB TO ET6
;6444  ;3AA  32A                    MOV ET0 TO ETC
;6445  ;3AC  37E                    MOV ETC TO ET6
;6446  ;3AE  330                    MOV ET0 TO ETD
;6447  ;3B0  384                    MOV ETD TO ET6
;6448  ;3B2  336                    MOV ET0 TO ETE
;6449  ;3B4  272                    MOV ETE TO ET6
;6450  ;3B6  33C                    MOV ET0 TO ETF
;6451  ;3B8  38E                    MOV ETF TO ET6
;6452  ;3BA  083              T12:  SHIFT RIGHT FWR[FT0]
;6453  ;3BC  2B0              T41:  TOGGLE Q16 DEFAULT
;6454  ;3BE  04F                    SHIFT LEFT FQ
;6455  ;3C0  3BF                    BRANCH ON MULI1 AND FRAC55 Q3
;6456  ;3C2  093                    MOV FT0 TO FQ
;6457  ;3C4  2D0                    SHIFT RIGHT FQ
;6458  ;3C6  142              T42:  ADD SHFL FWR[FT1] TO FWR[FT3]
;6459  ;3C8  165                    ADD SHFL FWR[FT0] TO FWR[FT2] + FCOUT
;6460  ;3CA  143                    SUB SHFL FWR[FT1] TO FWR[FT3]
;6461  ;3CC  159                    SUB SHFL FWR[FT0] TO FWR[FT2] + FCOUT
;6462  ;3CE  342                    SHIFT LEFT FT0
;6463  ;3D0  093                    MOV FT0 TO FQ
;6464  ;3D2  3A8                    MOV FT0 TO FT1
;6465  ;3D4  168                    MOV FT0 TO FT2
;6466  ;3D6  3B0                    MOV FT0 TO FT3
```

```
;6467  ;3D8   3C1          BRANCH ON FRAC<55:32>=0 and DIV I3
;6468  ;3DA   168    T43:  MOV FT0 TO FT2
;6469  ;3DC   37A          MOV FT0 TO FT4
;6470  ;3DE   093          MOV FT0 TO FQ
;6471  ;3E0   102          MUL ADD FT2 TO FT4
;6472  ;3E2   382          STORE FF FT4
;6473  ;3E4   38A          STORE SF FT4
;6474  ;3E6   2E4          STORE TF FT4
;6475  ;3E8   342          SHIFT LEFT FT0 IN[ZERO]
;6476  ;3EA   1D8          SHIFT FT4 RIGHT
;6477  ;3EC   0B9          CLR FT0
;6478  ;3EE   006          SHIFT LEFT FT0 IN[ONE]
;6479  ;3F0   3A8          MOV FT0 TO FT1
;6480  ;3F2   3B0          MOV FT0 TO FT3
;6481  ;3F4   142          ADD.LOOP
;6482  ;3F6   143          SUB.LOOP
;6483  ;3F8   083          SHIFT RIGHT FT0 IN[EXP.SHF]
;6484  ;3FA   157    T26:  SHF RIGHT FWR[FT1] AND FQ IN[ONE]X
;6485  ;3FC   2D0          SHF RIGHT FWR[FT0] AND FQ IN[ZERO]X
;6486  ;3FE   09B          SHF RIGHT FWR[FT0] AND FQ IN[HUGE.ADJ]X
;6487                      ;SHF RIGHT EWR[FT0] AND EQ
;6488  ;400   398          STORE FIRST FLOAT FT1
;6489  ;402   3A0          STORE SECOND FLOAT FT1
;6490  ;404   065          STORE THIRD FLOAT FT1
;6491  ;406   3A8          MOV FT0 TO FT1
;6492  ;408   093          MOV FT0 TO FQ
;6493  ;40A   094          MOV FQ TO FT2
;6494  ;40C   1F1          SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]
;6495                      ;AND DEC EQ
;6496  ;40E   150          SHF RIGHT FWR[FT5] AND FQ IN[DELAY.ROT]
;6497  ;410   083          SHF RIGHT FWR[FT0] IN[EXP.SHF]
;6498  ;412   199          SHF RIGHT HUGE FWR[FT0] AND FQ
;6499  ;414   392          MOV FT0 FT5
;6500  ;416   3B2          STORE FIRST FLT FT6
;6501  ;418   36C          STORE SECOND FLT FT6
;6502  ;41A   350          STORE THIRD FLT FT7
;6503  ;41C   364          MOV FT5 TO FT6
;6504  ;41E   18C          ADD SHFR FWR[FT2] TO FWR[FT0] + FCOUT
;6505  ;420   168          MOV FT0 TO FT2
;6506  ;422   232          CLR FT0
;6507  ;424   336          MOV FT0 TO D.RND
;6508  ;426   338          STORE FIRST FLT FTE
;6509  ;428   33A          STORE SEC FLT FTE
;6510  ;42A   359          STORE THIRD FLT FTE
;6511  ;42C   015          SHF LEFT FWR[D.RND] AND FQ IN[ONE], DEC EQ
;6512  ;42E   330          MOV ET0 TO G.MAX
;6513  ;430   332          STORE FIRST FLT FTD
;6514  ;432   334          STORE SEC FLT FTD
;6515  ;434   358          STORE THIRD FLT FTD
;6516  ;436   008          SHF LEFT EWR[G.MAX] AND EQ IN[ONE]
;6517  ;438   37A          MOV FT0 TO FT4
;6518  ;43A   382          STORE FIRST FLT FT4
;6519  ;43C   38A          STORE SEC FLT FT4
;6520  ;43E   2E4          STORE THIRD FLT FT4
;6521  ;440   0F0          SHF LEFT FWR[FT4] IN[LF.ROT], DEC EQ
```

```
;6522   ;442          342                    SHF LEFT FT0, ET0 IN[ZERO] AND FQ, EQ IN[FRAC]
;6523   ;444          3AA                    MOV FT0 TO FT6
;6524   ;446          3B2                    STORE FIRST FLT FT6
;6525   ;448          36C                    STORE SEC FLT FT6
;6526   ;44A          350                    STORE THIRD FLT FT6
;6527   ;44C          07E                    SHF LEFT FT0, ET0 IN[DIV.SHF] AND FQ, EQ
;6528                                        ;IN[HUGE.FRAC]
;6529   ;44E          0B9            T27:    CLR FT0, CLOCK SIGN OUT WITH[0]
;6530   ;450          17C                    CLOCK SIGN OUT WITH[1]
;6531   ;452          021                    CLOCK OP1 SIGN
;6532   ;454          018                    CLOCK OP2 SIGN
;6533   ;456          0B2                    CLOCK SIGN OUT WITH [OP1.SIGN]
;6534   ;458          1B0                    CLOCK SIGN OUT WITH [OP2.SIGN]
;6535   ;45A          234                    CLOCK SIGN OUT WITH[OP1.XOR.OP2]
;6536   ;45C          224                    CLOCK SIGN OUT WITH[SO.XOR.OP1]
;6537   ;45E          098            T28:    CLOCK CC SET[V,C]
;6538   ;460          09A                    CLOCK CC SET[C]
;6539   ;462          3E7                    CLOCK CC SET[V]
;6540   ;464          189                    CLOCK CC
;6541   ;466          093                    MOV FT0 TO FQ, ET0 TO EQ
;6542   ;468          0A0                    STORE CC
;6543   ;46A          232                    CLR FT0, ET0
;6544   ;46C          0B9                    CLOCK SIGN OUT WITH[0]
;6545   ;46E          17C                    CLOCK SIGN OUT WITH[1]
;6546   ;470          003            T29:    LITERAL[1] TO EWR[ONE]
;6547   ;472          2EC                    MOV FT0 TO FT9, ET0 TO ET9
;6548   ;474          2ED                    STORE FIRST FLT FT9, ET9
;6549   ;476          031                    LITERAL[OFF] TO EWR[FD.MAX]
;6550   ;478          32A                    MOV FT0 TO FTC, ET0 TO ETC
;6551   ;47A          32C                    STORE FIRST FLT FTC
;6552   ;47C          262                    LITERAL[80] TO EWR[ET6]
;6553   ;47E          3AA                    MOV FT0 TO FT6, ET0 TO ET6
;6554   ;480          3B2                    STORE FIRST FLT FT6, ET6
;6555   ;482          0AF                    LITERAL[0A] TO EQ
;6556   ;484          093                    MOV FT0 TO FQ, ET0 TO EQ
;6557   ;486          094                    MOV FQ TO FT0, EQ TO ET0
;6558   ;488          3C7            T44:    RETURN
;6559   ;48A          3CE                    EXTENDED BRANCH ON SIZE BITS
;6560   ;48C          0040                   EXPECTED DATA BYTE AND INPUT BYTE
;6561   ;48E          0041
;6562   ;490          1842
;6563   ;492          1843
;6564   ;494          3044
;6565   ;496          3045
;6566   ;498          2846
;6567   ;49A          2847
;6568   ;49C          4048
;6569   ;49E          4849
;6570   ;4A0          504A
;6571   ;4A2          584B
;6572   ;4A4          804C
;6573   ;4A6          884D
;6574   ;4A8          904E
;6575   ;4AA          004F
;6576   ;4AC          1051
```

```
;6577   ;4AE          3854
;6578   ;4B0          2055
;6579   ;4B2          6856
;6580   ;4B4          0160
;6581   ;4B6          0161
;6582   ;4B8          1962
;6583   ;4BA          1963
;6584   ;4BC          3164
;6585   ;4BE          3165
;6586   ;4C0          2966
;6587   ;4C2          2967
;6588   ;4C4          4168
;6589   ;4C6          4969
;6590   ;4C8          516A
;6591   ;4CA          596B
;6592   ;4CC          816C
;6593   ;4CE          896D
;6594   ;4D0          916E
;6595   ;4D2          016F
;6596   ;4D4          1171
;6597   ;4D6          3974
;6598   ;4D8          2175
;6599   ;4DA          6176
;6600   ;4DC          C0C4
;6601   ;4DE          C0C5
;6602   ;4E0          D0C6
;6603   ;4E2          D0C7
;6604   ;4E4          7932
;6605   ;4E6          6233
;6606   ;4E8          0240
;6607   ;4EA          0241
;6608   ;4EC          1A42
;6609   ;4EE          1A43
;6610   ;4F0          3244
;6611   ;4F2          3245
;6612   ;4F4          2A46
;6613   ;4F6          2A47
;6614   ;4F8          4248
;6615   ;4FA          4A49
;6616   ;4FC          524A
;6617   ;4FE          5A4B
;6618   ;500          824C
;6619   ;502          8A4D
;6620   ;504          924E
;6621   ;506          024F
;6622   ;508          1251
;6623   ;50A          3A54
;6624   ;50C          2255
;6625   ;50E          7A56
;6626   ;510          0360
;6627   ;512          0361
;6628   ;514          1B62
;6629   ;516          1B63
;6630   ;518          3364
;6631   ;51A          3365
```

```
;6632   ;51C            2B66
;6633   ;51E            2B67
;6634   ;520            4368
;6635   ;522            4B69
;6636   ;524            536A
;6637   ;526            5B6B
;6638   ;528            836C
;6639   ;52A            8B6D
;6640   ;52C            936E
;6641   ;52E            036F
;6642   ;530            1371
;6643   ;532            3B74
;6644   ;534            2375
;6645   ;536            7376
;6646   ;538            7898
;6647   ;53A            7099
;6648   ;53C            63F6
;6649   ;53E            6BF7
;6650   ;540            343             T45:    Fast clock
;6651   ;542            3AA                     MOV FT0 TO FT6
;6652   ;544            3B2                     STORE FF FT6
;6653   ;546            36C                     STORE SF FT6
;6654   ;548            350                     STORE TF FT6
;6655
;6656   ;** INDICATES THIS ADDRESS MUST BE FIXED IN DAVE STONER'S CODE
;6657
;6658   .REGION/600,6FF
```

```
                ;6659   .PAGE   "WCS SUBROUTINES FOR CONSOLE USE"
                ;6660   .TOC    "  GENERATE TRANSFER DATA"
                ;6661   ;
                ;6662   ;       This routine is used by the diagnostic monitor to load WR 0 with a
                ;6663   ;       data pattern from the console.  The monitor will set CONSOLE ATTN if
                ;6664   ;       a 1 is to be generated, or clear CONSOLE ATTN if a 0 is to be gen-
                ;6665   ;       erated.  It will then single step this routine to shift the data bit
                ;6666   ;       into WR 0.  This routine loads a 32 bit value much more quickly than
                ;6667   ;       shifting each instruction into the CSR from the monitor, and is used
                ;6668   ;       mainly to set up data to load in LS for constants.
                ;6669   ;
                ;6670
                ;6671   ;                                       ***WARNING***
                ;6672   ;
                ;6673   ;       This routine must start at 600 and end at 605.  DO NOT move this
                ;6674   ;       routine to any other area!!
                ;6675   ;
                ;6676   ;
                ;6677
                ;6678   GEN.XFER.DATA:
U 0600, 2F80,15 ;6679           CLR WR[0]                       ;CLEAR A WORKING REG
U 0601, A0C0,15 ;6680           COM WR[0]                       ;NOW WR 0 IS ALL ONES
                ;6681   LOOP.XFER:
U 0602, 5B00,18 ;6682           SKIP.IF[CONSOLE.ATTN]           ;SKIP NEXT INSTRUCTION IS CONSOLE ATTN
                ;6683                                           ; IS SET (GENERATING A 1)
                ;6684           ASHL WR[0],                     ;SHIFT A 0 INTO BIT 0 OF WR 0
U 0603, A3D0,1E ;6685            SKIP                           ; AND SKIP ROL
U 0604, A3C0,15 ;6686           ROL WR[0]                       ;SHIFT A 1 INTO BIT 0 OF WR 0
U 0605, 0860,24 ;6687           JMP [LOOP.XFER]                 ;REPEAT
                ;6688
                ;6689
                ;6690   ;
                ;6691   ;       DIAGNOSTIC VERSION NUMBER AND SECTION NAME ARE STORED HERE
                ;6692   ;
                ;6693
                ;6694   ;                                       ***WARNING***
                ;6695   ;
                ;6696   ;       These words must be at location 606 and 607.  DO NOT move these words
                ;6697   ;       to any other area!!
                ;6698   ;
                ;6699   ;
                ;6700
U 0606, 0001,10 ;6701           WORD[0110]                      ;VERSION 01.10
U 0607, 0043,45 ;6702           ASCII [CE]                      ;LAST TWO LETTERS OF FILE NAME
                ;6703
                ;6704
```

```
                ;6705   .PAGE   '' DEPOSIT MCT CSR ROUTINE''
                ;6706   ;+++
                ;6707   ;
                ;6708   ; FUNCTIONAL DESCRIPTION:
                ;6709   ;
                ;6710   ;       The deposit to CSR routine is used to write the memory controller's
                ;6711   ;       control and status register.  The memory controller has three CSRs
                ;6712   ;       named CSR 0, CSR 1, and CSR 2.
                ;6713   ;         CSR 0 contains checkbits or syndrome bits returned from the ECC logic
                ;6714   ;       after certain diagnostic routines.  It is NOT writable directly with
                ;6715   ;       this routine.
                ;6716   ;         CSR 1 contains error bits set if an error occurs on a memory access.
                ;6717   ;       It does contain five maintenance bits (bits 29-25) which are writable.
                ;6718   ;       There are also seven checkbits (bits 6-0) which are writable, but not
                ;6719   ;       readable.  These bits are used to write checkbits into the ECC chips.
                ;6720   ;         CSR 2 contains error bits set if a UNIBUS error occurs on a UNIBUS
                ;6721   ;       access.  These are read only bits and are NOT writable by this routine.
                ;6722   ;         Therefore, CSR 1 is the only writable CSR, and only bits 29-25 can be
                ;6723   ;       both written and read back.  This routine will thus force CSR 1 on a
                ;6724   ;       deposit to CSR.
                ;6725   ;         This routine will write LS 5 with data to access CSR 1, then write
                ;6726   ;       the data residing in LS 6 into the CSR.  It will then issue CPU ATTN
                ;6727   ;       to inform the console that it has completed the function.
                ;6728   ;         NOTE:  The error bits of CSR 1 are cleared on any write to CSR 1.
                ;6729   ;       These are bits 31, 30 and 23-14.Bits 13-0 and bit 24 are unused.
                ;6730   ;
                ;6731   ; CALLING SEQUENCE:
                ;6732   ;
                ;6733   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;6734   ;       struction at WCS location 50) into the UPC and starts the CPU.
                ;6735   ;
                ;6736   ; INPUT PARAMETERS:
                ;6737   ;
                ;6738   ;       LS location 6 must have been written by the console with the desired
                ;6739   ;       data to be deposited in CSR 1 previous to executing this routine.
                ;6740   ;
                ;6741   ; IMPLICIT INPUTS:
                ;6742   ;
                ;6743   ;       None
                ;6744   ;
                ;6745   ; OUTPUT PARAMATERS:
                ;6746   ;
                ;6747   ;       CSR 1 bits 29-25 and 6-0 are written with data from LS 6.
                ;6748   ;
                ;6749   ; IMPLICIT OUTPUTS:
                ;6750   ;
                ;6751   ;       None
                ;6752   ;
                ;6753   ; SIDE EFFECTS:
                ;6754   ;
                ;6755   ;       WR 0 will be modified by this routine.
                ;6756   ;       LS 5 will be modified by this routine.
                ;6757   ;       CSR 1 bits 31, 30 and 23-14 are cleared.
                ;6758   ;
                ;6759   ;---
```

```
                 ;6760
                 ;6761   DEPOSIT.CSR:
                 ;6762
U 0608, 3644,15  ;6763           MOV LS[CSR1] TO WR[0]              ;SET BIT 2 IN WR 0 AS CSR NUMBER TO
                 ;6764                                              ; WRITE (BITS 2 AND 3 ARE CSR NUMBER)
U 0609, BE0A,15  ;6765           MOV WR[0] TO LS[T5.SUB]            ;PUT CSR NUMBER IN LS LOCATION 5 (CSR
                 ;6766                                              ; 1)
U 060A, 1D0B,F5  ;6767           MEM.REQ[WRITE.CSR] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                 ;6768                                              ; ACCESS CSR 1
U 060B, B20C,15  ;6769           WRITE.MEM LS[T6.SUB]               ;SEND DATA IN LS LOCATION 6 TO CSR 1
U 060C, 8868,74  ;6770           JMP [WAIT.SUB]                     ;JUMP TO SET ATTN AND WAIT
```

```
                ;6771   .PAGE   '' EXAMINE MCT CSR ROUTINE''
                ;6772   ;+++
                ;6773   ;
                ;6774   ; FUNCTIONAL DESCRIPTION:
                ;6775   ;
                ;6776   ;       The examine CSR routine reads the memory controller's control and
                ;6777   ;       status register.  There are 3 CSRs labeled CSR 0, CSR 1 and CSR 2.
                ;6778   ;         CSR 0 contains checkbits or syndrome bits from the ECC logic.
                ;6779   ;       These are contained in bits 6-0 of the CSR.  Other bits are UNPRE-
                ;6780   ;       DICTABLE.  Bits 6-0 are only written by special diagnostic routines
                ;6781   ;       (they are not writtable by the DEPOSIT CSR command).
                ;6782   ;         CSR 1 contains error bits and maintenance bits.  The error bits are
                ;6783   ;       31, 30, and 23-14.  Maintenance bits are 29-25.  The other bits are
                ;6784   ;       UNPREDICTABLE.  The error bits are written during memory functions
                ;6785   ;       only and are not writable via the DEPOSIT CSR command. The maintenance
                ;6786   ;       bits are writable.  Note that a DEPOSIT CSR command will write the
                ;6787   ;       maintenance bits and clear all error bits.
                ;6788   ;         CSR 2 contains three bits (bits 16-14) which are readable.  These are
                ;6789   ;       set when a UNIBUS error occurs on a UNIBUS access.  They are not
                ;6790   ;       directly writable with the DEPOSIT CSR command.
                ;6791   ;         The console loads LS 5 with the CSR number (0, 1, or 2) to be read.
                ;6792   ;       This routine rotates LS 5 two places left to position it correctly.
                ;6793   ;       It then executes a read CSR and places the data in LS 6 for the console
                ;6794   ;       to read.
                ;6795   ;
                ;6796   ; CALLING SEQUENCE:
                ;6797   ;
                ;6798   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;6799   ;       struction at WCS location 51) into the UPC and starts the CPU.
                ;6800   ;
                ;6801   ; INPUT PARAMETERS:
                ;6802   ;
                ;6803   ;       LS 5 contains the CSR number to be read.
                ;6804   ;
                ;6805   ; IMPLICIT INPUTS:
                ;6806   ;
                ;6807   ;       None
                ;6808   ;
                ;6809   ; OUTPUT PARAMATERS:
                ;6810   ;
                ;6811   ;       LS 6 - Data read from CSR selected.
                ;6812   ;
                ;6813   ; IMPLICIT OUTPUTS:
                ;6814   ;
                ;6815   ;       None
                ;6816   ;
                ;6817   ; SIDE EFFECTS:
                ;6818   ;
                ;6819   ;       LS 5 is rotated 2 places left.
                ;6820   ;
                ;6821   ;---
                ;6822
                ;6823   EXAMINE.CSR:
                ;6824
U 060D, 360A,15 ;6825           MOV LS[T5.SUB] TO WR[0]         ;GET CSR NUMBER FROM LS 5
```

```
U 060E, A2D0,15 ;6826           SHL2 WR[0]                          ;SHIFT NUMBER 2 PLACES LEFT TO GET
                ;6827                                               ; CSR NUMBER IN BITS 2 AND 3
U 060F, BE0A,15 ;6828           MOV WR[0] TO LS[T5.SUB]             ;PUT SHIFTED NUMBER IN LS 5
                ;6829
U 0610, 9D0B,75 ;6830           MEM.REQ[READ.CSR] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                ;6831                                               ; ACCESS CSR SPECIFIED
U 0611, 3022,15 ;6832           MOV MEM.DATA TO WR[0]               ;READ CSR SELECTED
                ;6833
                ;6834   CHECK.CSR2:
U 0612, B60A,95 ;6835           MOV LS[T5.SUB] TO WR[1]             ;GET SHIFTED CSR NUMBER
                ;6836           CMP LS[#8] WITH WR[1],              ;SEE IF CSR 2 WAS SELECTED
U 0613, CE46,B5 ;6837            DT(LONG)&SET.ALU.CC                ; SET UP CONDITION CODES
U 0614, 0861,61 ;6838           JMP.IF[NEQ] TO [CHECK.CSR1]         ;JUMP IF CSR 2 WAS NOT SELECTED
U 0615, C530,15 ;6839           BIC LS[CSR2.MASK] TO WR[0]          ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                ;6840                                               ; CSR 2.
                ;6841   CHECK.CSR1:
                ;6842           CMP LS[#4] WITH WR[1],              ;SEE IF CSR 1 WAS SELECTED
U 0616, 4E44,B5 ;6843            DT(LONG)&SET.ALU.CC                ; SET UP CONDITION CODES
U 0617, 0861,91 ;6844           JMP.IF[NEQ] TO [CHECK.CSR0]         ;JUMP IF CSR 1 WAS NOT SELECTED
U 0618, C52E,15 ;6845           BIC LS[CSR1.MASK] TO WR[0]          ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                ;6846                                               ; CSR 1.
                ;6847   CHECK.CSR0:
                ;6848           CMP LS[#0] WITH WR[1],              ;SEE IF CSR 0 WAS SELECTED
U 0619, 4E9C,B5 ;6849            DT(LONG)&SET.ALU.CC                ; SET UP CONDITION CODES
U 061A, 8861,D1 ;6850           JMP.IF[NEQ] TO [LOAD.LS6]           ;JUMP IF CSR 0 WAS NOT SELECTED
U 061B, 452C,15 ;6851           BIC LS[CSR0.MASK] TO WR[0]          ;CLEAR UNUSED (UNPREDICTABLE) BITS IN
                ;6852                                               ; CSR 0.
U 061C, C54E,15 ;6853           BIC LS[BIT7] TO WR[0]               ;BIT 7 IS ALSO UNUSED
                ;6854
                ;6855   LOAD.LS6:
U 061D, BE0C,15 ;6856           MOV WR[0] TO LS[T6.SUB]             ;LOAD CSR DATA INTO LS 6 FOR PRINTOUT
U 061E, 8868,74 ;6857           JMP [WAIT.SUB]                      ;ISSUE CPU ATTN AND WAIT FOR RESPONSE
```

```
                ;6858   .page   "  DEPOSIT MCT TRANSLATION BUFFER ROUTINE"
                ;6859   ;+++
                ;6860   ;
                ;6861   ; FUNCTIONAL DESCRIPTION:
                ;6862   ;
                ;6863   ;       This routine allows a write to the translation buffer portion of the
                ;6864   ;       memory controller.  The buffer area contains 128 decimal locations
                ;6865   ;       addressed from 0-7F(H).  The remainder of the TB RAM is either unused
                ;6866   ;       or contains the UNIBUS MAP.  The DEPOSIT UB command is used to write
                ;6867   ;       the UNIBUS map portion of the TB RAMs.  The address specified with the
                ;6868   ;       DEPOSIT command will be the actual RAM address accessed.  This routine
                ;6869   ;       will do any shifting necessary on the address specified to position it
                ;6870   ;       correctly.  The address specified has been loaded by the console in LS
                ;6871   ;       location 5 prior to executing this routine.  The address must be in HEX
                ;6872   ;       format.
                ;6873   ;         The data specified has been placed in LS 6 by the console prior to
                ;6874   ;       executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
                ;6875   ;       and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
                ;6876   ;       contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
                ;6877   ;       through 23 and bit 0 are unused. This routine will do any shifting nec-
                ;6878   ;       essary to write the bits in TB as described.  The data specified must
                ;6879   ;       be in HEX format.
                ;6880   ;
                ;6881   ; CALLING SEQUENCE:
                ;6882   ;
                ;6883   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;6884   ;       struction at WCS location 52) into the UPC and starts the CPU.
                ;6885   ;
                ;6886   ; INPUT PARAMETERS:
                ;6887   ;
                ;6888   ;       LS 5 - TB address to be written (range 0-7F)
                ;6889   ;       LS 6 - TB data to write (22 bits, from 1-22).  Bits 0 and 23-31 are
                ;6890   ;              ignored.
                ;6891   ;
                ;6892   ; IMPLICIT INPUTS:
                ;6893   ;
                ;6894   ;       None
                ;6895   ;
                ;6896   ; OUTPUT PARAMATERS:
                ;6897   ;
                ;6898   ;       TB will be written with specified data at specified address
                ;6899   ;
                ;6900   ; IMPLICIT OUTPUTS:
                ;6901   ;
                ;6902   ;       None
                ;6903   ;
                ;6904   ; SIDE EFFECTS:
                ;6905   ;
                ;6906   ;       WR 0 will be modified
                ;6907   ;       WR 1 will be modified
                ;6908   ;
                ;6909   ;---
                ;6910
                ;6911   DEPOSIT.TB:
U 061F, 8862,FC ;6912           JSR [SHIFT.TB.ADR]              ;GO TO SUBROUTINE TO POSITION TB AD-
```

```
                ;6913                                   ; DRESS CORRECTLY
U 0620, 360C,15 ;6914          MOV LS[T6.SUB] TO WR[0]  ;GET DATA FROM LS 6
U 0621, 4526,15 ;6915          BIC LS[#FF] TO WR[0]     ;CLEAR LOW BYTE OF DATA SPECIFIED.
                ;6916                                   ; LOW BYTE WILL BE PUT IN BITS 24-31
                ;6917                                   ; LATER
U 0622, 8862,AC ;6918          JSR [SHIFT.LOOP]         ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                ;6919                                   ; PLACES RIGHT
U 0623, E40C,15 ;6920          SWAP LS[T6.SUB] WITH WR[0] ;SWAP ORIGINAL DATA SPECIFIED WITH MOD-
                ;6921                                   ; IFIED DATA.  NOW LS 6 CONTAINS BITS
                ;6922                                   ; 8-22 IN BITS 0-14 AS REQUIRED.  WR 0
                ;6923                                   ; CONTAINS ORIGINAL DATA SPECIFIED
U 0624, 452C,15 ;6924          BIC LS[#FFFFFF00] TO WR[0] ;CLEAR ALL BUT LOW BYTE OF ORIGINAL
                ;6925                                   ; DATA SPECIFIED
U 0625, 8862,AC ;6926          JSR [SHIFT.LOOP]         ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                ;6927                                   ; PLACES RIGHT,  ON RETURN, WR 0 WILL
                ;6928                                   ; HAVE BITS 0-7 IN BITS 24-31, OTHER
                ;6929                                   ; BITS ARE CLEAR
U 0626, ED0C,15 ;6930          BIS WR[0] TO LS[T6.SUB]  ;NOW LS 6 CONTAINS DATA TO WRITE IN
                ;6931                                   ; TB IN CORRECT FORMAT  I.E. PA BITS
                ;6932                                   ; IN BITS 00-14 AND BYTE OFFSET, MODIFY
                ;6933                                   ; PROT A THROUGH PROT D, AND VALID IN
                ;6934                                   ; BITS 25-31.
                ;6935
U 0627, 980A,F5 ;6936          MEM.REQ[WRITE.TB] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO WRITE
                ;6937                                   ; THE TB
U 0628, B20C,15 ;6938          WRITE.MEM LS[T6.SUB]     ;WRITE DATA FROM LS 6 IN TB
U 0629, 8868,74 ;6939          JMP [WAIT.SUB]           ;JUMP TO SET ATTN AND WAIT
                ;6940
                ;6941   ;
                ;6942   ;       SUBROUTINE TO SHIFT WR 0 TO THE RIGHT 8 PLACES
                ;6943   ;
                ;6944   SHIFT.LOOP:
U 062A, 3646,95 ;6945          MOV LS[#8] TO WR[1]      ;COUNTER IN WR 1 TO SHIFT 8 PLACES
                ;6946   SHIFT.LOOP.1:
U 062B, 2340,15 ;6947          ROR WR[0]                ;SHIFT WR 0 ONE PLACE RIGHT
                ;6948          DEC WR[1],               ;DECREMENT COUNTER
U 062C, A102,B5 ;6949           DT(LONG)&SET.ALU.CC     ; AND SET CONDITION CODES
U 062D, 8862,B1 ;6950          JMP.IF[NEQ] TO [SHIFT.LOOP.1] ;REPEAT UNTIL 8 SHIFTS HAVE OCCURRED
U 062E, 5B00,14 ;6951          RETURN                   ;THEN RETURN
                ;6952   ;
                ;6953   ;       SUBROUTINE TO POSITION TB ADDRESS IN BITS 9-14 AND BIT 31 (TB ADDRESS
                ;6954   ;       BIT 0 MUST BE IN BIT 31).
                ;6955   ;
                ;6956   SHIFT.TB.ADR:
U 062F, 360A,15 ;6957          MOV LS[T5.SUB] TO WR[0]  ;GET TB ADDRESS FROM LS 5
U 0630, 452C,15 ;6958          BIC LS[#FFFFFF00] TO WR[0] ;CLEAR ALL BUT LOW BYTE (8 BIT ADDRESS)
U 0631, A2D0,15 ;6959          SHL2 WR[0]               ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0632, A2D0,15 ;6960          SHL2 WR[0]               ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0633, A2D0,15 ;6961          SHL2 WR[0]               ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0634, A2D0,15 ;6962          SHL2 WR[0]               ;SHIFT 2 PLACES LEFT TO POSITION ADDRESS
U 0635, 23D0,15 ;6963          ASHL WR[0]               ;SHIFT ONE MORE PLACE
                ;6964                                   ; NOW BITS 0-6 IN BITS 9-15
                ;6965          BIT LS[BIT15] WITH WR[0], ;SEE IF BIT 15 (MSB) IS SET OR CLEAR.
U 0636, 595E,35 ;6966           DT(LONG)&SET.ALU.CC     ; MSB MUST GO IN BIT 31 IN ORDER TO AD-
                ;6967                                   ; DRESS TB CORRECTLY
```

```
U 0637, 5B00,09 ;6968          SKIP.IF[EQL]                    ;SKIP NEXT INSTRUCTION IF MSB IS 0
U 0638, 477E,15 ;6969          BIS LS[BIT31] TO WR[0]          ;ELSE SET BIT 31 FOR MSB
                ;6970
                ;6971          MOV WR[0] TO LS[T5.SUB],        ;NOW TB ADDRESS IS IN CORRECT FORM IN
                ;6972                                          ; LS LOCATION 5
U 0639, 3E0A,14 ;6973           RETURN                         ; THEN RETURN TO MAIN CODE
```

```
                ;6974   .PAGE   "  EXAMINE TRANSLATION BUFFER ROUTINE"
                ;6975   ;+++
                ;6976   ;
                ;6977   ; FUNCTIONAL DESCRIPTION:
                ;6978   ;
                ;6979   ;       This routine will read the translation buffer of the memory controller.
                ;6980   ;       The TB will be returned with the byte offset, modify, prot A through
                ;6981   ;       prot D, and valid in bits 01-07 respectively.  PA bits 09-23 are re-
                ;6982   ;       turned in bits 08-22 respectively.  Bits 23-31 and bit 0 are not used
                ;6983   ;       and so will be cleared before printing the result.  The result will be
                ;6984   ;       put in LS 6 for the console to print.
                ;6985   ;         The routine will issue a memory request with memory function=READ.TB
                ;6986   ;       and data type=LONGWORD.  The console will load LS 5 with the address
                ;6987   ;       specified before executing this routine.  The routine will shift the
                ;6988   ;       address around as required to address the TB location specified.  The
                ;6989   ;       address specified must be in HEX format.
                ;6990   ;         The TB consists of 128 decimal locations (addresses from 0-7F).  The
                ;6991   ;       remaining locations in the TB RAM are either unused or used by the
                ;6992   ;       UNIBUS map.  UNIBUS map addresses are read via the EXAMINE UB routine.
                ;6993   ;
                ;6994   ; CALLING SEQUENCE:
                ;6995   ;
                ;6996   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;6997   ;       struction at WCS location 53) into the UPC and starts the CPU.
                ;6998   ;
                ;6999   ; INPUT PARAMETERS:
                ;7000   ;
                ;7001   ;       LS 5 - TB address (0-7F) in hex format
                ;7002   ;
                ;7003   ; IMPLICIT INPUTS:
                ;7004   ;
                ;7005   ;       None
                ;7006   ;
                ;7007   ; OUTPUT PARAMATERS:
                ;7008   ;
                ;7009   ;       LS 6 - TB data from address specified
                ;7010   ;
                ;7011   ; IMPLICIT OUTPUTS:
                ;7012   ;
                ;7013   ;       None
                ;7014   ;
                ;7015   ; SIDE EFFECTS:
                ;7016   ;
                ;7017   ;       LS 5 is modified
                ;7018   ;       WR 0 is modified
                ;7019   ;
                ;7020   ;---
                ;7021
                ;7022   EXAMINE.TB:
U 063A, 8862,FC ;7023           JSR [SHIFT.TB.ADR]              ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
                ;7024                                           ; CORRECT POSITION AND REPLACE IN LS 5
U 063B, 9C0B,F5 ;7025           MEM.REQ[READ.TB] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO READ
                ;7026                                           ; THE TB AT ADDRESS CONTAINED IN LS 5
U 063C, 3022,15 ;7027           MOV MEM.DATA TO WR[0]           ;PLACE RESULT IN WR 0
                ;7028
```

```
U 063D, 452A,15 ;7029           BIC LS[#FF000000] TO WR[0]      ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
                ;7030                                           ; IS NOT PART OF TB AND IS UNPREDICT-
                ;7031                                           ; ABLE)
U 063E, 456E,15 ;7032           BIC LS[BIT23] TO WR[0]          ;DITTO FOR BIT 23
U 063F, 4540,15 ;7033           BIC LS[BIT0] TO WR[0]           ;DITTO FOR BIT 0.  WR 0 NOW CONTAINS
                ;7034                                           ; RESULT TO PRINT
U 0640, BE0C,15 ;7035           MOV WR[0] TO LS[T6.SUB]         ;RESULT NOW IN LS 6
U 0641, 8868,74 ;7036           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
                ;7037   .PAGE   '' DEPOSIT UNIBUS MAP ROUTINE''
                ;7038   ;+++
                ;7039   ;
                ;7040   ; FUNCTIONAL DESCRIPTION:
                ;7041   ;
                ;7042   ;       This routine allows a write to the UNIBUS map portion of the TB on the
                ;7043   ;       memory controller.  The map area contains 512 decimal locations
                ;7044   ;       addressed from 200-3FF.  The remainder of the TB RAM is either unused
                ;7045   ;       or contains the Translation Buffer information.  The DEPOSIT TB command
                ;7046   ;       is used to write the TB portion of the TB RAMs.  The address specified
                ;7047   ;       with the DEPOSIT command will be the actual RAM address accessed.  This
                ;7048   ;       routine will do any shifting necessary on the address specified to pos-
                ;7049   ;       ition it correctly.  The address specified has been loaded by the con-
                ;7050   ;       sole in LS location 5 prior to executing this routine.  The address
                ;7051   ;       must be in HEX format.
                ;7052   ;         The data specified has been placed in LS 6 by the console prior to
                ;7053   ;       executing this routine.  Bits 07-01 will be the Valid, Prot, Modify,
                ;7054   ;       and Byte Offset bits (there are four Prot bits).  Bits 22-08 will
                ;7055   ;       contain the PA bits from PA 23 through PA 09 respectively.  Bits 31
                ;7056   ;       through 23 and bit 0 are unused. This routine will do any shifting nec-
                ;7057   ;       essary to write the bits in TB as described.  The data specified must
                ;7058   ;       be in HEX format.
                ;7059   ;
                ;7060   ; CALLING SEQUENCE:
                ;7061   ;
                ;7062   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;7063   ;       struction at WCS location 58) into the UPC and starts the CPU.
                ;7064   ;
                ;7065   ; INPUT PARAMETERS:
                ;7066   ;
                ;7067   ;       LS 5 - UNIBUS Map address in TB (must be in range 200-3FF HEX).
                ;7068   ;       LS 6 - Contains the data to write into the UNIBUS Map in bits 1-22.
                ;7069   ;
                ;7070   ; IMPLICIT INPUTS:
                ;7071   ;
                ;7072   ;       None
                ;7073   ;
                ;7074   ; OUTPUT PARAMATERS:
                ;7075   ;
                ;7076   ;       The UNIBUS map portion of the TB RAMs gets written with the data from
                ;7077   ;       LS 6 at the address specified in LS 5.
                ;7078   ;
                ;7079   ; IMPLICIT OUTPUTS:
                ;7080   ;
                ;7081   ;       None
                ;7082   ;
                ;7083   ; SIDE EFFECTS:
                ;7084   ;
                ;7085   ;       WR 0 will be modified
                ;7086   ;       WR 1 will be modified
                ;7087   ;
                ;7088   ;---
                ;7089
                ;7090   DEPOSIT.UBS:
U 0642, 0864,DC ;7091           JSR [SHIFT.UB.ADR]              ;GO TO SUBROUTINE TO ARRANGE UNIBUS
```

```
                    ;7092                                   ; MAP ADDRESS CORRECTLY
U 0643, 360C,15     ;7093           MOV LS[T6.SUB] TO WR[0]  ;GET DATA FROM LS 6
U 0644, 4526,15     ;7094           BIC LS[#FF] TO WR[0]     ;CLEAR LOW BYTE OF DATA SPECIFIED.
                    ;7095                                   ; LOW BYTE WILL BE PUT IN BITS 24-31
                    ;7096                                   ; LATER
U 0645, 8862,AC     ;7097           JSR [SHIFT.LOOP]         ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                    ;7098                                   ; PLACES RIGHT
U 0646, E40C,15     ;7099           SWAP LS[T6.SUB] WITH WR[0] ;SWAP ORIGINAL DATA SPECIFIED WITH MOD-
                    ;7100                                   ; IFIED DATA.  NOW LS 6 CONTAINS BITS
                    ;7101                                   ; 8-22 IN BITS 0-14 AS REQUIRED.  WR 0
                    ;7102                                   ; CONTAINS ORIGINAL DATA SPECIFIED
U 0647, 452C,15     ;7103           BIC LS[#FFFFFF00] TO WR[0] ;CLEAR ALL BUT LOW BYTE OF ORIGINAL
                    ;7104                                   ; DATA SPECIFIED
U 0648, 8862,AC     ;7105           JSR [SHIFT.LOOP]         ;GO TO SUBROUTINE TO SHIFT WR 0 EIGHT
                    ;7106                                   ; PLACES RIGHT.  ON RETURN, WR 0 WILL
                    ;7107                                   ; HAVE BITS 0-7 IN BITS 24-31, OTHER
                    ;7108                                   ; BITS ARE CLEAR
U 0649, ED0C,15     ;7109           BIS WR[0] TO LS[T6.SUB]  ;NOW LS 6 CONTAINS DATA TO WRITE IN
                    ;7110                                   ; TB IN CORRECT FORMAT  I.E. PA BITS
                    ;7111                                   ; IN BITS 00-14 AND BYTE OFFSET, MODIFY
                    ;7112                                   ; PROT A THROUGH PROT D, AND VALID IN
                    ;7113                                   ; BITS 25-31.
                    ;7114
U 064A, 1B0B,F5     ;7115           MEM.REQ[WRITE.UBS.MAP] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                    ;7116                                   ; WRITE THE UNIBUS MAP
U 064B, B20C,15     ;7117           WRITE.MEM LS[T6.SUB]     ;WRITE DATA FROM LS 6 IN UNIBUS MAP
U 064C, 8868,74     ;7118           JMP [WAIT.SUB]           ;JUMP TO SET ATTN AND WAIT
                    ;7119   ;
                    ;7120   ;       SUBROUTINE TO POSITION ADDRESS SPECIFIED CORRECTLY TO ADDRESS THE
                    ;7121   ;       UNIBUS MAP.
                    ;7122   ;
                    ;7123   SHIFT.UB.ADR:
U 064D, 360A,15     ;7124           MOV LS[T5.SUB] TO WR[0]  ;GET ADDRESS SPECIFIED IN WR 0
U 064E, A2D0,15     ;7125           SHL2 WR[0]               ;SHIFT ADDRESS 2 PLACES LEFT
U 064F, A2D0,15     ;7126           SHL2 WR[0]               ;SHIFT ADDRESS 2 PLACES LEFT
U 0650, A2D0,15     ;7127           SHL2 WR[0]               ;SHIFT ADDRESS 2 PLACES LEFT
U 0651, A2D0,15     ;7128           SHL2 WR[0]               ;SHIFT ADDRESS 2 PLACES LEFT
U 0652, 23D0,15     ;7129           ASHL WR[0]               ;SHIFT LEFT ONE MORE PLACE
                    ;7130                                   ; NOW ADDRESS IS IN BITS 9-17 OF WR 0
                    ;7131           MOV WR[0] TO LS[T5.SUB], ;NOW LS 5 CONTAINS CORRECT ADDRESS.
U 0653, 3E0A,14     ;7132            RETURN                  ; RETURN TO MAIN
                    ;7133
                    ;7134
```

```
                ;7135   .PAGE   "  EXAMINE UNIBUS MAP ROUTINE"
                ;7136   ;+++
                ;7137   ;       This routine will read the UNIBUS map portion of the memory controller.
                ;7138   ;       The UNIBUS MAP contents will be returned with the byte offset, modify,
                ;7139   ;       prot A through prot D, and valid in bits 01-07 respectively.  PA bits
                ;7140   ;       09-23 are returned in bits 08-22 respectively.  Bits 23-31 and bit 0
                ;7141   ;       are not used and so will be cleared before printing the result.  The
                ;7142   ;       result will be put in LS 6 for the console to print.
                ;7143   ;         The routine will issue a memory request with memory function=READ.UBS
                ;7144   ;       .MAP and data type=LONGWORD.  The console will load LS 5 with the ad-
                ;7145   ;       dress specified before executing this routine.  The routine will shift
                ;7146   ;       the address around as required to address the UNIBUS map location spec-
                ;7147   ;       ified.  The address specified must be in HEX format.
                ;7148   ;         The UNIBUS map consists of 512 decimal locations (addresses from 200-
                ;7149   ;       3FF)  The remaining locations in the TB RAM are either unused or used
                ;7150   ;       by the translation buffer.  TB contents are read via the EXAMINE TB
                ;7151   ;       routine.
                ;7152   ;
                ;7153   ; CALLING SEQUENCE:
                ;7154   ;
                ;7155   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;7156   ;       struction at WCS location 59) into the UPC and starts the CPU.
                ;7157   ;
                ;7158   ; INPUT PARAMETERS:
                ;7159   ;
                ;7160   ;       LS 5 - TB address in hex format
                ;7161   ;
                ;7162   ; IMPLICIT INPUTS:
                ;7163   ;
                ;7164   ;       None
                ;7165   ;
                ;7166   ; OUTPUT PARAMATERS:
                ;7167   ;
                ;7168   ;       LS 6 - UNIBUS map data from address specified
                ;7169   ;
                ;7170   ; IMPLICIT OUTPUTS:
                ;7171   ;
                ;7172   ;       None
                ;7173   ;
                ;7174   ; SIDE EFFECTS:
                ;7175   ;
                ;7176   ;       LS 5 is modified
                ;7177   ;       WR 0 will be modified
                ;7178   ;
                ;7179   ;---
                ;7180
                ;7181   EXAMINE.UBS:
U 0654, 0864,DC ;7182           JSR [SHIFT.UB.ADR]              ;SHIFT ADDRESS SPECIFIED FROM LS 5 INTO
                ;7183                                           ; CORRECT POSITION AND REPLACE IN LS 5
U 0655, 1C0B,75 ;7184           MEM.REQ[READ.UBS.MAP] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                ;7185                                           ; READ THE UBS MAP AT ADDRESS CONTAINED
                ;7186                                           ; IN LS 5
U 0656, 3022,15 ;7187           MOV MEM.DATA TO WR[0]           ;PLACE RESULT IN WR 0
                ;7188
U 0657, 452A,15 ;7189           BIC LS[#FF000000] TO WR[0]      ;CLEAR UPPER BYTE OF RESULT (UPPER BYTE
```

```
                ;7190                                   ; IS NOT PART OF TB AND IS UNPREDICT-
                ;7191                                   ; ABLE)
U 0658, 456E,15 ;7192          BIC LS[BIT23] TO WR[0]   ;DITTO FOR BIT 23
U 0659, 4540,15 ;7193          BIC LS[BIT0] TO WR[0]    ;DITTO FOR BIT 0.  WR 0 NOW CONTAINS
                ;7194                                   ; RESULT TO PRINT
U 065A, BE0C,15 ;7195          MOV WR[0] TO LS[T6.SUB]  ;RESULT NOW IN LS 6
U 065B, 8868,74 ;7196          JMP [WAIT.SUB]           ;JUMP TO SET ATTN AND WAIT
                ;7197                                   ; CONTROL
```

```
                ;7198   .PAGE   "  DEPOSIT MAIN MEMORY ROUTINE"
                ;7199   ;+++
                ;7200   ;
                ;7201   ; FUNCTIONAL DESCRIPTION:
                ;7202   ;
                ;7203   ;       This routine write a longword of data into main memory at the Physical
                ;7204   ;       address specified.  This address need not be on a longword boundary.
                ;7205   ;       The console will use this routine for data transfers to main memory as
                ;7206   ;       well as for the DEPOSIT MM command.  The console will have loaded the
                ;7207   ;       address into LS 5 and the data into LS 6 before executing this routine.
                ;7208   ;         The routine issues a memory request with the memory function=WRITE.P
                ;7209   ;       and the data type=longword.  It then issues a WRITE.MEM LS[6] instruc-
                ;7210   ;       tion to write the data into memory.  A check on ERROR SUM is made to
                ;7211   ;       assure that the write occured without an error.  If ERROR SUM is as-
                ;7212   ;       serted, the CPU will issue a CPU ACK before asserting CPU ATTN to in-
                ;7213   ;       form the console that an error occurred.
                ;7214   ;
                ;7215   ; CALLING SEQUENCE:
                ;7216   ;
                ;7217   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;7218   ;       struction at WCS location 54) into the UPC and starts the CPU.
                ;7219   ;
                ;7220   ; INPUT PARAMETERS:
                ;7221   ;
                ;7222   ;       LS 5 - Main memory physical address
                ;7223   ;       LS 6 - Main memory longword data
                ;7224   ;
                ;7225   ; IMPLICIT INPUTS:
                ;7226   ;
                ;7227   ;       None
                ;7228   ;
                ;7229   ; OUTPUT PARAMATERS:
                ;7230   ;
                ;7231   ;       Main Memory address specified is written with data from LS 6.
                ;7232   ;
                ;7233   ; IMPLICIT OUTPUTS:
                ;7234   ;
                ;7235   ;       A memory error asserts CPU ACK to inform the console of a write error.
                ;7236   ;
                ;7237   ; SIDE EFFECTS:
                ;7238   ;
                ;7239   ;       None
                ;7240   ;
                ;7241   ;---
                ;7242
                ;7243   DEPOSIT.MM:
U 065C, 990A,75 ;7244           MEM.REQ[WRITE.P] ADRS[T5.SUB] DT[LONG]  ;SET UP FOR WRITE TO MAIN
                ;7245                                                   ; MEMORY USING THE PHYSICAL ADDRESS
                ;7246                                                   ; STORED IN LS 5
                ;7247           WRITE.MEM LS[T6.SUB],                   ;WRITE THE DATA FROM LS 6 INTO MAIN
                ;7248                                                   ; MEMORY
U 065D, 320C,1D ;7249            SKIP.IF[MEM.REF.OK]                    ;SKIP NEXT INSTRUCTION IF NO MEMORY
                ;7250                                                   ; ERROR (NO ERR SUM)
U 065E, 10D0,15 ;7251           MISC [SET.CP.ACK]                       ;SET CPU ACK IF ERROR SUM ASSERTED
                ;7252
```

```
U 065F, 3644,15 ;7253          MOV LS[#4] TO WR[0]          ;PUT A 4 IN WR 0
U 0660, 6C0A,15 ;7254          ADD WR[0] TO LS[T5.SUB]      ;INCREMENT LS 5 FOR NEXT LONGWORD DEP
U 0661, 8868,74 ;7255          JMP [WAIT.SUB]               ;JUMP TO SET ATTN AND WAIT
```

```
                ;7256   .PAGE   "  EXAMINE MAIN MEMORY ROUTINE"
                ;7257   ;+++
                ;7258   ;
                ;7259   ; FUNCTIONAL DESCRIPTION:
                ;7260   ;
                ;7261   ;       This routine is used to read a longword form main memory at the Phy-
                ;7262   ;       sical address specified.  The address need not be on a longword bound-
                ;7263   ;       ary.  This routine is used by the console on an EXAMINE.MM command.
                ;7264   ;       The console will have loaded LS 5 with the physical address specified
                ;7265   ;       before executing this routine.
                ;7266   ;         The routine will first write CSR 1 to set the INH REP CRD (inhibit
                ;7267   ;       report correctable read data) bit to prevent an error sum from occuring
                ;7268   ;       on a single bit error which has been corrected.  Then the routine is-
                ;7269   ;       sues a memory request with memory function=READ.P and DT=longword.  It
                ;7270   ;       then reads the memory location and puts the result in LS 6.  It also
                ;7271   ;       checks for an ERROR SUM and issues a CPU ACK if an error occurred (sin-
                ;7272   ;       gle bit errors that have been corrected will not cause an ERR SUM).  If
                ;7273   ;       an error occurred, CPU ACK will be set before issueing CPU ATTN to in-
                ;7274   ;       form the console that an error occurred.
                ;7275   ;
                ;7276   ; CALLING SEQUENCE:
                ;7277   ;
                ;7278   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;7279   ;       struction at WCS location 55) into the UPC and starts the CPU.
                ;7280   ;
                ;7281   ; INPUT PARAMETERS:
                ;7282   ;
                ;7283   ;       LS 5 - Physical address in main memory to read.
                ;7284   ;
                ;7285   ; IMPLICIT INPUTS:
                ;7286   ;
                ;7287   ;       LS[INH.CRD] - Data to write into CSR 1 to inhibit error sum on a CRD.
                ;7288   ;       LS[CSR1]  - Address for writing CSR 1.
                ;7289   ;
                ;7290   ; OUTPUT PARAMATERS:
                ;7291   ;
                ;7292   ;       LS 6 - Longword data from physical memory address specified.
                ;7293   ;
                ;7294   ; IMPLICIT OUTPUTS:
                ;7295   ;
                ;7296   ;       CPU ACK if an error sum occurs.
                ;7297   ;
                ;7298   ; SIDE EFFECTS:
                ;7299   ;
                ;7300   ;       None
                ;7301   ;
                ;7302   ;---
                ;7303
                ;7304   EXAMINE.MM:
U 0662, 1D45,F5 ;7305           MEM.REQ[WRITE.CSR] ADRS[CSR1] DT[LONG]  ;ISSUE MEMORY REQUEST TO
                ;7306                                                   ; WRITE CSR 1
U 0663, B278,15 ;7307           WRITE.MEM LS[INH.CRD]                   ;SET INH REP CRD IN CSR 1 AND CLEAR
                ;7308                                                   ; ECC DISABLE
                ;7309
U 0664, 180B,F5 ;7310           MEM.REQ[READ.P] ADRS[T5.SUB] DT[LONG]  ;ISSUE MEMORY REQUEST TO
```

```
                 ;7311                                          ; READ LONGWORD DATA FROM MAIN MEMORY
                 ;7312                                          ; PHYSICAL ADDRESS IN LS 5
                 ;7313          MOV MEM.DATA TO LS[T6.SUB],     ;READ DATA AND PUT IN LS 6
U 0665, 380C,1D  ;7314           SKIP.IF[MEM.REF.OK]           ; SKIP NEXT INSTRUCTION IF NO MEMORY
                 ;7315                                          ; ERROR (NO ERROR SUM)
U 0666, 10D0,15  ;7316          MISC [SET.CP.ACK]              ;SET CPU ACK IF AN ERROR OCCURRED
U 0667, 8868,74  ;7317          JMP [WAIT.SUB]                 ;JUMP TO SET ATTN AND WAIT
```

```
                    ;7318   .page   "  EXAMINE IDC ROUTINES"
                    ;7319   ;+++
                    ;7320   ;
                    ;7321   ; FUNCTIONAL DESCRIPTION:
                    ;7322   ;
                    ;7323   ;       The following routines are used to read data from the optional IDC
                    ;7324   ;       module's registers.  The result is put in LS 6 for the console to
                    ;7325   ;       print out.
                    ;7326   ;
                    ;7327   ; CALLING SEQUENCE:
                    ;7328   ;
                    ;7329   ;       The console loads the address of a pointer to this routine (a JMP in-
                    ;7330   ;       struction at WCS location 5A through 5E) into the UPC and starts the
                    ;7331   ;       CPU.
                    ;7332   ;
                    ;7333   ; INPUT PARAMETERS:
                    ;7334   ;
                    ;7335   ;       None
                    ;7336   ;
                    ;7337   ; IMPLICIT INPUTS:
                    ;7338   ;
                    ;7339   ;       None
                    ;7340   ;
                    ;7341   ; OUTPUT PARAMATERS:
                    ;7342   ;
                    ;7343   ;       LS 6 - Data from IDC register specified.
                    ;7344   ;
                    ;7345   ; IMPLICIT OUTPUTS:
                    ;7346   ;
                    ;7347   ;       None
                    ;7348   ;
                    ;7349   ; SIDE EFFECTS:
                    ;7350   ;
                    ;7351   ;       None
                    ;7352   ;
                    ;7353   ;---
                    ;7354
                    ;7355   EXAMINE.ICSR:
U 0668, 9090,15     ;7356           MISC [SET.PORT.I.0]             ;SELECT PORT TO BUS
U 0669, 9780,15     ;7357           PORT.READ PORT.ADRS[CSR]        ;SEND THE READ COMMAND
U 066A, 0867,64     ;7358           JMP [READ.IDC]                  ;READ THE DATA
                    ;7359
                    ;7360   EXAMINE.IDAR:
U 066B, 9090,15     ;7361           MISC [SET.PORT.I.0]             ;SELECT PORT TO BUS
U 066C, 1784,15     ;7362           PORT.READ PORT.ADRS[DAR]        ;SEND THE READ COMMAND
U 066D, 0867,64     ;7363           JMP [READ.IDC]                  ;READ THE DATA
                    ;7364
                    ;7365   EXAMINE.DBUF:
U 066E, 9090,15     ;7366           MISC [SET.PORT.I.0]             ;SELECT PORT TO BUS
U 066F, 978C,15     ;7367           PORT.READ PORT.ADRS[DATA.WORD]  ;SEND THE READ COMMAND
U 0670, 0867,64     ;7368           JMP [READ.IDC]                  ;READ THE DATA
                    ;7369
                    ;7370   EXAMINE.PATT:
U 0671, 9090,15     ;7371           MISC [SET.PORT.I.0]             ;SELECT PORT TO BUS
U 0672, 1790,15     ;7372           PORT.READ PORT.ADRS[PATTERN]    ;SEND THE READ COMMAND
```

```
U 0673, 0867,64 ;7373            JMP [READ.IDC]                    ;READ THE DATA
                ;7374
                ;7375    EXAMINE.POSIT:
U 0674, 9090,15 ;7376            MISC [SET.PORT.I.0]               ;SELECT PORT TO BUS
U 0675, 9794,15 ;7377            PORT.READ PORT.ADRS[POSITION]     ;SEND THE READ COMMAND
                ;7378
                ;7379
                ;7380    READ.IDC:
U 0676, 3A0C,15 ;7381            MOV PORT TO LS[T6.SUB]            ;READ DATA AND PUT IN LS 6
U 0677, 1080,15 ;7382            MISC [SET.ACC.I.0]                ;RETURN SELECT TO ACCELERATOR
U 0678, 8868,74 ;7383            JMP [WAIT.SUB]                    ;JUMP TO SET ATTN AND WAIT
```

```
                    ;7384   .page   "  DEPOSIT IDC ROUTINES"
                    ;7385   ;+++
                    ;7386   ;
                    ;7387   ; FUNCTIONAL DESCRIPTION:
                    ;7388   ;
                    ;7389   ;       The following routines are used to write data to the optional IDC
                    ;7390   ;       module's registers.  The data is taken from LS 6 which is previously
                    ;7391   ;       loaded by the monitor when the DEPOSIT command is executed.
                    ;7392   ;
                    ;7393   ; CALLING SEQUENCE:
                    ;7394   ;
                    ;7395   ;       The console loads the address of a pointer to this routine (a JMP in-
                    ;7396   ;       struction at WCS location 5F through 61) into the UPC and starts the
                    ;7397   ;       CPU.
                    ;7398   ;
                    ;7399   ; INPUT PARAMETERS:
                    ;7400   ;
                    ;7401   ;       LS 6 - Data pattern to write.
                    ;7402   ;
                    ;7403   ; IMPLICIT INPUTS:
                    ;7404   ;
                    ;7405   ;       None
                    ;7406   ;
                    ;7407   ; OUTPUT PARAMATERS:
                    ;7408   ;
                    ;7409   ;       None
                    ;7410   ;
                    ;7411   ; IMPLICIT OUTPUTS:
                    ;7412   ;
                    ;7413   ;       None
                    ;7414   ;
                    ;7415   ; SIDE EFFECTS:
                    ;7416   ;
                    ;7417   ;       None
                    ;7418   ;
                    ;7419   ;---
                    ;7420
                    ;7421   DEPOSIT.ICSR:
U 0679, 360C,15     ;7422           MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 067A, 17A0,15     ;7423           PORT.WRITE PORT.ADRS[CSR] WITH WR[0]    ;WRITE TO IDC
U 067B, 8868,74     ;7424           JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                    ;7425
                    ;7426
                    ;7427   DEPOSIT.IDAR:
U 067C, 360C,15     ;7428           MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 067D, 97A4,15     ;7429           PORT.WRITE PORT.ADRS[DAR] WITH WR[0]    ;WRITE TO IDC
U 067E, 8868,74     ;7430           JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                    ;7431
                    ;7432
                    ;7433   DEPOSIT.DBUF:
U 067F, 360C,15     ;7434           MOV LS[T6.SUB] TO WR[0]          ;GET DATA PATTERN TO WRITE
U 0680, 17AC,15     ;7435           PORT.WRITE PORT.ADRS[DATA.WORD] WITH WR[0]      ;WRITE TO IDC
U 0681, 8868,74     ;7436           JMP [WAIT.SUB]                   ;JUMP TO SET ATTN AND WAIT
                    ;7437
                    ;7438
```

```
                    ;7439   CLEAR.FIFO.ADDR:
U 0682, 17C0,15     ;7440           PORT.CONTROL [CLEAR.FIFO.CNTR]  ;CLEAR ADDRESS IN FIFO A OR FIFO B
U 0683, 8868,74     ;7441           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                    ;7442
                    ;7443
                    ;7444   SELECT.FIFO.A:
U 0684, 17D8,15     ;7445           PORT.CONTROL [SEL.FIFO.A]       ;SELECT FIFO A
U 0685, 8868,74     ;7446           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
                    ;7447
                    ;7448
                    ;7449   SELECT.FIFO.B:
U 0686, 97DC,15     ;7450           PORT.CONTROL [SEL.FIFO.B]       ;SELECT FIFO B
                    ;7451
                    ;7452   WAIT.SUB:
U 0687, 10E0,15     ;7453           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;7454   WAIT.DE.EX:
U 0688, 8868,84     ;7455           JMP [WAIT.DE.EX]                ;LOOP SELF UNTIL CONSOLE TAKES CONTROL
```

```
                ;7456   .PAGE   '' SAVE WR ROUTINE''
                ;7457   ;+++
                ;7458   ;
                ;7459   ; FUNCTIONAL DESCRIPTION:
                ;7460   ;
                ;7461   ;       This routine saves WRs 0-3 in LS locations 1-4.  When the console takes
                ;7462   ;       control of the CPU, it calls this routine so that the working registers
                ;7463   ;       can be used by the console.  The WRs are restored by another routine
                ;7464   ;       called by the console before the CPU resumes execution (on a COntinue
                ;7465   ;       command, for example).
                ;7466   ;         This routine simply moves data from WR 0-3 to LS locations 1-4 and
                ;7467   ;       then issues a CPU ATTN to the console.  It then loops on a JMP self
                ;7468   ;       until the console takes over.
                ;7469   ;
                ;7470   ; CALLING SEQUENCE:
                ;7471   ;
                ;7472   ;       The console loads the address of a pointer to this routine (a JMP in-
                ;7473   ;       struction at WCS location 46) into the UPC and starts the CPU.
                ;7474   ;
                ;7475   ; INPUT PARAMETERS:
                ;7476   ;
                ;7477   ;       None
                ;7478   ;
                ;7479   ; IMPLICIT INPUTS:
                ;7480   ;
                ;7481   ;       None
                ;7482   ;
                ;7483   ; OUTPUT PARAMATERS:
                ;7484   ;
                ;7485   ;       LS 1 - Contents of WR 0
                ;7486   ;       LS 2 - Contents of WR 1
                ;7487   ;       LS 3 - Contents of WR 2
                ;7488   ;       LS 4 - Contents of WR 3
                ;7489   ;
                ;7490   ; IMPLICIT OUTPUTS:
                ;7491   ;
                ;7492   ;       None
                ;7493   ;
                ;7494   ; SIDE EFFECTS:
                ;7495   ;
                ;7496   ;       None
                ;7497   ;
                ;7498   ;---
                ;7499
                ;7500   SAVE.WR:
                ;7501
U 0689, 3E02,15 ;7502           MOV WR[0] TO LS[SAV.WR0]        ;SAVE WR 0 IN LS LOCATION 1
U 068A, BE04,95 ;7503           MOV WR[1] TO LS[SAV.WR1]        ;SAVE WR 1 IN LS LOCATION 2
U 068B, 3E07,15 ;7504           MOV WR[2] TO LS[SAV.WR2]        ;SAVE WR 2 IN LS LOCATION 3
U 068C, 3E09,95 ;7505           MOV WR[3] TO LS[SAV.WR3]        ;SAVE WR 3 IN LS LOCATION 4
U 068D, 8868,74 ;7506           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
                    ;7507   .PAGE   '' RESTORE WR ROUTINE''
                    ;7508   ;+++
                    ;7509   ;
                    ;7510   ; FUNCTIONAL DESCRIPTION:
                    ;7511   ;
                    ;7512   ;       This routine restores the 2901 working registers save in LS 1-4.  The
                    ;7513   ;       console calls this routine before the CPU is restarted after it has
                    ;7514   ;       been halted by the console.
                    ;7515   ;         The routine simply moves the data from LS locations 1-4 to WRs 0-3
                    ;7516   ;       and then issues a CPU ATTN to the console.  A JMP self is executed next
                    ;7517   ;       waiting for the console to take control.
                    ;7518   ;
                    ;7519   ; CALLING SEQUENCE:
                    ;7520   ;
                    ;7521   ;       The console loads the address of a pointer to this routine (a JMP in-
                    ;7522   ;       struction at WCS location 47) into the UPC and starts the CPU.
                    ;7523   ;
                    ;7524   ; INPUT PARAMETERS:
                    ;7525   ;
                    ;7526   ;       LS 1 contains the data to be restored in WR 0.
                    ;7527   ;       LS 2 contains the data to be restored in WR 1.
                    ;7528   ;       LS 3 contains the data to be restored in WR 2.
                    ;7529   ;       LS 4 contains the data to be restored in WR 3.
                    ;7530   ;
                    ;7531   ; IMPLICIT INPUTS:
                    ;7532   ;
                    ;7533   ;       None
                    ;7534   ;
                    ;7535   ; OUTPUT PARAMATERS:
                    ;7536   ;
                    ;7537   ;       WR 0 gets data from LS 1.
                    ;7538   ;       WR 1 gets data from LS 2.
                    ;7539   ;       WR 2 gets data from LS 3.
                    ;7540   ;       WR 3 gets data from LS 4.
                    ;7541   ;
                    ;7542   ; IMPLICIT OUTPUTS:
                    ;7543   ;
                    ;7544   ;       None
                    ;7545   ;
                    ;7546   ; SIDE EFFECTS:
                    ;7547   ;
                    ;7548   ;       None
                    ;7549   ;
                    ;7550   ;---
                    ;7551
                    ;7552   RESTORE.WR:
U 068E, B602,15     ;7553           MOV LS[SAV.WR0] TO WR[0]        ;RESTORE WR 0
U 068F, 3604,95     ;7554           MOV LS[SAV.WR1] TO WR[1]        ;RESTORE WR 1
U 0690, B607,15     ;7555           MOV LS[SAV.WR2] TO WR[2]        ;RESTORE WR 2
U 0691, B609,95     ;7556           MOV LS[SAV.WR3] TO WR[3]        ;RESTORE WR 3
U 0692, 8868,74     ;7557           JMP [WAIT.SUB]                  ;JUMP TO SET ATTN AND WAIT
```

```
;7558    .PAGE   "WCS SUBROUTINES FOR CPU USE"
;7559    .TOC    "  ERROR ROUTINE"
;7560    ;+++
;7561    ;
;7562    ; FUNCTIONAL DESCRIPTION:
;7563    ;
;7564    ;       This routine is used to detect and report diagnostic errors occurring
;7565    ;       during WCS based micro-diagnostics.  The WCS micro-code sets up the
;7566    ;       parameters listed below and calls this routine.  The routine compares
;7567    ;       WR 1 (expected data) and WR 0 (received data) according to the error
;7568    ;       mask.  Bits set in the mask indicate bits that are not checked for
;7569    ;       errors.
;7570    ;        If WR 0 and WR 1 are equal (no error) the routine executes a return+1
;7571    ;       to the calling point.  The only exception to this is if loop on error
;7572    ;       is set.  In that case, the error number is checked to see if that error
;7573    ;       occurred previously.  If so, then the error loop is forced by doing a
;7574    ;       return.  This will lock an error loop in on intermittent errors.
;7575    ;        If WR 0 and WR 1 are not equal (an error) then bit 8 is set in the
;7576    ;       CONTROL AND STATUS word, expected and received data is set up in LS for
;7577    ;       printout, and flags are checked in the ERROR CONTROL word.  This con-
;7578    ;       trols whether printouts are disabled, loop on error is enabled, etc.
;7579    ;       After printing the error, a return+1 is made to the calling point.
;7580    ;        Loop on error causes a return rather than return+1 to cause an error
;7581    ;       loop.  Also CPU ACKNOWLEDGE is toggled at the end of the routine for
;7582    ;       a scope sync point.
;7583    ;
;7584    ; CALLING SEQUENCE:
;7585    ;
;7586    ;       JSR [CHECK.RESULT]
;7587    ;
;7588    ; INPUT PARAMETERS:
;7589    ;
;7590    ;       WR[0] = Received data i.e. the actual result of the test
;7591    ;       WR[1] = Expected data i.e. what the result should have been
;7592    ;
;7593    ; IMPLICIT INPUTS:
;7594    ;
;7595    ;       LS[ERROR.MASK] = Used to mask out bits that are not to be tested
;7596    ;       LS[ERROR.NUMBER] = Current error number
;7597    ;       LS[PREVIOUS.ERROR] = Error number of previous data
;7598    ;       LS[CONTROL.STATUS] = Control and Status word.  Contains information
;7599    ;           written by the CPU to inform the monitor what to do.
;7600    ;       LS[ERROR.CONTROL] = Information such as loop-on-error, bell-on-error,
;7601    ;           no error report, and halt on error.
;7602    ;
;7603    ; OUTPUT PARAMATERS:
;7604    ;
;7605    ;       NONE
;7606    ;
;7607    ; IMPLICIT OUTPUTS:
;7608    ;
;7609    ;       LS[CONTROL.STATUS] = Control and Status with new status information
;7610    ;       LS[PREVIOUS.ERROR] = Last error number (modified only if error and
;7611    ;           loop on error is enabled)
;7612    ;       LS[EXPECTED.DATA] = Expected result if an error was detected
```

```
                ;7613   ;       LS[RECEIVED.DATA] = Actual result if an error was detected
                ;7614   ;
                ;7615   ; SIDE EFFECTS:
                ;7616   ;
                ;7617   ;       WR[0], WR[1], and the Q reg are modified and are not restored before
                ;7618   ;           returning.
                ;7619   ;       If loop-on-error is enabled and an error loop is to be executed, CPU
                ;7620   ;           ACKNOWLEDGE will be toggled for a scope sync.
                ;7621   ;
                ;7622   ;---
                ;7623
                ;7624   CHECK.RESULT:
U 0693, 458A,15 ;7625           BIC LS[ERROR.MASK] TO WR[0]     ;MASK OFF NOT TESTED BITS (CLEAR THEM)
                ;7626                                           ; FOR RECEIVED DATA
U 0694, C58A,95 ;7627           BIC LS[ERROR.MASK] TO WR[1]     ;DITTO FOR EXPECTED DATA
                ;7628
                ;7629           XOR WR[0] WITH WR[1] TO Q,      ;CMP RECEIVED DATA IN WR[0] WITH
U 0695, AB80,B5 ;7630            DT(LONG)&SET.ALU.CC            ; EXPECTED DATA IN WR[1] FOR ERROR
U 0696, 8869,F1 ;7631           JMP.IF[NEQ] TO [ERROR]          ;JUMP IF ERROR
                ;7632
U 0697, 3690,15 ;7633           MOV LS[ERROR.CONTROL] TO WR[0]  ;GET ERROR CONTROL WORD FROM LS
                ;7634           BIT WR[0] WITH LS[LOE],         ;SEE IF LOOP ON ERROR IS ENABLED
U 0698, 5940,35 ;7635            DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 0699, DB00,01 ;7636           SKIP.IF[BIT.SET]                ;SKIP IF IT IS
U 069A, DB00,16 ;7637           RETURN+1                        ;ELSE RETURN+1 (NO ERROR AND NO LOOP)
                ;7638
U 069B, 36A0,15 ;7639           MOV LS[PREVIOUS.ERROR] TO WR[0] ;IF NO ERROR BUT LOOP ON ERROR IS
                ;7640                                           ; ENABLED, SEE IF THERE WAS A
                ;7641                                           ; PREVIOUS ERROR
                ;7642           XOR WR[0] WITH LS[ERROR.NUMBER] TO Q,
U 069C, CD82,35 ;7643            DT(LONG)&SET.ALU.CC            ;IS THIS THE SAME SPOT THAT HAD AN
                ;7644                                           ; ERROR BEFORE? (INTERMITTANT ERROR)
U 069D, 086B,91 ;7645           JMP.IF[NEQ] TO [RET+1]          ;JUMP IF NOT TO RETURN WITHOUT ERROR
                ;7646                                           ; LOOPING (RETURN+1)
U 069E, 086B,C4 ;7647           JMP [RET]                       ;ELSE CONTINUE TO LOOP (WE WILL LOOP
                ;7648                                           ; ON ERROR EVEN IF IT GOES AWAY)
                ;7649
                ;7650   ERROR:
U 069F, 3E86,15 ;7651           MOV WR[0] TO LS[RECEIVED.DATA]  ;PUT RESULT OF TEST IN LS FOR PRINTOUT
U 06A0, 3690,15 ;7652           MOV LS[ERROR.CONTROL] TO WR[0]  ;GET ERROR CONROL BITS TO CHECK FOR
                ;7653                                           ; LOOP COMMAND
                ;7654           BIT WR[0] WITH LS[LOOP.COMMAND],;SEE IF BIT 6 SET
U 06A1, 594C,35 ;7655            DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 06A2, 086B,C1 ;7656           JMP.IF[BIT.SET] TO [RET]        ;GO LOOP IF SET (FAST LOOP)
                ;7657
U 06A3, 3E84,95 ;7658           MOV WR[1] TO LS[EXPECTED.DATA]  ;PUT EXPECTED DATA IN LS FOR PRINTOUT
                ;7659
U 06A4, 3680,95 ;7660           MOV LS[CONTROL.STATUS] TO WR[1] ;GET CONTROL AND STATUS WORD FROM LS
U 06A5, C532,95 ;7661           BIC LS[#FE7FFFFF] TO WR[1]      ;CLEAR ALL BUT BITS 23&24 IN CONTROL
                ;7662                                           ; AND STATUS WORD
U 06A6, C750,95 ;7663           BIS LS[ERROR] TO WR[1]          ;SET BIT 8 IN CONTROL AND STATUS WORD
                ;7664                                           ; (INDICATES AN ERROR WAS DETECTED)
U 06A7, BE80,95 ;7665           MOV WR[1] TO LS[CONTROL.STATUS] ;WRITE UPDATED CONTROL AND STATUS WORD
                ;7666                                           ; BACK TO LS[40]
                ;7667
```

```
                ;7668           BIT WR[0] WITH LS[BELL],          ;SEE IF BELL ON ERROR IS SET (WR 0 STILL
                ;7669                                             ; CONTAINS ERROR CONTROL WORD)
U 06A8, D944,35 ;7670            DT(LONG)&SET.ALU.CC              ; SET CONDITION CODES
U 06A9, 886A,D9 ;7671           JMP.IF[BITS.CLR] TO [CHECK.NER]  ;IF BELL ON ERROR IS NOT SET, JUMP TO
                ;7672                                             ; SEE IF ERROR REPORTING IS INHIBITED
U 06AA, 10E0,15 ;7673           MISC [SET.CP.ATTN]               ;ELSE SET CPU ATTN (RING MY BELL)
                ;7674
U 06AB, 086A,B4 ;7675   WAIt.1: JMP [WAIT.1]                     ;WAIT FOR CONSOLE TO STOP ME
U C6AC, 086B,64 ;7676           JMP [LOOP]                       ;GO TO CHECK LOOP-ON-ERROR AFTER
                ;7677                                             ; CONSOLE HAS FINISHED
                ;7678
                ;7679   CHECK.NER:
                ;7680           BIT WR[0] WITH LS[NER],          ;IF NO BELL SEE IF ERROR REPORT IS
                ;7681                                             ; INHIBITED
U 06AD, D942,35 ;7682            DT(LONG)&SET.ALU.CC              ; SET CONDITION CODES
U 06AE, 886B,21 ;7683           JMP.IF[BIT.SET] TO [CHECK.HALT]  ;JUMP IF ERROR REPORT IS INHIBITED TO
                ;7684                                             ; CHECK FOR HALT ON ERROR
                ;7685
U 06AF, 10E0,15 ;7686           MISC [SET.CP.ATTN]               ;SET CPU ATTN (REPORT ERROR)
U 06B0, 086B,04 ;7687   WAIT.2: JMP [WAIT.2]                     ;WAIT FOR CONSOLE TO STOP ME
U 06B1, 086B,64 ;7688           JMP [LOOP]                       ;GO TO CHECK LOOP-ON-ERROR AFTER
                ;7689                                             ; CONSOLE HAS FINISHED
                ;7690
                ;7691   CHECK.HALT:
                ;7692           BIT WR[0] WITH LS[HOE],          ;SEE IF HALT ON ERROR IS ENABLED
U 06B2, 5946,35 ;7693            DT(LONG)&SET.ALU.CC              ; SET CONDITION CODES
U 06B3, 886B,69 ;7694           JMP.IF[BITS.CLR] TO [LOOP]       ;JUMP IF NOT TO CHECK LOOP-ON-ERROR
                ;7695
U 06B4, 10E0,15 ;7696           MISC [SET.CP.ATTN]               ;ELSE SET CPU ATTN (HALT ON ERROR)
U 06B5, 086B,54 ;7697   WAIT.3: JMP [WAIT.3]                     ;WAIT FOR CONSOLE TO STOP ME
                ;7698
U 06B6, 3690,15 ;7699   LOOP:   MOV LS[ERROR.CONTROL] TO WR[0]    ;GET ERROR CONTROL BITS (NER, HOE ETC.)
                ;7700           BIT WR[0] WITH LS[LOE],          ;SEE IF LOOP-ON-ERROR
U 06B7, 5940,35 ;7701            DT(LONG)&SET.ALU.CC              ; SET CONDITION CODES
U 06B8, DB00,01 ;7702           SKIP.IF[BIT.SET]                 ; SKIP IF LOOP ON ERROR IS ENABLED
                ;7703
U 06B9, DB00,16 ;7704   RET+1:  RETURN+1                         ;ELSE RETURN+1 FOR NO ERROR LOOP
                ;7705
U 06BA, 3682,15 ;7706           MOV LS[ERROR.NUMBER] TO WR[0]    ;GET ERROR NUMBER IF LOOPING
                ;7707
U 06BB, BEA0,15 ;7708           MOV WR[0] TO LS[PREVIOUS.ERROR]  ;AND SAVE IT IN PREVIOUS ERROR
                ;7709
U 06BC, 10D0,15 ;7710   RET:    MISC [SET.CP.ACK]                ;SET CPU ACKNOWLEDGE FOR SCOPE SYNC
                ;7711           MISC [CLR.CP.ATTN.AND.ACK],      ;AND THEN CLEAR IT
U 06BD, 10C0,14 ;7712            RETURN                          ; RETURN TO TEST AND LOOP ON ERROR
                ;7713
```

```
                ;7714   .PAGE
                ;7715   ;
                ;7716   ;
                ;7717   ;
                ;7718   ERR.NO.TEST:
U 06BE, DB00,15 ;7719           NOP                                 ;NOP WILL CAUSE A SEQUENCE ERROR
U 06BF, B65E,15 ;7720           MOV LS[BEGIN.TEST] TO WR[0]         ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;7721                                               ; STATUS WORD
U 06C0, 3E80,15 ;7722           MOV WR[0] TO LS[CONTROL.STATUS]     ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;7723                                               ; BIT 15 INDICATES START OF TEST TO
                ;7724                                               ; CONSOLE
U 06C1, 8868,74 ;7725           JMP [WAIT.SUB]                      ;JUMP TO SET ATTN AND WAIT
                ;7726   .TOC    "START TRANSFER ROUTINE"
                ;7727   ;+++
                ;7728   ;
                ;7729   ;       This routine loads the LS and main memory with information neccesary
                ;7730   ;       for these tests to run.  It will then issue a CPU ATTN with the control
                ;7731   ;       and status word clear to indicate that all transfers are complete.
                ;7732   ;
                ;7733   ;---
                ;7734
                ;7735   TRANSFER.POINT:
                ;7736
U 06C2, 2F80,15 ;7737           CLR WR[0]                           ;START WITH 0 IN WR[0]
U 06C3, BFFE,15 ;7738           MOV WR[0] TO LS[PSL.HW]             ;MAKE SURE COMPAT MODE IS CLEAR
U 06C4, 2040,15 ;7739           INC WR[0]                           ;NOW HAVE 1 IN WR[0]
U 06C5, 23D0,15 ;7740           ASHL WR[0]                          ;0 IN BIT 0, 1 IN BIT 1
U 06C6, 2040,15 ;7741           INC WR[0]                           ;11(B)
U 06C7, 23D0,15 ;7742           ASHL WR[0]                          ;110(B)
U 06C8, 2040,15 ;7743           INC WR[0]                           ;111(B)
U 06C9, 23D0,15 ;7744           ASHL WR[0]                          ;1110(B)
U 06CA, 23D0,15 ;7745           ASHL WR[0]                          ;1 1100(B)
U 06CB, 23D0,15 ;7746           ASHL WR[0]                          ;11 1000(B)
U 06CC, 23D0,15 ;7747           ASHL WR[0]                          ;111 0000(B)  NOW HAVE 70(H) IN WR 0.
                ;7748                                               ; THIS IS CORRECT START ADDRESS OF DATA
                ;7749                                               ; SECTION (INCLUDING LONGWORD COUNT,
                ;7750                                               ; DESTINATION, AND DESTINATION ADDRESS)
U 06CD, 3E0E,15 ;7751           MOV WR[0] TO LS[TRANSFER.POINTER]   ;LOAD LS 7 WITH POINTER TO TRANSFER
                ;7752                                               ; DATA SECTION.
                ;7753
U 06CE, 2F80,15 ;7754           CLR WR[0]                           ;0 IN WR 0
U 06CF, 2040,15 ;7755           INC WR[0]                           ;1 IN WR 0
U 06D0, 23D0,15 ;7756           ASHL WR[0]                          ;10(B)
U 06D1, 23D0,15 ;7757           ASHL WR[0]                          ;100(B)
U 06D2, 23D0,15 ;7758           ASHL WR[0]                          ;1000(B)
U 06D3, 23D0,15 ;7759           ASHL WR[0]                          ;1 0000(B)
U 06D4, 23D0,15 ;7760           ASHL WR[0]                          ;10 0000(B)
U 06D5, 23D0,15 ;7761           ASHL WR[0]                          ;100 0000(B)
U 06D6, 23D0,15 ;7762           ASHL WR[0]                          ;1000 0000(B)
U 06D7, 23D0,15 ;7763           ASHL WR[0]                          ;1 0000 0000(B)
U 06D8, 23D0,15 ;7764           ASHL WR[0]                          ;10 0000 0000(B)
U 06D9, 23D0,15 ;7765           ASHL WR[0]                          ;100 0000 0000(B)
U 06DA, 23D0,15 ;7766           ASHL WR[0]                          ;1000 0000 0000(B)
U 06DB, 23D0,15 ;7767           ASHL WR[0]                          ;1 0000 0000 0000(B)
U 06DC, 23D0,15 ;7768           ASHL WR[0]                          ;10 0000 0000 0000(B)
```

```
U 06DD, 3E80,15 ;7769          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 13 IN LS CONTROL AND STATUS
                ;7770                                         ; WORD (BIT 13 INDICATES CONSOLE MUST
                ;7771                                         ; PERFORM A DATA TRANSFER)
U 06DE, 8870,0C ;7772          JSR [SET.ATTN]                 ;DO FIRST LS TRANSFER TO LOCATIONS 0
                ;7773                                         ; THROUGH 77(H)
                ;7774
                ;7775
U 06DF, 36CA,15 ;7776          MOV LS[#162(H)] TO WR[0]       ;GET START ADDRESS OF SECOND HALF OF LS
                ;7777                                         ; TRANSFER
U 06E0, 3E0E,15 ;7778          MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD START ADDRESS IN LS FOR CONSOLE
U 06E1, 365A,15 ;7779          MOV LS[XFER.DATA] TO WR[0]     ;BIT 13 INDICATES TO DO DATA TRANSFER
U 06E2, 3E80,15 ;7780          MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD
U 06E3, 8870,0C ;7781          JSR [SET.ATTN]                 ;DO SECOND LS TRANSFER TO LOCATIONS 80
                ;7782                                         ; THROUGH F7(H)
                ;7783
                ;7784
U 06E4, 3022,15 ;7785          MOV MEM.DATA TO WR[0]          ;THIS INSTRUCTION USED TO FREE MEMORY
                ;7786                                         ; IF HUNG IN A OCTA FLOW
U 06E5, 3022,15 ;7787          MOV MEM.DATA TO WR[0]          ;NEEDS 4 OF THESE
U 06E6, 3022,15 ;7788          MOV MEM.DATA TO WR[0]
U 06E7, 3022,15 ;7789          MOV MEM.DATA TO WR[0]
U 06E8, 9B9D,75 ;7790          MEM.REQ[READ.V.RCHK.IFILL] ADRS[ZERO] DT[LONG]
                ;7791                                         ;THIS INSTRUCTION USED TO FREE MEMORY
                ;7792                                         ; IF HUNG WAITING FOR A DATA RCVD
                ;7793
U 06E9, B70A,15 ;7794          MOV LS[#254] TO WR[0]          ;GET START ADDRESS OF MAIN MEMORY
                ;7795                                         ; TRANSFER DATA
U 06EA, 3E0E,15 ;7796          MOV WR[0] TO LS[TRANSFER.POINTER] ;LOAD START ADDRESS IN LS FOR CONSOLE
U 06EB, 8870,0C ;7797          JSR [SET.ATTN]                 ;DO MAIN MEMORY TRANSFER TO LOCATIONS
                ;7798                                         ; SPECIFIED IN DATA FILE
                ;7799
U 06EC, 17CC,15 ;7800          PORT.CONTROL [CLEAR.IDC]       ;JAM IDC TO IDLE
U 06ED, 17CC,15 ;7801          PORT.CONTROL [CLEAR.IDC]
U 06EE, 17CC,15 ;7802          PORT.CONTROL [CLEAR.IDC]
U 06EF, 17CC,15 ;7803          PORT.CONTROL [CLEAR.IDC]       ;DONE JAMMING IDC
                ;7804
U 06F0, B64E,95 ;7805          MOV LS[#80] TO WR[1]           ;SET BIT 7 OF WR 1
U 06F1, 97A0,95 ;7806          PORT.WRITE PORT.ADRS[CSR] WITH WR[1] ; CLEAR CRDY IN IDC
U 06F2, 17D4,15 ;7807          PORT.CONTROL [CLEAR.AUTO]      ;CLEAR AUTO MODE IN IDC
U 06F3, 97C4,15 ;7808          PORT.CONTROL [RESET.BR]        ;CLEAR BR 7 IN IDC
                ;7809
U 06F4, 1080,15 ;7810          MISC [SET.ACC.I.O]             ;SET FPA I/O MODE
                ;7811
U 06F5, 3660,15 ;7812          MOV LS[BIT16] TO WR[0]         ;SET PSL TO 7
U 06F6, C762,15 ;7813          BIS LS[BIT17] TO WR[0]
U 06F7, C764,15 ;7814          BIS LS[BIT18] TO WR[0]
U 06F8, 4766,15 ;7815          BIS LS[BIT19] TO WR[0]
U 06F9, C768,15 ;7816          BIS LS[BIT20] TO WR[0]
U 06FA, BFFE,15 ;7817          MOV WR[0] TO LS[PSL.HW]
U 06FB, 3660,15 ;7818          MOV LS[INTERUPT.EN] TO WR[0]   ;SET BIT 16 IN WR
U 06FC, C77A,15 ;7819          BIS LS[UBS.DCLO] TO WR[0]
U 06FD, 3E80,15 ;7820          MOV WR[0] TO LS[CONTROL.STATUS] ;SET IN CONTROL AND STATUS WORD TO
                ;7821                                         ; FORCE CLEAR ANY INTERRUPTS
U 06FE, 0870,04 ;7822          JMP [SET.ATTN]                 ;DONE. INFORM CONSOLE
```

```
                    ;7823   .PAGE
                    ;7824   .REGION/700,3FFF
                    ;7825   ; SUBROUTINE FOR LOADING FPA ADDRESSES INTO WR[0] OR WR[1]
                    ;7826   ;
                    ;7827   ;       This is a three line subroutine that is used to move the address that's
                    ;7828   ;       going to be forced in the FPA from Main Memory to either WR[0] or WR[1]
                    ;7829   ;       Since either WR[0] or WR[1] can be forced there will be two subroutines
                    ;7830
                    ;7831
                    ;7832   SET.ATTN:
U 0700, 10E0,15     ;7833           MISC [SET.CP.ATTN]                  ;CALL CONSOLE
                    ;7834   WAIT.XFER:
U 0701, 8870,14     ;7835           JMP [WAIT.XFER]                     ;WAIT FOR CONSOLE TO RESPOND
U 0702, 5B00,14     ;7836           RETURN                              ;BACK TO CALLING PROGRAM
                    ;7837
                    ;7838
U 0703, 9813,B5     ;7839   L0:     MEM.REQ[READ.P] ADRS[T9] DT[WORD]       ;Accesses the memory location
                    ;7840                                                   ; defined by the contents of T9
U 0704, 3022,15     ;7841           MOV MEM.DATA TO WR[0]                   ;Load the contents of the mem-
                    ;7842                                                   ; location accessed into WR[0].
U 0705, FF12,15     ;7843           INC LS[T9]                              ;Update the pointer to memory.
U 0706, FF12,15     ;7844           INC LS[T9]
U 0707, 5B00,14     ;7845           RETURN
                    ;7846
U 0708, 9813,B5     ;7847   L1:     MEM.REQ[READ.P] ADRS[T9] DT[WORD]       ;Accesses the memory location
                    ;7848                                                   ; defined by the contents of T9
U 0709, B022,95     ;7849           MOV MEM.DATA TO WR[1]                   ;Load the contents of the mem-
                    ;7850                                                   ; location accessed into WR[1].
U 070A, FF12,15     ;7851           INC LS[T9]                              ;Update the pointer to memory.
U 070B, FF12,15     ;7852           INC LS[T9]
U 070C, 5B00,14     ;7853           RETURN
                    ;7854
U 070D, 9813,B5     ;7855   -L0:    MEM.REQ[READ.P] ADRS[T9] DT[WORD]       ;Accesses the memory location
                    ;7856                                                   ; defined by th contents of T9
U 070E, 3022,15     ;7857           MOV MEM.DATA TO WR[0]                   ;Load the contents of the mem-
                    ;7858                                                   ; ory location accessed into 0
U 070F, FC12,15     ;7859           DEC LS[T9]                              ;Update pointer to memory
U 0710, FC12,15     ;7860           DEC LS[T9]
U 0711, 5B00,14     ;7861           RETURN
                    ;7862
U 0712, 9813,B5     ;7863   T84:    MEM.REQ[READ.P] ADRS[T9] DT[WORD]       ;Accesses the memory location
                    ;7864                                                   ; defined by the contents of T9
U 0713, B908,15     ;7865           MOV MEM.DATA TO LS[T84]                  ;Load the contents of the mem-
                    ;7866                                                   ; location accessed into WR[1].
U 0714, FF12,15     ;7867           INC LS[T9]                              ;Update the pointer to memory.
U 0715, FF12,15     ;7868           INC LS[T9]
U 0716, 5B00,14     ;7869           RETURN
                    ;7870
                    ;7871
                    ;7872
                    ;7873   ; Subroutine for generating data for WR test
                    ;7874   ;
                    ;7875   ;       This subroutine will generate the 7 patterns required to test the 64
                    ;7876   ;       bits of fraction WR and exponent WR.
                    ;7877
```

```
                    ;7878   GET.DATA:
U 0717, 2045,15     ;7879           INC WR[2]                       ;Increment pointer to next pattern
                    ;7880           CMP WR[2] WITH LS[#0],          ;Is the pointer pointing to the 1ST
U 0718, CF9D,35     ;7881            DT(LONG)&SET.ALU.CC            ; pattern?
U 0719, 0871,E1     ;7882           JMP.IF[NEQ] TO [GD1]            ;If it isn't then go onto GD1
U 071A, B698,15     ;7883           MOV LS[#AAAAAAAA] TO WR[0]      ;Move pattern AAAAAAAA to WR 0
U 071B, 3EA2,15     ;7884           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 071C, 3EA4,15     ;7885           MOV WR[0] TO LS[DATA.2]
U 071D, 5B00,14     ;7886           RETURN                          ;Return to calling routine
                    ;7887   GD1:    CMP WR[2] WITH LS[#1],          ;Is the pointer pointing to this
U 071E, 4F41,35     ;7888            DT(LONG)&SET.ALU.CC            ; pattern?
U 071F, 0872,41     ;7889           JMP.IF[NEQ] TO [GD2]            ;If it isn't then go to GD2
U 0720, 369A,15     ;7890           MOV LS[#55555555] TO WR[0]      ;Move pattern to WR 0
U 0721, 3EA2,15     ;7891           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 0722, 3EA4,15     ;7892           MOV WR[0] TO LS[DATA.2]
U 0723, 5B00,14     ;7893           RETURN                          ;Return to calling routine
                    ;7894   GD2:    CMP WR[2] WITH LS[#2],          ;Is the pointer pointing to this
U 0724, CF43,35     ;7895            DT(LONG)&SET.ALU.CC            ; pattern?
U 0725, 8872,A1     ;7896           JMP.IF[NEQ] TO [GD3]            ;If it isn't then go to GD3
U 0726, B6C4,15     ;7897           MOV LS[#33333333] TO WR[0]      ;Move pattern to WR 0
U 0727, 3EA2,15     ;7898           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 0728, 3EA4,15     ;7899           MOV WR[0] TO LS[DATA.2]
U 0729, 5B00,14     ;7900           RETURN                          ;Return to calling routine
                    ;7901   GD3:    CMP WR[2] WITH LS[#3(H)],       ;Is the pointer pointing to this
U 072A, CFC1,35     ;7902            DT(LONG)&SET.ALU.CC            ; pattern?
U 072B, 0873,01     ;7903           JMP.IF[NEQ] TO [GD4]            ;If it isn't then go to GD4
U 072C, 36C6,15     ;7904           MOV LS[#0F0F0F0F] TO WR[0]      ;Move pattern to WR 0
U 072D, 3EA2,15     ;7905           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 072E, 3EA4,15     ;7906           MOV WR[0] TO LS[DATA.2]
U 072F, 5B00,14     ;7907           RETURN                          ;Return to calling routine
                    ;7908   GD4:    CMP WR[2] WITH LS[#4],          ;Is the pointer pointing to this
U 0730, CF45,35     ;7909            DT(LONG)&SET.ALU.CC            ; pattern?
U 0731, 0873,61     ;7910           JMP.IF[NEQ] TO [GD5]            ;If it isn't then go to GD5
U 0732, B6C8,15     ;7911           MOV LS[#00FF00FF] TO WR[0]      ;Move pattern to WR 0
U 0733, 3EA2,15     ;7912           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 0734, 3EA4,15     ;7913           MOV WR[0] TO LS[DATA.2]
U 0735, 5B00,14     ;7914           RETURN                          ;Return to calling routine
U 0736, B712,15     ;7915   GD5:    MOV LS[#5] TO WR[0]             ;Is the pointer pointing to this
                    ;7916           CMP WR[2] WITH WR[0],           ; pattern?
U 0737, 2644,35     ;7917            DT(LONG)&SET.ALU.CC
U 0738, 8873,E1     ;7918           JMP.IF[NEQ] TO [GD6]            ;If it isn't then go to GD6
U 0739, 8873,E4     ;7919           JMP [GD6]                       ; else go onto next pattern
U 073A, 3628,15     ;7920           MOV LS[#FFFF] TO WR[0]          ;Move pattern to WR 0
U 073B, 3EA2,15     ;7921           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1 and DATA.2
U 073C, 3EA4,15     ;7922           MOV WR[0] TO LS[DATA.2]
U 073D, 5B00,14     ;7923           RETURN                          ;Return to calling routine
U 073E, 369C,15     ;7924   GD6:    MOV LS[#0] TO WR[0]             ;Move first half of data to WR 0
U 073F, 3EA2,15     ;7925           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1
U 0740, B69E,15     ;7926           MOV LS[#FFFFFFFF] TO WR[0]      ;Move second half of data to WR 0
U 0741, 3EA4,15     ;7927           MOV WR[0] TO LS[DATA.2]         ;Move WR 0 to DATA.2
U 0742, 5B00,14     ;7928           RETURN                          ;Return to calling routine
                    ;7929
                    ;7930   ; CLEAR INSTRUCTION ENCODE AND SIZE BITS
                    ;7931   ;
                    ;7932   ;       This routine sets the instruction encode bits and the size bits in the
```

```
                ;7933   ;       FPA to zero by enabling bits 10 - 17 onto the FPA BUS and through a
                ;7934   ;       muxed input into the instruction encode and size bits. Bit 18 of the
                ;7935   ;       FPA BUS enables the MUX on a force instruction.
                ;7936
                ;7937   CLR.INST.AND.SIZE.BITS:
U 0743, B718,15 ;7938           MOV LS[#00040300] TO WR[0]      ;Get address to force
U 0744, 90F6,15 ;7939           MISC2 [TRAP.ACC] WR[0]          ;Force address
U 0745, 5B00,14 ;7940           RETURN                          ;Return
                ;7941
                ;7942   ; GET PATTERN FOR UPPER EXPONENT BITS
                ;7943   ;
                ;7944   ;       This subroutine sets up the data that will be used to test the upper
                ;7945   ;       eight bits of the exponent data path
                ;7946
                ;7947   GET.PATTERN:
U 0746, 2045,15 ;7948           INC WR[2]                       ;Increment pointer to pattern
                ;7949           CMP WR[2] WITH LS[#1],          ;Is the pointer pointing to this
U 0747, 4F41,35 ;7950            DT(LONG)&SET.ALU.CC            ; pattern?
U 0748, 8874,C1 ;7951           JMP.IF[NEQ] TO [PATTERN.1]      ;If it isn't then go to next pattern
U 0749, 369C,15 ;7952           MOV LS[#0] TO WR[0]             ;Move pattern to WR 0
U 074A, 3EA2,15 ;7953           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1
U 074B, 5B00,14 ;7954           RETURN                          ;Return to calling routine
                ;7955   PATTERN.1:
                ;7956           CMP WR[2] WITH LS[#2],          ;Is the pointer pointing to this
U 074C, CF43,35 ;7957            DT(LONG)&SET.ALU.CC            ; pattern?
U 074D, 8875,11 ;7958           JMP.IF[NEQ] TO [PATTERN.2]      ;If it isn't then go to next pattern
U 074E, B69E,15 ;7959           MOV LS[#FFFFFFFF] TO WR[0]      ;Move pattern to WR 0
U 074F, 3EA2,15 ;7960           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1
U 0750, 5B00,14 ;7961           RETURN                          ;Return to calling routine
                ;7962   PATTERN.2:
                ;7963           CMP WR[2] WITH LS[#3(H)],       ;Is the pointer pointing to this
U 0751, CFC1,35 ;7964            DT(LONG)&SET.ALU.CC            ; pattern?
U 0752, 0875,61 ;7965           JMP.IF[NEQ] TO [PATTERN.3]      ;If it isn't then go to next pattern
U 0753, 365C,15 ;7966           MOV LS[#4000] TO WR[0]          ;Move pattern to WR 0
U 0754, 3EA2,15 ;7967           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1
U 0755, 5B00,14 ;7968           RETURN                          ;Return to calling routine
                ;7969   PATTERN.3:
U 0756, B71E,15 ;7970           MOV LS[#FFFFBFFF] TO WR[0]      ;Move pattern to WR 0
U 0757, 3EA2,15 ;7971           MOV WR[0] TO LS[DATA.1]         ;Move WR 0 to DATA.1
U 0758, 5B00,14 ;7972           RETURN                          ;Return to calling routine
                ;7973   ; Shift Routines
                ;7974   ;
                ;7975   ;       In order to test the upper bits of the exponent data path shifting in
                ;7976   ;       and out of those registers is required. There are four routines, they
                ;7977   ;       are to shift left four times, to shift right four times, to shift left
                ;7978   ;       eight times and to shift right eight times.
                ;7979   SHIFT.LEFT.4:
U 0759, 2F87,95 ;7980           CLR WR[3]                       ;Clear counter
U 075A, B616,15 ;7981           MOV LS[T11] TO WR[0]            ;Move address of shf left to WR 0
U 075B, B716,95 ;7982           MOV LS[#300] TO WR[1]           ;Move address of loop self to WR 1
                ;7983   SHIFT.LEFT.4.LOOP:
U 075C, 90F6,15 ;7984           MISC2 [TRAP.ACC] WR[0]          ;Force shift left
U 075D, 10F6,95 ;7985           MISC2 [TRAP.ACC] WR[1]          ;Force loop self
U 075E, 2047,95 ;7986           INC WR[3]                       ;Add one to the counter
                ;7987           CMP WR[3] WITH LS[#4],          ;Is the counter equal to 4?
```

```
U 075F, 4F45,B5 ;7988            DT(LONG)&SET.ALU.CC
U 0760, 0875,C1 ;7989            JMP.IF[NEQ] TO [SHIFT.LEFT.4.LOOP]
                ;7990                                    ;If it isn't then loop
U 0761, 5B00,14 ;7991            RETURN                  ;Return to calling routine
                ;7992
                ;7993    SHIFT.LEFT.8:
U 0762, 2F87,95 ;7994            CLR WR[3]               ;Clear counter
U 0763, B616,15 ;7995            MOV LS[T11] TO WR[0]    ;Move address of shf left instr to WR 0
U 0764, B716,95 ;7996            MOV LS[#300] TO WR[1]   ;Move address of loop self to WR 1
                ;7997    SHIFT.LEFT.8.LOOP:
U 0765, 90F6,15 ;7998            MISC2 [TRAP.ACC] WR[0]  ;Force shift left
U 0766, 10F6,95 ;7999            MISC2 [TRAP.ACC] WR[1]  ;Force loop self
U 0767, 2047,95 ;8000            INC WR[3]               ;Add one to the counter
                ;8001            CMP WR[3] WITH LS[#8],  ;Is thecounter equal to 8?
U 0768, CF47,B5 ;8002             DT(LONG)&SET.ALU.CC
U 0769, 0876,51 ;8003            JMP.IF[NEQ] TO [SHIFT.LEFT.8.LOOP]
                ;8004                                    ;If is isn't then loop
U 076A, 5B00,14 ;8005            RETURN                  ;Return to calling routine
                ;8006
                ;8007    SHIFT.RIGHT.4:
U 076B, 2F87,95 ;8008            CLR WR[3]               ;Clear counter
U 076C, B610,15 ;8009            MOV LS[T8] TO WR[0]     ;Move address of shift right to WR 0
U 076D, B716,95 ;8010            MOV LS[#300] TO WR[1]   ;Move address of loop self to WR 1
                ;8011    SHIFT.RIGHT.4.LOOP:
U 076E, 90F6,15 ;8012            MISC2 [TRAP.ACC] WR[0]  ;Force shift right
U 076F, 10F6,95 ;8013            MISC2 [TRAP.ACC] WR[1]  ;Force loop self
U 0770, 2047,95 ;8014            INC WR[3]               ;Add one to the counter
                ;8015            CMP WR[3] WITH LS[#4],  ;Is the counter equal to 4?
U 0771, 4F45,B5 ;8016             DT(LONG)&SET.ALU.CC
U 0772, 8876,E1 ;8017            JMP.IF[NEQ] TO [SHIFT.RIGHT.4.LOOP]
                ;8018                                    ;If it isn't then loop
U 0773, 5B00,14 ;8019            RETURN                  ;Return to calling routine
                ;8020
                ;8021    SHIFT.RIGHT.8:
U 0774, 2F87,95 ;8022            CLR WR[3]               ;Clear counter
U 0775, B610,15 ;8023            MOV LS[T8] TO WR[0]     ;Move address of shift right to WR 0
U 0776, B716,95 ;8024            MOV LS[#300] TO WR[1]   ;Move address of loop self to WR 1
                ;8025    SHIFT.RIGHT.8.LOOP:
U 0777, 90F6,15 ;8026            MISC2 [TRAP.ACC] WR[0]  ;Force shift right
U 0778, 10F6,95 ;8027            MISC2 [TRAP.ACC] WR[1]  ;Force loop self
U 0779, 2047,95 ;8028            INC WR[3]               ;Add one to counter
                ;8029            CMP WR[3] WITH LS[#8],  ;Is the counter equal to 8?
U 077A, CF47,B5 ;8030             DT(LONG)&SET.ALU.CC
U 077B, 0877,71 ;8031            JMP.IF[NEQ] TO [SHIFT.RIGHT.8.LOOP]
                ;8032                                    ;If it isn't then loop
U 077C, 5B00,14 ;8033            RETURN                  ;return to calling routine
                ;8034
                ;8035    ; LOAD ROUTINE, STORE AND MOVE ROUTINES
                ;8036    ;
                ;8037    ;       Since it is frequently necessary to load and store data to and from the
                ;8038    ;       FPA a single routine to do this will minimize the code necessary to do
                ;8039    ;       the loading and storing.
                ;8040    LOAD.DATA:
U 077D, 3614,15 ;8041            MOV LS[T10] TO WR[0]    ;Move address of Load first flt to WR 0
U 077E, 90F6,15 ;8042            MISC2 [TRAP.ACC] WR[0]  ;Force load first float
```

```
U 077F, B6A2,15 ;8043          MOV LS[DATA.1] TO WR[0]          ;Move DATA.1 to WR 0
U 0780, 10F4,15 ;8044          MISC2 [ASSERT.DA.AND.PASS.WR]    ;Pass data in WR 0 to the FPA
U 0781, 5B00,14 ;8045          RETURN                           ;Return to calling routine
                ;8046
                ;8047  STORE.DATA.1:
U 0782, B616,15 ;8048          MOV LS[T11] TO WR[0]             ;Move address of store first flt to WR -
U 0783, 90F6,15 ;8049          MISC2 [TRAP.ACC] WR[0]           ;Force store first float
U 0784, BA1A,15 ;8050          MOV FPA TO LS[T13]               ;Enable the FPA BUS to the CPU
U 0785, 5B00,14 ;8051          RETURN                           ;Return to calling routine
                ;8052
                ;8053  STORE.DATA.2:
U 0786, 3618,15 ;8054          MOV LS[T12] TO WR[0]             ;Mov address of store first flt to WR 0
U 0787, 90F6,15 ;8055          MISC2 [TRAP.ACC] WR[0]           ;Force store first float
U 0788, BA1A,15 ;8056          MOV FPA TO LS[T13]               ;Enable FPA BUS to the CPU
U 0789, 5B00,14 ;8057          RETURN                           ;Return to calling routine
                ;8058
                ;8059  MOV.DATA:
U 078A, B61C,15 ;8060          MOV LS[T14] TO WR[0]             ;Move address of mov ftx to fty to WR 0
U 078B, B716,95 ;8061          MOV LS[#300] TO WR[1]            ;Move address of loop self
U 078C, 90F6,15 ;8062          MISC2 [TRAP.ACC] WR[0]           ;Force move ftx to fty
U 078D, 10F6,95 ;8063          MISC2 [TRAP.ACC] WR[1]           ;Force loop self
U 078E, 5B00,14 ;8064          RETURN                           ;Return to calling routine
                ;8065
                ;8066
                ;8067
                ;8068  ; TRIPLE LOAD AND TRIPLE STORE ROUTINES
                ;8069  ;
                ;8070  ;       These routines allow you to load to the three possible floats and
                ;8071  ;       store from the three different floats
                ;8072
                ;8073
                ;8074  LOAD.TRIPLE:
U 078F, B72E,15 ;8075          MOV LS[#2A7] TO WR[0]            ;ADDRESS OF LOAD DOUBLE DIAG U-ROUTINE
U 0790, 90F6,15 ;8076          MISC2 [TRAP.ACC] WR[0]           ;FORCE ADDRESS AND EXECUTE.  LOOP IN
                ;8077                                           ; U-CODE WAITING FOR CPUD DATA AVAIL
U 0791, B6A2,15 ;8078          MOV LS[DATA.1] TO WR[0]          ;SET UP FIRST FLOAT DATA
U 0792, 36A4,95 ;8079          MOV LS[DATA.2] TO WR[1]          ;SET UP SECOND FLOAT DATA
U 0793, 10F4,15 ;8080          MISC2 [ASSERT.DA.AND.PASS.WR]    ;PASS FIRST FLOAT DATA
U 0794, BEA4,95 ;8081          MOV WR[1] TO LS[DATA.2]          ;PUT SECOND FLOAT DATA ON Y BUS TO FPA
                ;8082                                           ; WR 0
                ;8083
U 0795, B730,15 ;8084          MOV LS[#2DC] TO WR[0]            ;ADDRESS OF HUGE LOAD DIAG U-ROUTINE
U 0796, 90F6,15 ;8085          MISC2 [TRAP.ACC] WR[0]           ;FORCE ADDRESS AND EXECUTE.
U 0797, 36A6,15 ;8086          MOV LS[DATA.3] TO WR[0]          ;SET UP THIRD FLOAT DATA
U 0798, 10F4,15 ;8087          MISC2 [ASSERT.DA.AND.PASS.WR]    ;PASS THIRD FLOAT DATA TO FPA WR 0
                ;8088
U 0799, 5B00,14 ;8089          RETURN                           ;DONE LOADING 56 FRAC BITS, 8 EXPONENT
                ;8090                                           ; BITS AND 8 EXTENSION BITS
                ;8091
                ;8092  LOAD.DOUBLE:
U 079A, B72E,15 ;8093          MOV LS[#2A7] TO WR[0]            ;ADDRESS OF LOAD DOUBLE DIAG U-ROUTINE
U 079B, 90F6,15 ;8094          MISC2 [TRAP.ACC] WR[0]           ;FORCE ADDRESS AND EXECUTE.  LOOP IN
                ;8095                                           ; U-CODE WAITING FOR CPUD DATA AVAIL
U 079C, B6A2,15 ;8096          MOV LS[DATA.1] TO WR[0]          ;SET UP FIRST FLOAT DATA
U 079D, 36A4,95 ;8097          MOV LS[DATA.2] TO WR[1]          ;SET UP SECOND FLOAT DATA
```

```
U 079E, 10F4,15 ;8098           MISC2 [ASSERT.DA.AND.PASS.WR]   ;PASS FIRST FLOAT DATA
U 079F, BEA4,95 ;8099           MOV WR[1] TO LS[DATA.2]         ;PUT SECOND FLOAT DATA ON Y BUS TO FPA
                ;8100                                           ; WR 0
U 07A0, 5B00,14 ;8101           RETURN
                ;8102
                ;8103   STORE.2.TRIPLE:
U 07A1, 3732,15 ;8104           MOV LS[#368] TO WR[0]           ;FPA ADDRESS OF FIRST FLOAT STORE
U 07A2, 90F6,15 ;8105           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07A3, BAA8,15 ;8106           MOV FPA TO LS[FIRST.FLT]        ;STORE IN LS
                ;8107
U 07A4, 3734,15 ;8108           MOV LS[#370] TO WR[0]           ;FPA ADDRESS OF SECOND FLOAT STORE
U 07A5, 90F6,15 ;8109           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07A6, 3AAA,15 ;8110           MOV FPA TO LS[SEC.FLT]          ;STORE IN LS
                ;8111
U 07A7, B736,15 ;8112           MOV LS[#2DD] TO WR[0]           ;FPA ADDRESS OF THIRD (HUGE) FLOAT STORE
U 07A8, 90F6,15 ;8113           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07A9, 3AAC,15 ;8114           MOV FPA TO LS[THIRD.FLT]        ;STORE IN LS
                ;8115
U 07AA, 5B00,14 ;8116           RETURN                          ;DONE GETTING DATA FROM FPA (56 BITS OF
                ;8117                                           ; FRAC, 8 BITS OF EXPONENT, 8 BITS OF EXT)
                ;8118
                ;8119   STORE.X.TRIPLE:
U 07AB, B6AE,15 ;8120           MOV LS[T57] TO WR[0]            ;FPA ADDRESS OF FIRST FLOAT STORE
U 07AC, 90F6,15 ;8121           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07AD, BAA8,15 ;8122           MOV FPA TO LS[FIRST.FLT]        ;STORE IN LS
                ;8123
U 07AE, B6B0,15 ;8124           MOV LS[T58] TO WR[0]            ;FPA ADDRESS OF SECOND FLOAT STORE
U 07AF, 90F6,15 ;8125           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07B0, 3AAA,15 ;8126           MOV FPA TO LS[SEC.FLT]          ;STORE IN LS
                ;8127
U 07B1, 36B2,15 ;8128           MOV LS[T59] TO WR[0]            ;FPA ADDRESS OF THIRD (HUGE) FLOAT STORE
U 07B2, 90F6,15 ;8129           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 2
U 07B3, 3AAC,15 ;8130           MOV FPA TO LS[THIRD.FLT]        ;STORE IN LS
                ;8131
U 07B4, 5B00,14 ;8132           RETURN                          ;DONE GETTING DATA FROM FPA (56 BITS OF
                ;8133                                           ; FRAC, 8 BITS OF EXPONENT, 8 BITS OF EXT)
                ;8134
                ;8135   STORE.0.TRIPLE:
U 07B5, 37BA,15 ;8136           MOV LS[#11D] TO WR[0]           ;FPA ADDRESS OF FIRST FLOAT STORE
U 07B6, B716,95 ;8137           MOV LS[#300] TO WR[1]
U 07B7, 90F6,15 ;8138           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 0
U 07B8, BAA8,15 ;8139           MOV FPA TO LS[FIRST.FLT]        ;STORE IN LS
U 07B9, 10F6,95 ;8140           MISC2 [TRAP.ACC] WR[1]
                ;8141
U 07BA, 37BC,15 ;8142           MOV LS[#3BA] TO WR[0]           ;FPA ADDRESS OF SECOND FLOAT STORE
U 07BB, 90F6,15 ;8143           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 0
U 07BC, 3AAA,15 ;8144           MOV FPA TO LS[SEC.FLT]          ;STORE IN LS
                ;8145
U 07BD, B7BE,15 ;8146           MOV LS[#3BC] TO WR[0]           ;FPA ADDRESS OF THIRD (HUGE) FLOAT STORE
U 07BE, 90F6,15 ;8147           MISC2 [TRAP.ACC] WR[0]          ;EXECUTE STORE FROM FPA WR 0
U 07BF, 3AAC,15 ;8148           MOV FPA TO LS[THIRD.FLT]        ;STORE IN LS
                ;8149
U 07C0, 5B00,14 ;8150           RETURN                          ;DONE GETTING DATA FROM FPA (56 BITS OF
                ;8151                                           ; FRAC, 8 BITS OF EXPONENT, 8 BITS OF EXT)
                ;8152
```

```
                    ;8153   ;
                    ;8154   ;SUBROUTINE TO CLEAR DATA IN FTO
                    ;8155   ;
                    ;8156
                    ;8157   CLR.FTO:
U 07C1, E5A2,15     ;8158           CLR LS[DATA.1]                  ;LOAD DATA1 WITH ZEROS
U 07C2, E5A4,15     ;8159           CLR LS[DATA.2]                  ;LOAD DATA2 WITH ZEROS
U 07C3, 65A6,15     ;8160           CLR LS[DATA.3]                  ;LOAD DATA3 WITH ZEROS
U 07C4, 0878,FC     ;8161           JSR [LOAD.TRIPLE]               ;LOAD ZERO DATA IN FTO
U 07C5, 5B00,14     ;8162           RETURN                          ;DONE
                    ;8163
                    ;8164   ; CHECK SUB
                    ;8165   ;
                    ;8166   ;       This routine checks the three floats of data from the FPA. The
                    ;8167   ;       expected and received data for the three floats is passed from
                    ;8168   ;       the calling routine, everything else is setup internal to the
                    ;8169   ;       routine.
                    ;8170   ;
                    ;8171   ;
                    ;8172   CHECK.SUB:
U 07C6, B640,15     ;8173           MOV LS[BIT0] TO WR[0]                   ;Set up other data
U 07C7, BE88,15     ;8174           MOV WR[0] TO LS[ADDRESS.DATA]
U 07C8, BE8B,15     ;8175           MOV WR[2] TO LS[ERROR.MASK]             ;Setup error mask
U 07C9, B6A8,15     ;8176           MOV LS[FIRST.FLT] TO WR[0]              ;Setup received data
U 07CA, 36A2,95     ;8177           MOV LS[DATA.1] TO WR[1]                 ;Setup expected data
U 07CB, 0869,3C     ;8178           JSR [CHECK.RESULT]                      ;Check for error
U 07CC, 5B00,14     ;8179           RETURN                                  ;Loop on error
U 07CD, 3642,15     ;8180           MOV LS[BIT1] TO WR[0]                   ;Setup other data
U 07CE, BE88,15     ;8181           MOV WR[0] TO LS[ADDRESS.DATA]
U 07CF, E58A,15     ;8182           CLR LS[ERROR.MASK]                      ;Setup error mask
U 07D0, 36AA,15     ;8183           MOV LS[SEC.FLT] TO WR[0]                ;Setup received data
U 07D1, 36A4,95     ;8184           MOV LS[DATA.2] TO WR[1]                 ;Setup expected data
U 07D2, 0869,3C     ;8185           JSR [CHECK.RESULT]                      ;Check for error
U 07D3, 5B00,14     ;8186           RETURN                                  ;Loop on error
U 07D4, 36C0,15     ;8187           MOV LS[#3(H)] TO WR[0]                  ;Setup other data
U 07D5, BE88,15     ;8188           MOV WR[0] TO LS[ADDRESS.DATA]
U 07D6, 3E8B,95     ;8189           MOV WR[3] TO LS[ERROR.MASK]             ;Setup error mask
U 07D7, 36AC,15     ;8190           MOV LS[THIRD.FLT] TO WR[0]              ;Setup received data
U 07D8, B6A6,95     ;8191           MOV LS[DATA.3] TO WR[1]                 ;Setup expected data
U 07D9, 0869,3C     ;8192           JSR [CHECK.RESULT]                      ;Check for error
U 07DA, 5B00,14     ;8193           RETURN                                  ;Loop on error
U 07DB, DB00,16     ;8194           RETURN+1                                ;Return
                    ;8195
                    ;8196
                    ;8197   LOAD.SHIFT.EWR0:
U 07DC, B72E,15     ;8198           MOV LS[#2A7] TO WR[0]           ;ADDRESS OF DOUBLE LOAD ROUTINE
U 07DD, 90F6,15     ;8199           MISC2 [TRAP.ACC] WR[0]          ;FORCE ADDRESS OF DOUBLE LOAD
U 07DE, B6A2,15     ;8200           MOV LS[DATA.1] TO WR[0]         ;DATA FOR FIRST FLOAT
U 07DF, 10F4,15     ;8201           MISC2 [ASSERT.DA.AND.PASS.WR]   ;LOAD FIRST FLOAT INTO EWR[0] AND UPPER
                    ;8202                                           ; BITS OF FRAC PATH TO SHIFT INTO 16
                    ;8203                                           ; BITS OF EXPONENT
U 07E0, 361E,15     ;8204           MOV LS[T15] TO WR[0]            ;ADDRESS OF 'SHF LEFT FWR[FTO]'.  THIS
                    ;8205                                           ; SHIFT WILL GET RID OF HIDDEN BIT (BIT
                    ;8206                                           ; 55 OF FRAC PATH).
U 07E1, B716,95     ;8207           MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
```

```
U 07E2, 90F6,15 ;8208          MISC2 [TRAP.ACC] WR[0]          ;FORCE ADDRESS OF SHF LEFT FWR[FT0]
U 07E3, 10F6,95 ;8209          MISC2 [TRAP.ACC] WR[1]          ;FORCE ADDRESS OF LOOP
U 07E4, 0876,2C ;8210          JSR [SHIFT.LEFT.8]              ;SHIFT DATA INTO UPPER 8 BITS OF EXP.
                ;8211                                          ; FRAC 47-54 GO INTO LOWER 8 BITS OF EXP
U 07E5, 5B00,14 ;8212          RETURN                          ;DONE WITH LOAD, SHIFT OF EXPONENT
                ;8213
                ;8214  ;
                ;8215  ;ROUTINE TO SET UP DATA AT BEGINNING OF EACH TEST
                ;8216  ;
                ;8217
                ;8218  SETUP:
U 07E6, 65A0,15 ;8219          CLR LS[PREVIOUS.ERROR]          ;Clear previous error number in LS
U 07E7, E58A,15 ;8220          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 07E8, B646,15 ;8221          MOV LS[FPA] TO WR[0]            ;Store module code in LS to indicate
U 07E9, 3E8C,15 ;8222          MOV WR[0] TO LS[MODULE.NUM]     ; FPA board.
U 07EA, B640,15 ;8223          MOV LS[#1] TO WR[0]             ;1 in WR
U 07EB, BE82,15 ;8224          MOV WR[0] TO LS[ERROR.NUMBER]   ;Set up error number in LS.
U 07EC, 0874,3C ;8225          JSR [CLR.INST.AND.SIZE.BITS]    ;Clear instruction and size bits
U 07ED, 5B00,14 ;8226          RETURN                          ;DONE WITH SETUP
                ;8227
                ;8228  SETUP.10:
U 07EE, 65A0,15 ;8229          CLR LS[PREVIOUS.ERROR]          ;Clear previous error number in LS
U 07EF, B700,15 ;8230          MOV LS[FFFFFC00.MASK] TO WR[0]  ;Set up error mask to check lower 10
U 07F0, 3E8A,15 ;8231          MOV WR[0] TO LS[ERROR.MASK]     ; bits.
U 07F1, B646,15 ;8232          MOV LS[FPA] TO WR[0]            ;Store module code in LS to indicate
U 07F2, 3E8C,15 ;8233          MOV WR[0] TO LS[MODULE.NUM]     ; FPA board.
U 07F3, B640,15 ;8234          MOV LS[#1] TO WR[0]             ;1 in WR
U 07F4, BE82,15 ;8235          MOV WR[0] TO LS[ERROR.NUMBER]   ;Set up error number in LS.
U 07F5, 0874,3C ;8236          JSR [CLR.INST.AND.SIZE.BITS]    ;Clear instruction and size bits
U 07F6, 5B00,14 ;8237          RETURN                          ;DONE WITH SETUP
```

```
                ;8238   .page
                ;8239   .TOC    "TEST 1 VERIFICATION OF FPA(M8389 FPA MODULE OR CPU MODULE M8390)"
                ;8240   ;++
                ;8241   ;
                ;8242   ;
                ;8243   ; Test Description:
                ;8244   ;
                ;8245   ;       The purpose of this test is to verify that if the microdiagnostics are
                ;8246   ;       being run as a package whether there is an FPA or there isn't an FPA.
                ;8247   ;       This can be done by issuing a force to the FPA to execute an FPA
                ;8248   ;       instruction that will assert OPTION SYNC. The CPU is able to skip on
                ;8249   ;       OPTION SYNC. This means that the if OPTION SYNC is asserted then there
                ;8250   ;       will be a skip and if there is no FPA and OPTION SYNC is not asserted
                ;8251   ;       then there will be no skip and the FPA will be reported as not being
                ;8252   ;       there.
                ;8253   ;
                ;8254   ; Assumptions:
                ;8255   ;
                ;8256   ;       It is assumed that the there is a functioning CPU, WCS and MCT boards.
                ;8257   ;
                ;8258   ;       NOTE: The SER (single error report) flag must be clear unless looping
                ;8259   ;       on this test is desired (see debug)!
                ;8260   ;
                ;8261   ; Test Steps:
                ;8262   ;
                ;8263   ;       1) Issue a force to the FPA causing OPTION SYNC to be asserted. Then
                ;8264   ;          issue a nop in the CPU with the skip field set to 1F. If OPTION
                ;8265   ;          SYNC is asserted then issue a CPU ATTN with the FPA bit asserted.
                ;8266   ;          If OPTION SYNC is not asserted then issue a CPU ATTN with the FPA
                ;8267   ;          bit unasserted.
                ;8268   ;
                ;8269   ; Error:
                ;8270   ;
                ;8271   ;       None
                ;8272   ;
                ;8273   ; Debug:
                ;8274   ;
                ;8275   ;       If this test fails, that is, there is an FPA but OPTION SYNC is not
                ;8276   ;       asserted then it's likely that the logic causing OPTION SYNC to be
                ;8277   ;       asserted is in error. It's also possible that the CPU or the lines
                ;8278   ;       between the boards are the cause of error. If the FPA is the cause of
                ;8279   ;       error then the BRANCH 2 PAL should be checked for error. If the PAL is
                ;8280   ;       not the cause of error then the latches that the branch control bits
                ;8281   ;       come from should be checked for error. If the CPU is the cause of the
                ;8282   ;       error then the USEQ. PAL should be checked for error. If the PAL is not
                ;8283   ;       the cause of error then the OR gate below the PAL should be checked for
                ;8284   ;       error.
                ;8285   ;       If the FPA is present but not found by this test, the SYNC can be
                ;8286   ;       looped on by typing SE SER (set single error report), SE LO (set loop
                ;8287   ;       on error to prevent timeout), and starting this test over again.
                ;8288   ;
                ;8289   ;
                ;8290   ;--
                ;8291   T.1:
U 07F7, B65E,15 ;8292           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
```

```
                ;8293                                           ; STATUS WORD
U 07F8, 3E80,15 ;8294           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;8295                                           ; BIT 15 INDICATES START OF TEST TO
                ;8296                                           ; CONSOLE
U 07F9, 10E0,15 ;8297           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;8298   WAIT.T1.0:
U 07FA, 087F,A4 ;8299           JMP [WAIT.T1.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;8300                                           ; RESPOND
U 07FB, 087E,6C ;8301           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;8302                                           ; AND SIZE BITS
U 07FC, B646,15 ;8303           MOV LS[FPA] TO WR[0]            ;Store module code in LS to indicate
U 07FD, C542,15 ;8304           BIC LS[CPU] TO WR[0]            ; FPA and CPU board.
U 07FE, 3E8C,15 ;8305           MOV WR[0] TO LS[MODULE.NUM]
U 07FF, E50E,15 ;8306           CLR LS[FPA.FOUND]               ;Set up for message
U 0800, 6512,15 ;8307           CLR LS[T9]                      ;Sets up the location in physical
                ;8308                                           ; memory to be referenced.
U 0801, 8870,3C ;8309           JSR [LO]                        ;Load WR[0] with address to be
                ;8310                                           ; forced.
U 0802, BE14,15 ;8311           MOV WR[0] TO LS[T10]            ;Save no sync address
                ;8312   T1.LOOP:
U 0803, 3614,15 ;8313           MOV LS[T10] TO WR[0]            ;Restore no sync address
U 0804, 1080,15 ;8314           MISC [SET.ACC.I.0]              ;Set to ACC mode
U 0805, 90F6,15 ;8315           MISC2 [TRAP.ACC] WR[0]          ;Issue the force to force OPT SYNC
U 0806, 90F6,15 ;8316           MISC2 [TRAP.ACC] WR[0]          ; low
U 0807, DB00,1F ;8317           SKIP.IF[SYNC]                   ; and skip if it is asserted
U 0808, 8881,94 ;8318           JMP [NO.SYNC]                   ;If not asserted go to no sync.
U 0809, B690,95 ;8319           MOV LS[ERROR.CONTROL] TO  WR[1] ;Set up to loop if SER
                ;8320           BIT LS[SER] WITH WR[1],
U 080A, 5948,B5 ;8321            DT(LONG)&SET.ALU.CC
U 080B, 0880,31 ;8322           JMP.IF[BIT.SET] TO [T1.LOOP]
U 080C, B658,15 ;8323           MOV LS[PRINT.FPA] TO WR[0]      ;If sync then there is no FPA so
U 080D, 3E80,15 ;8324           MOV WR[0] TO LS[CONTROL.STATUS] ; set up control status word to print
U 080E, 10E0,15 ;8325           MISC [SET.CP.ATTN]              ; no FPA found.
U 080F, 0880,F4 ;8326   T1A:    JMP [T1A]                       ;Wait for console to return to me
U 0810, 3690,15 ;8327           MOV LS[ERROR.CONTROL] TO WR[0]  ;Setup phony data for APT
                ;8328           BIT LS[APT.PRESENT] WITH WR[0],
U 0811, 594A,35 ;8329            DT(LONG)&SET.ALU.CC
U 0812, 0AEE,19 ;8330           JMP.IF[BITS.CLR] TO [END.SECTION]
U 0813, 3640,95 ;8331           MOV LS[BIT0] TO WR[1]
U 0814, 366F,15 ;8332           MOV LS[NA.EXP.REC] TO WR[2]
U 0815, BE81,15 ;8333           MOV WR[2] TO LS[CONTROL.STATUS]
U 0816, 0869,3C ;8334           JSR [CHECK.RESULT]              ;Print and error for APT
U 0817, 0880,34 ;8335           JMP [T1.LOOP]                   ;Loop on error
U 0818, 8AEE,14 ;8336           JMP [END.SECTION]               ;Since there is no FPA jump to the end
U 0819, B690,95 ;8337   NO.SYNC:MOV LS[ERROR.CONTROL] TO  WR[1] ;Setup to loop if SER
                ;8338           BIT LS[SER] WITH WR[1],
U 081A, 5948,B5 ;8339            DT(LONG)&SET.ALU.CC
U 081B, 0880,31 ;8340           JMP.IF[NEQ] TO [T1.LOOP]
                ;8341   NO.APT:
U 081C, B658,15 ;8342           MOV LS[PRINT.FPA] TO WR[0]      ;IF there is an FPA then set up control
U 081D, 3E80,15 ;8343           MOV WR[0] TO LS[CONTROL.STATUS] ; status word to print FPA found.
U 081E, 7F0E,15 ;8344           INC LS[FPA.FOUND]
U 081F, 10E0,15 ;8345           MISC [SET.CP.ATTN]
U 0820, 8882,04 ;8346   T1B:    JMP [T1B]                       ;Wait for console to return control to
                ;8347                                           ;me
```

```
                ;8348   T1.END:
```

```
                ;8349   .page
                ;8350   .TOC    "TEST 2 VERIFICATION OF Y BUS TEST(M8383 FPA MODULE)"
                ;8351   ;++
                ;8352   ;
                ;8353   ;
                ;8354   ; Test Description:
                ;8355   ;
                ;8356   ;       This test will verify that the Y BUS as far as the CPU is concerned is
                ;8357   ;       working. This test is run in the event that only the FPA microdiagnos-
                ;8358   ;       tics are being run and the CPU has not been verified. The test will
                ;8359   ;       consist of moving data from local store to a WR and back to a temporary
                ;8360   ;       location in local store. The data that will be passed will be zeros,
                ;8361   ;       ones, and a floating pattern of ones through zeros.
                ;8362   ;
                ;8363   ; Assumptions:
                ;8364   ;
                ;8365   ;       It is assumed that main memory and the CPU have been tested and been
                ;8366   ;       found to be working.
                ;8367   ;
                ;8368   ; Test Steps:
                ;8369   ;
                ;8370   ;       1) MOV zeros from LS to WR[0]. MOV WR[0] back to a temporary LS
                ;8371   ;          location. If the temperary location doesn't contain zeros then
                ;8372   ;          issue error 1.
                ;8373   ;       2) Repeat step 1 for ones. If there is a failure then issue error 2.
                ;8374   ;       3) Repeat step 1 for floating ones through zeros. If there is a
                ;8375   ;          failure then issue error 3.
                ;8376   ;
                ;8377   ; Error:
                ;8378   ;
                ;8379   ;       Error1 - Y BUS failed with zeros on it.
                ;8380   ;       Error2 - Y BUS failed with ones on it.
                ;8381   ;       Error3 - Y BUS failed with floating ones on it.
                ;8382   ; Debug:
                ;8383   ;
                ;8384   ;       Error1: If this error occurs then it's likely that the cause of the
                ;8385   ;               error is with the Y BUS outside of the FPA. In this case the
                ;8386   ;               CPU module that the FPA is attached to should be checked for
                ;8387   ;               error.
                ;8388   ;       Error2: Same as error 1.
                ;8389   ;       Error3: Same as error 1.
                ;8390   ;
                ;8391   ;
                ;8392   ;--
                ;8393   T.2:
U 0821, B65E,15 ;8394           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;8395                                           ; STATUS WORD
U 0822, 3E80,15 ;8396           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;8397                                           ; BIT 15 INDICATES START OF TEST TO
                ;8398                                           ; CONSOLE
U 0823, 10E0,15 ;8399           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;8400   WAIT.T2.0:
U 0824, 0882,44 ;8401           JMP [WAIT.T2.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;8402                                           ; RESPOND
U 0825, 087E,6C ;8403           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
```

```
                    ;8404                                          ; AND SIZE BITS
U 0826, 369C,15 ;8405   T2.1:   MOV LS[#0] TO WR[0]                ;Load a WR with zero.
U 0827, BE14,15 ;8406           MOV WR[0] TO LS[T10]                ;Test the Y BUS by moving it
                ;8407                                              ; back to LS.
U 0828, 3614,15 ;8408           MOV LS[T10] TO WR[0]                ;Load up received data.
U 0829, 2F82,95 ;8409           CLR WR[1]                           ;Set up expected data.
U 082A, 0869,3C ;8410           JSR [CHECK.RESULT]                  ;Issue an error if there is one
U 082B, 8882,64 ;8411           JMP [T2.1]                          ;Loop on error.
U 082C, FF82,15 ;8412           INC LS[ERROR.NUMBER]                ;Add one to the error number.
U 082D, B69E,15 ;8413   T2.2:   MOV LS[#FFFFFFFF] TO WR[0]          ;Load a WR with ones.
U 082E, BE14,15 ;8414           MOV WR[0] TO LS[T10]                ;Test the Y BUS with ones.
U 082F, 3614,15 ;8415           MOV LS[T10] TO WR[0]                ;Set up for received data.
U 0830, 369E,95 ;8416           MOV LS[#FFFFFFFF] TO WR[1]          ;Set up for expected data.
U 0831, 0869,3C ;8417           JSR [CHECK.RESULT]                  ;If error issue one.
U 0832, 0882,D4 ;8418           JMP [T2.2]                          ;Loop on error.
U 0833, FF82,15 ;8419           INC LS[ERROR.NUMBER]                ;Add one to the error number.
U 0834, 2F85,15 ;8420           CLR WR[2]                           ;Put zeros in WR[2]
U 0835, 2045,15 ;8421           INC WR[2]                           ;Put a one in the LSB of WR[2]
U 0836, 3E15,15 ;8422   T2.A:   MOV WR[2] TO LS[T10]                ;Test Y BUS with pattern.
U 0837, 3614,15 ;8423           MOV LS[T10] TO WR[0]                ;Set up for received data
U 0838, A004,95 ;8424           MOV WR[2] TO WR[1]                  ;Set up expected data
U 0839, 0869,3C ;8425           JSR [CHECK.RESULT]                  ;If error issue one.
U 083A, 0883,64 ;8426           JMP [T2.A]                          ;Loop on error.
                ;8427           CMP WR[2] WITH LS[BIT31],           ;If last pattern
U 083B, CF7F,35 ;8428            DT(LONG)&SET.ALU.CC
U 083C, 8883,F9 ;8429           JMP.IF[EQL] TO [T2.END] ; then go to the next test
U 083D, A3D1,15 ;8430           ASHL WR[2]                          ; else shift left
U 083E, 0883,64 ;8431           JMP [T2.A]                          ; and jump to T2.A.
                ;8432   T2.END:
```

```
;8433   .page
;8434   .TOC    "TEST 3 NUA LINES TEST (M8389 FPA MODULE)"
;8435   ;++
;8436   ;
;8437   ;
;8438   ; Test Description:
;8439   ;
;8440   ;       This test checks the lower ten lines of the FPA BUS and the following
;8441   ;       data path. The Y BUS can be enabled onto the FPA BUS by issuing a
;8442   ;       MISC instruction with the PORT/MISC SELECT(CSR 18) bit clear and an
;8443   ;       eight in the MISC FUNCTION 1 field and a three in MISC FUNCTION 2
;8444   ;       field. This instruction also enables bits 0 through 9 of the FPA BUS
;8445   ;       onto the NUA BUS. The NUA BUS feeds into six 1K X 8 PROMS. The output
;8446   ;       of these PROMS are the 48 CS bits. The lower ten bits of the CS are
;8447   ;       the next address bits. These bits will be enabled onto the NUA BUS
;8448   ;       through the 2909's as long as the next address isn't being forced. The
;8449   ;       NUA BUS can be enabled onto the FPA BUS by issuing a MISC instruction
;8450   ;       with the PORT/MISC SELECT bit clear and a four in the in the MISC
;8451   ;       FUNCTION 2 field(read micro PC). A MOV instruction enables the FPA
;8452   ;       BUS onto the D BUS where it can be checked. The type of data that
;8453   ;       will be forced through this data path is ones rippled through zeros.
;8454   ;
;8455   ; Assumptions:
;8456   ;
;8457   ;       It's assumed that the tests for the WCS, CPU, and the MCT boards have
;8458   ;       been run successfully.
;8459   ;
;8460   ; Test Steps:
;8461   ;
;8462   ;       1) Load WR[0] with the next micro address to be forced in order to test
;8463   ;          the NUA BUS.
;8464   ;       2) Issue a MISC instruction with the PORT/MISC SELECT bit clear and an
;8465   ;          eight in the MISC FUNCTION 1 field. This causes the I/O mode to be
;8466   ;          set to FPA mode.
;8467   ;       3) Issue a MISC instruction with the PORT/MISC SELECT bit clear and a
;8468   ;          NOP in MISC FIELD 1 and a TRAP ACC (3) in MISC FIELD 2. This will
;8469   ;          force the address that's contained in WR[0].
;8470   ;       4) Issue a MISC instruction with the PORT/MISC SELECT bit clear and
;8471   ;          a NOP in MISC FIELD 1 and a READ ACC (4) in MISC FIELD 2. This
;8472   ;          will stop the clocks and enable the NUA onto the FPA BUS so the NUA
;8473   ;          can be read back to the CPU.
;8474   ;       5) Issue a MOV instruction to read the Y BUS back to the CPU. This will
;8475   ;          contain the NUA of the microword that was being executed at the time
;8476   ;          of the read instruction.
;8477   ;       6) Issue a MISC instruction with the PORT/MISC SELECT bit clear and a
;8478   ;          a NOP in MISC FIELD 1 and a TRAP ACC (3) in MISC FIELD 2. This will
;8479   ;          force an address in the FPA that jumps to itself. Repeat the test
;8480   ;          steps until all positions of the NUA have been tested.
;8481   ;
;8482   ; Error:
;8483   ;
;8484   ;       Error1 - One of the NUA lines failed or the control logic is in error.
;8485   ;       Error2 - Either the microsequencer or the PROM's failed.
;8486   ;
;8487   ; Debug:
```

```
                ;8488   ;
                ;8489   ;       Error1: There are two possibilities for error. If an error occurs on
                ;8490   ;               every pattern then it is likely that the control logic is in
                ;8491   ;               error. If the error occurs only on one pattern then it's likely
                ;8492   ;               that one of the NUA BUS lines are stuck high or low.
                ;8493   ;
                ;8494   ;               Control Logic Error
                ;8495   ;
                ;8496   ;               If the control logic is suspected of being in error then the
                ;8497   ;               DIR and the ENABLE to the transceiver should be checked for
                ;8498   ;               error. The enable should be asserted. It is not then FPAL ALLOW
                ;8499   ;               CPU Y BUS H and DAPK TRAP ACC L should be checked to be
                ;8500   ;               asserted and FPAC CPU PHO, FPAB ENB CP LOAD, and FPAB RCV DATA
                ;8501   ;               L should be checked to be unasserted.  The DIR of the tran-
                ;8502   ;               sceiver should be unasserted(high) to enable the Y BUS onto the
                ;8503   ;               FPA BUS and asserted(low) to go the other way. If data is being
                ;8504   ;               passed to the FPA then DAPA READ PORT L and DAPK SEL ACC IN H
                ;8505   ;               should have at least one or both asserted. If data is being
                ;8506   ;               passed from the FPA to the CPU then both DAPA READ PORT L and
                ;8507   ;               DAPK SEL ACC IN H should be unasserted. The other logic which
                ;8508   ;               could cause a complete logic of all NUA lines is the force
                ;8509   ;               logic. This can be checked on the NUA transceivers DIR input.
                ;8510   ;               If FPAA FORCE UADDR (1) H is not asserted during the MISC
                ;8511   ;               [TRAP.ACC] instruction then the associated logic should be
                ;8512   ;               checked for error.
                ;8513   ;
                ;8514   ;               Single NUA Line Failures
                ;8515   ;
                ;8516   ;               If only on NUA line is failing then the lines nearest to it
                ;8517   ;               should be checked for shorts.
                ;8518   ;       Error2: If error 1 occurs then error 2 will also occur and the
                ;8519   ;               problem is likely to be found in the debug for error1. If
                ;8520   ;               only error 2 occurs then it's likely that either the
                ;8521   ;               Cntrol Store PROM's or the microsequencer are the cause of
                ;8522   ;               the error.
                ;8523   ;
                ;8524   ;
                ;8525   ;--
                ;8526   T.3:
U 083F, B65E,15 ;8527           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;8528                                           ; STATUS WORD
U 0840, 3E80,15 ;8529           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;8530                                           ; BIT 15 INDICATES START OF TEST TO
                ;8531                                           ; CONSOLE
U 0841, 10E0,15 ;8532           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;8533   WAIT.T3.0:
U 0842, 0884,24 ;8534           JMP [WAIT.T3.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;8535                                           ; RESPOND
U 0843, 887E,EC ;8536           JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;8537                                           ; AND SIZE BITS. CHECK LOWER 10 BITS
U 0844, 3641,15 ;8538           MOV LS[#1] TO WR[2]             ;Initialize NUA data.
U 0845, 6512,15 ;8539           CLR LS[T9]                      ;Initialize memory location.
U 0846, FF12,15 ;8540           INC LS[T9]
U 0847, FF12,15 ;8541           INC LS[T9]
U 0848, 0870,8C ;8542   T3.A:   JSR [L1]                        ;Get expected data from main
```

```
                ;8543                                   ; memory.
U 0849, 3E14,95 ;8544           MOV WR[1] TO LS[T10]
U 084A, 2004,15 ;8545   T3.1:   MOV WR[2] TO WR[0]      ;Load Uaddress to be forced.
U 084B, 1080,15 ;8546           MISC [SET.ACC.I.0]      ;Set to ACC mode.
U 084C, 90F6,15 ;8547           MISC2 [TRAP.ACC] WR[0]  ;Force the Uaddress in WR[0].
U 084D, 10F8,15 ;8548           MISC2 [READ.ACC.UPC]    ;Stop the FPA clocks and load
                ;8549                                   ; the NUA on the FPA BUS.
U 084E, BA16,15 ;8550           MOV FPA TO LS[T11]      ;Enable the FPA BUS onto the
                ;8551                                   ; Y BUS and into LS.
U 084F, 3716,15 ;8552           MOV LS[#300] TO WR[0]   ;Force the FPA to loop on self
U 0850, 90F6,15 ;8553           MISC2 [TRAP.ACC] WR[0]
U 0851, B616,15 ;8554           MOV LS[T11] TO WR[0]    ;Set up for received data.
U 0852, B614,95 ;8555           MOV LS[T10] TO WR[1]    ;Set up expected data
U 0853, 0869,3C ;8556           JSR [CHECK.RESULT]      ;Report if there's an error.
U 0854, 8884,A4 ;8557           JMP [T3.1]              ;Loop on error
U 0855, A3D1,15 ;8558           ASHL WR[2]              ;Create next uaddress to be
                ;8559                                   ; forced.
                ;8560           CMP WR[2] WITH LS[#10],  ;If the address is 10 then
U 0856, CF49,35 ;8561            DT(LONG)&SET.ALU.CC
U 0857, DB00,01 ;8562           SKIP.IF[NEQ]            ;Skip to next address
U 0858, A3D1,15 ;8563           ASHL WR[2]
                ;8564           CMP WR[2] WITH LS[#4],   ;If the address is 4 then skip
U 0859, CF45,35 ;8565            DT(LONG)&SET.ALU.CC    ; to next address.
U 085A, DB00,01 ;8566           SKIP.IF[NEQ]
U 085B, A3D1,15 ;8567           ASHL WR[2]              ;Go on to next pattern.
                ;8568           CMP WR[2] WITH LS[#200], ;Have all nine bits been
U 085C, 4F53,35 ;8569            DT(LONG)&SET.ALU.CC    ; tested?
U 085D, 0884,81 ;8570           JMP.IF[NEQ] TO [T3.A]   ;If not then loop
U 085E, 0870,8C ;8571           JSR [L1]                ;Get expected data
U 085F, 3E14,95 ;8572           MOV WR[1] TO LS[T10]
U 0860, B64A,15 ;8573   T3.1A:  MOV LS[#20] TO WR[0]     ;Setup 24(H) for forcing
U 0861, 4744,15 ;8574           BIS LS[#4] TO WR[0]
U 0862, 1080,15 ;8575           MISC [SET.ACC.I.0]      ;Set to ACC mode
U 0863, 90F6,15 ;8576           MISC2 [TRAP.ACC] WR[0]  ;Force Uaddress in WR[0].
U 0864, 10F8,15 ;8577           MISC2 [READ.ACC.UPC]    ;Stop the FPA clocks and load
                ;8578                                   ; the NUA on the FPA BUS.
U 0865, BA16,15 ;8579           MOV FPA TO LS[T11]      ;Enable the FPA BUS onto the
                ;8580                                   ; Y BUS and into LS.
U 0866, 3716,15 ;8581           MOV LS[#300] TO WR[0]   ;Force the FPA to loop on self
U 0867, 90F6,15 ;8582           MISC2 [TRAP.ACC] WR[0]
U 0868, B616,15 ;8583           MOV LS[T11] TO WR[0]    ;Set up for received data.
U 0869, B614,95 ;8584           MOV LS[T10] TO WR[1]    ;Set up expected data
U 086A, 0869,3C ;8585           JSR [CHECK.RESULT]      ;Report if there's an error.
U 086B, 0886,04 ;8586           JMP [T3.1A]             ;Loop on error.
U 086C, 0870,8C ;8587           JSR [L1]
U 086D, 3E14,95 ;8588           MOV WR[1] TO LS[T10]    ;Setup expected data
U 086E, 3648,15 ;8589   T3.1B:  MOV LS[#10] TO WR[0]     ;Setup 24(H) for forcing
U 086F, C740,15 ;8590           BIS LS[#1] TO WR[0]
U 0870, 1080,15 ;8591           MISC [SET.ACC.I.0]      ;Set to ACC mode
U 0871, 90F6,15 ;8592           MISC2 [TRAP.ACC] WR[0]  ;Force Uaddress in WR[0].
U 0872, 10F8,15 ;8593           MISC2 [READ.ACC.UPC]    ;Stop the FPA clocks and load
                ;8594                                   ; the NUA on the FPA BUS.
U 0873, BA16,15 ;8595           MOV FPA TO LS[T11]      ;Enable the FPA BUS onto the
                ;8596                                   ; Y BUS and into LS.
U 0874, 3716,15 ;8597           MOV LS[#300] TO WR[0]   ;Force the FPA to loop on self
```

```
U 0875, 90F6,15 ;8598           MISC2 [TRAP.ACC] WR[0]
U 0876, B616,15 ;8599           MOV LS[T11] TO WR[0]            ;Set up for received data.
U 0877, B614,95 ;8600           MOV LS[T10] TO WR[1]            ;Set up expected data
U 0878, 0869,3C ;8601           JSR [CHECK.RESULT]              ;Report if there's an error.
U 0879, 8886,E4 ;8602           JMP [T3.1B]                     ;Loop on error.
U 087A, FF82,15 ;8603           INC LS[ERROR.NUMBER]            ;Error 2
U 087B, 8870,3C ;8604           JSR [L0]                        ;Get address of Last address forced
U 087C, 2001,15 ;8605           MOV WR[0] TO WR[2]
U 087D, 8870,3C ;8606   T3.2A:  JSR [L0]                        ;Get next Address to be forced
U 087E, BE14,15 ;8607           MOV WR[0] TO LS[T10]            ;Save next address to be forced
                ;8608                                           ; in LS temp 10.
U 087F, 0870,8C ;8609           JSR [L1]                        ;Get expected data and load it
                ;8610                                           ; into WR[1].
U 0880, 3E18,95 ;8611           MOV WR[1] TO LS[T12]            ;Save expected data
U 0881, 3614,15 ;8612   T3.2:   MOV LS[T10] TO WR[0]            ;Replace current address to be
                ;8613                                           ; forced for loop on error
U 0882, 90F6,15 ;8614           MISC2 [TRAP.ACC] WR[0]          ;Force the address in WR[0]
U 0883, 10F8,15 ;8615           MISC2 [READ.ACC.UPC]            ;Enable the Uaddress on the NUA
                ;8616                                           ; BUS to the FPA BUS.
U 0884, BA16,15 ;8617           MOV FPA TO LS[T11]              ;Load the FPA BUS into LS temp
                ;8618                                           ; 11.
U 0885, 3716,15 ;8619           MOV LS[#300] TO WR[0]           ;Force the FPA to loop on self
U 0886, 90F6,15 ;8620           MISC2 [TRAP.ACC] WR[0]
U 0887, B616,15 ;8621           MOV LS[T11] TO WR[0]            ;Load received data into WR[0]
U 0888, B618,95 ;8622           MOV LS[T12] TO WR[1]            ;Set up expected data
U 0889, 0869,3C ;8623           JSR [CHECK.RESULT]              ;Check for error and report.
U 088A, 0888,14 ;8624           JMP [T3.2]                      ;Loop on error.
                ;8625           CMP WR[2] WITH LS[T10],         ;If the last address has not
U 088B, CF15,35 ;8626            DT(LONG)&SET.ALU.CC            ; been tested
U 088C, 0887,D1 ;8627           JMP.IF[NEQ] TO [T3.2A]          ; then continue testing
                ;8628   T3.END:
```

```
;8629    .page
;8630    .TOC    "TEST 4 PARITY LOGIC TEST (M8389 FPA MODULE)"
;8631    ;++
;8632    ;
;8633    ;
;8634    ; Test Description:
;8635    ;
;8636    ;       This test checks the parity logic. The parity is checked by two
;8637    ;       parity bits which are bits 45 through 46 of the control store bits.
;8638    ;       The Control Store will contain a number of words with bad parity
;8639    ;       that will be used to test the parity logic. Parity bit 0 (CS45) will
;8640    ;       be set if there are an odd number of ones in bits 0 - 8, 15 - 21, 37,
;8641    ;       38, and 47 of the microword. However parity bit 0 is determined by 3
;8642    ;       lines coming into the PARITY CTL PAL so it will have to be tested for
;8643    ;       all 3 cases. Parity bit 1 (CS46) will be set if there are an odd
;8644    ;       number of ones in bits 9 - 14, 22 - 36, and 39 - 44. Parity bit 0 is
;8645    ;       also a result of three inputs into the PARITY CTL PAL and will also
;8646    ;       have to be tested for all three of these cases. The words with bad
;8647    ;       parity will be stored in the Control Store and can be accessed by
;8648    ;       issuing a MISC instruction with the address of the word with bad parity
;8649    ;       on the Y BUS. If the parity of parity bit 1 is wrong then FPA D 02 is
;8650    ;       low. If the parity of the parity bit 0 is wrong then FPA D 01 is low.
;8651    ;       Since the bad parity of each parity bit can be determined regardless of
;8652    ;       the parity of the other parity bits then the two parity bits can be
;8653    ;       tested simultaneously. A MOV instruction will load the contents of the
;8654    ;       FPA BUS onto the Y BUS and into the location specified by the MOV
;8655    ;       instruction. Bad parity should also cause the micro sequencer to force
;8656    ;       the next micro address to zero. This can be checked by issuing a MISC
;8657    ;       instruction to access the micro word with bad parity. The next micro
;8658    ;       address can be determined by issuing the MISC instruction to force the
;8659    ;       micro word with bad parity followed by a MISC instruction with a four
;8660    ;       in the MISC FUNCTION 2 field and the PORT/MISC SELECT bit set. This
;8661    ;       enables the NUA BUS onto the FPA BUS. The FPA BUS can then be enabled
;8662    ;       onto the Y BUS by issuing a MOV instruction with a five in the DATA
;8663    ;       PATH CONTROL FIELD. Once in Local Store the data will be checked by
;8664    ;       check result.
;8665    ;
;8666    ; Assumptions:
;8667    ;
;8668    ;       It's assumed that the NUA BUS has been checked. The enables and dir-
;8669    ;       rections of the transceivers from the Y BUS to the FPA BUS and the FPA
;8670    ;       BUS to the NUA BUS have been checked and are working.
;8671    ;
;8672    ; Test Steps:
;8673    ;
;8674    ;       1) Issue a MISC instruction to access a micro word with an even number
;8675    ;          of bits set in each group of bits in each parity bit and set all
;8676    ;          the parity bits.
;8677    ;       2) Issue a MOV instruction to load the FPA BUS onto the Y BUS and then
;8678    ;          load the data into WR[0] and the expected data into WR[1] and call
;8679    ;          Check Result.
;8680    ;       3) Issue a MISC instruction with an odd number of bits set and the
;8681    ;          two parity bits clear.
;8682    ;       4) Issue a MOV instruction to load them into Local Store and load the
;8683    ;          data into WR[0] and the expected data into WR[1] and call Check
```

```
                ;8684   ;           Result.
                ;8685   ;       5) Repeat steps 1) through 4) using different micro words with bad
                ;8686   ;          parity in order to test the other inputs to the PARITY CTL PAL.
                ;8687   ;       6) Issue a MISC instruction to access a micro word with bad parity.
                ;8688   ;       7) Issue a MISC instruction to enable the NUA BUS onto the FPA BUS.
                ;8689   ;          The micro word with bad parity should have caused the micro
                ;8690   ;          sequencer to force zero onto the NUA BUS.
                ;8691   ;       8) Issue a MOV instruction to enable the FPA BUS onto the Y BUS. Bits
                ;8692   ;          0 through 9 should be be zero. Load the data into WR[0] and zeros
                ;8693   ;          into WR[1] and call Check Result.
                ;8694   ;
                ;8695   ; Error:
                ;8696   ;
                ;8697   ;       Error1 - One or more of the parity bits failed.
                ;8698   ;       Error2 - Either the parity bits failed or the FORCE LOW UADDR failed.
                ;8699   ;
                ;8700   ; Debug:
                ;8701   ;
                ;8702   ;       Error1: By examining the expected and received data the parity bit
                ;8703   ;               that failed can be determined. If bit one is low then parity
                ;8704   ;               bit zero failed. If bit 2 is low then parity bit 1 failed.
                ;8705   ;               If either of the two bits failed then the PARITY CTL PAL should
                ;8706   ;               be checked for error. If parity bit zero is low then then a
                ;8707   ;               parity error was not detected in UPF CK, ACC SYNC, or ROM CHK.
                ;8708   ;               It's also possible that the parity bit itself could be in error
                ;8709   ;               and the latch that it comes from should be checked in this case
                ;8710   ;               If parity bit one is low then a parity error was not detected
                ;8711   ;               in PAR CHK1, BCTL CHK, PAR2 CHK, or the parity bit itself, PAR1
                ;8712   ;               If either PAR1 CHK, or PAR0 CHK are in error then the respec-
                ;8713   ;               tive parity checkers and the associated latches should be
                ;8714   ;               checked for error. If BCTL CHK is in error then the INSTRUCTION
                ;8715   ;               PAL should be checked for error. If the cause of error is not
                ;8716   ;               the PAL then the latches before the INSTRUCTION PAL should be
                ;8717   ;               checked for error. If the parity bit is in error then the latch
                ;8718   ;               it came from should be checked for error.
                ;8719   ;       Error2: This error will occur in one of two cases if the parity logic
                ;8720   ;               failed or the PARITY CTL PAL failed to output the correct
                ;8721   ;               signal to enable the force zero onto the micro sequencer. The
                ;8722   ;               parity logic would have had to fail on all two parity bits
                ;8723   ;               for there not to be an error so more than likely the PARITY
                ;8724   ;               CTL PAL is the source of error. If the signal FPAD FORCE LOW
                ;8725   ;               UADDR L is asserted then the only other source of error is the
                ;8726   ;               micro sequencer which didn't force zeros onto the NUA BUS.
                ;8727   ;
                ;8728   ;
                ;8729   ;--
                ;8730   T.4:
U 088D, B65E,15 ;8731           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;8732                                           ; STATUS WORD
U 088E, 3E80,15 ;8733           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;8734                                           ; BIT 15 INDICATES START OF TEST TO
                ;8735                                           ; CONSOLE
U 088F, 10E0,15 ;8736           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;8737   WAIT.T4.0:
U 0890, 0889,04 ;8738           JMP [WAIT.T4.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
```

```
                ;8739                                                   ; RESPOND
U 0891, 087E,6C ;8740           JSR [SETUP]                             ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;8741                                                   ; AND SIZE BITS
U 0892, B706,15 ;8742           MOV LS[FFFFFFF8.MASK] TO WR[0]          ;Set up error mask to check lower 10
U 0893, 3E8A,15 ;8743           MOV WR[0] TO LS[ERROR.MASK]             ; bits.
U 0894, 3702,15 ;8744           MOV LS[T.4POINTER] TO WR[0]             ;Initialize pointer for main memory
                ;8745                                                   ; references.
U 0895, BE12,15 ;8746           MOV WR[0] TO LS[T9]
U 0896, B705,15 ;8747           MOV LS[LST.BPW] TO WR[2]                ;Load the address last bad parity word
                ;8748                                                   ; into WR[2] for end of test checking
U 0897, 0870,8C ;8749   T.4A:   JSR [L1]                                ;Get the expected data.
U 0898, BE16,95 ;8750           MOV WR[1] TO LS[T11]                    ;Save expected data
U 0899, 8871,2C ;8751           JSR [T84]                               ;Get the address to be forced
U 089A, 3708,15 ;8752   T4.1:   MOV LS[T84] TO WR[0]                    ;Put address to be forced in wr 0.
U 089B, 90F6,15 ;8753           MISC2 [TRAP.ACC] WR[0]                  ;Force the word with bad parity
U 089C, DB00,15 ;8754           NOP                                     ;Wait for force
U 089D, DB00,15 ;8755           NOP                                     ;Wait for jump to loc. 0
U 089E, 3A14,15 ;8756           MOV FPA TO LS[T10]                      ;Read the data from the FPA to
                ;8757                                                   ; LS temp 10.
U 089F, 3614,15 ;8758           MOV LS[T10] TO WR[0]                    ; Load receicved data into WR0
U 08A0, 3616,95 ;8759           MOV LS[T11] TO WR[1]                    ;Setup expected data
U 08A1, 0869,3C ;8760           JSR [CHECK.RESULT]                      ;If there's an error report it
U 08A2, 0889,A4 ;8761           JMP [T4.1]                              ;Loop on error.
                ;8762           CMP WR[2] WITH LS[T9],                  ;If not last bad parity word
U 08A3, CF13,35 ;8763            DT(LONG)&SET.ALU.CC                    ; (BPW)
U 08A4, 8889,71 ;8764           JMP.IF[NEQ] TO [T.4A]                   ; then continue testing BPW's
U 08A5, FF82,15 ;8765           INC LS[ERROR.NUMBER]                    ;Increase error number
U 08A6, 3708,15 ;8766   T4.2:   MOV LS[T84] TO WR[0]                    ;Put the address to be forced in wr0
U 08A7, 90F6,15 ;8767           MISC2 [TRAP.ACC] WR[0]                  ;Force the Uaddress.
U 08A8, 10F8,15 ;8768           MISC2 [READ.ACC.UPC]                    ;Enable the next Uaddress bits
                ;8769                                                   ; onto the FPA BUS.
U 08A9, 3A14,15 ;8770           MOV FPA TO LS[T10]                      ;Enable the FPA BUS into LS
                ;8771                                                   ; temp 10.
U 08AA, B700,15 ;8772           MOV LS[FFFFFC00.MASK] TO WR[0]          ;Get 3FF mask.
U 08AB, 3E8A,15 ;8773           MOV WR[0] TO LS[ERROR.MASK]             ;And put it in error mask
U 08AC, 3614,15 ;8774           MOV LS[T10] TO WR[0]                    ;Set up received data.
U 08AD, 2F82,95 ;8775           CLR WR[1]                               ;Set up expected data.
U 08AE, 0869,3C ;8776           JSR [CHECK.RESULT]                      ;If there's an error report it.
U 08AF, 088A,64 ;8777           JMP [T4.2]
                ;8778   T4.END:
```

```
;8779    .page
;8780    .TOC    "TEST 5 CONTROL STORE TEST (M8389 FPA MODULE)"
;8781    ;++
;8782    ;
;8783    ;
;8784    ; Test Description:
;8785    ;
;8786    ;       The purpose of this test is to check the micro words in the CONTROL
;8787    ;       STORE for parity errors. This can be done by issuing a force micro
;8788    ;       address through the FPA BUS to the NUA BUS to the micro sequencer to
;8789    ;       force the address of the word in CONTROL STORE who's parity is to be
;8790    ;       checked. In order to see if the parity is OK the next micro address
;8791    ;       can be read from the NUA BUS to the FPA BUS and onto the Y BUS where
;8792    ;       it is checked. If there's a parity error then the next micro address
;8793    ;       is forced to zero. In order to prevent the micro code from continuing
;8794    ;       to execute if there's not a parity error the instruction after the read
;8795    ;       from the NUA BUS should be a force instruction to a location in CON-
;8796    ;       TROL STORE that executes no instruction and the next micro address is
;8797    ;       the address pointed to now. In this manner all of the locations in
;8798    ;       CONTROL STORE can be tested for parity errors. The words with bad
;8799    ;       parity used in the previous test will be skipped.
;8800    ;
;8801    ; Assumtions:
;8802    ;
;8803    ;       It is assumed that the parity logic has been checked and is working
;8804    ;       and the lines on the NUA BUS and the FPA BUS have been checked and
;8805    ;       work.
;8806    ;
;8807    ; Test Steps:
;8808    ;
;8809    ;       1) Clear a counter to zero.
;8810    ;       2) Issue a MISC instruction to force the next micro instruction with
;8811    ;          the next micro address being the contents of the counter.
;8812    ;       3) Issue a MOV instruction to read the next micro address from the NUA
;8813    ;          BUS to the FPA BUS to the Y BUS and into a local store location.
;8814    ;       4) Issue a MISC instruction to force the next micro instruction to a
;8815    ;          location in local store that's a NOP with the next micro address
;8816    ;          being the address that's pointed to.
;8817    ;       5) Compare the LS location to one. If LS location is zero then load
;8818    ;          zero into WR[0] and the counter into WR[1] and call Check Result.
;8819    ;       6) Else check counter to see if it's greater than 995, if not then
;8820    ;          increment counter and jump to step 2.
;8821    ;
;8822    ; Error:
;8823    ;
;8824    ;       NOTE: Other data in this case refers to the micro-address being checked
;8825    ;             for parity.
;8826    ;
;8827    ;       Error1 - A location in CONTROL STORE is in error.
;8828    ;
;8829    ; Debug:
;8830    ;
;8831    ;       Error1: A location in the FPA CONTROL STORE has a parity error. The
;8832    ;               location in the CONTROL STORE that's in error is in WR[1]
;8833    ;               which is printed as expected data.
```

```
                ;8834   ;
                ;8835   ;
                ;8836   ;--
                ;8837   T.5:
U 08B0, B65E,15 ;8838           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;8839                                           ; STATUS WORD
U 08B1, 3E80,15 ;8840           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;8841                                           ; BIT 15 INDICATES START OF TEST TO
                ;8842                                           ; CONSOLE
U 08B2, 10E0,15 ;8843           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;8844   WAIT.T5.0:
U 08B3, 888B,34 ;8845           JMP [WAIT.T5.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;8846                                           ; RESPOND
U 08B4, 887E,EC ;8847           JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;8848                                           ; AND SIZE BITS. CHECK LOWER 10 BITS
U 08B5, B79A,15 ;8849           MOV LS[T5.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 08B6, BE12,15 ;8850           MOV WR[0] TO LS[T9]
U 08B7, B670,15 ;8851           MOV LS[OTHER.DATA] TO WR[0]     ;Set up other data fiald
U 08B8, 3E80,15 ;8852           MOV WR[0] TO LS[CONTROL.STATUS]
U 08B9, B716,95 ;8853           MOV LS[#300] TO WR[1]           ;Setup to loop self
U 08BA, 10F6,95 ;8854           MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 08BB, 2F85,15 ;8855           CLR WR[2]                       ;Initialize pointer
U 08BC, 2045,15 ;8856   T5.1:   INC WR[2]                       ;Check location 1 for a parity error
U 08BD, B652,15 ;8857           MOV LS[#200] TO WR[0]           ;Check for locations with known bad
U 08BE, 4742,15 ;8858           BIS LS[#2] TO WR[0]
U 08BF, 474E,15 ;8859           BIS LS[#80] TO WR[0]
U 08C0, C74A,15 ;8860           BIS LS[#20] TO WR[0]
U 08C1, 4748,15 ;8861           BIS LS[#10] TO WR[0]
                ;8862           CMP WR[0] WITH WR[2],           ;Check for location 2B2
U 08C2, 2641,35 ;8863            DT(LONG)&SET.ALU.CC
U 08C3, 088C,61 ;8864           JMP.IF[NEQ] TO [T5.2B4]         ;If not 2B2 the jump to T5.2B4
U 08C4, 2045,15 ;8865           INC WR[2]                       ; else go onto next location
U 08C5, 888F,24 ;8866           JMP [T5.LOOP]
U 08C6, 2040,15 ;8867   T5.2B4: INC WR[0]
U 08C7, 2040,15 ;8868           INC WR[0]
                ;8869           CMP WR[0] WITH WR[2],           ;Check for location 2B4
U 08C8, 2641,35 ;8870            DT(LONG)&SET.ALU.CC
U 08C9, 088C,C1 ;8871           JMP.IF[NEQ] TO [T5.2B6]         ;If not 2B4 then check for next bpw
U 08CA, 2045,15 ;8872           INC WR[2]
U 08CB, 888F,24 ;8873           JMP [T5.LOOP]
U 08CC, 2040,15 ;8874   T5.2B6: INC WR[0]
U 08CD, 2040,15 ;8875           INC WR[0]
                ;8876           CMP WR[0] WITH WR[2],           ;Check for location 2B6
U 08CE, 2641,35 ;8877            DT(LONG)&SET.ALU.CC
U 08CF, 088D,21 ;8878           JMP.IF[NEQ] TO [T5.2B8]         ;If not 2B8 then check for next bpw
U 08D0, 2045,15 ;8879           INC WR[2]
U 08D1, 888F,24 ;8880           JMP [T5.LOOP]
U 08D2, 2040,15 ;8881   T5.2B8: INC WR[0]
U 08D3, 2040,15 ;8882           INC WR[0]
                ;8883           CMP WR[0] WITH WR[2],           ;Check location 2B8
U 08D4, 2641,35 ;8884            DT(LONG)&SET.ALU.CC
U 08D5, 088D,81 ;8885           JMP.IF[NEQ] TO [T5.2BC]         ;If not 2B8 then check for mext bpw
U 08D6, 2045,15 ;8886           INC WR[2]
U 08D7, 888F,24 ;8887           JMP [T5.LOOP]
U 08D8, 4744,15 ;8888   T5.2BC: BIS LS[#4] TO WR[0]
```

```
                ;8889           CMP WR[0] WITH WR[2],
U 08D9, 2641,35 ;8890            DT(LONG)&SET.ALU.CC
U 08DA, 088D,E1 ;8891           JMP.IF[NEQ] TO [T5.2C0]        ;If not 2BC then skip 2BD and go to
U 08DB, 2045,15 ;8892           INC WR[2]                      ; next bad parity word
U 08DC, 2045,15 ;8893           INC WR[2]
U 08DD, 888F,24 ;8894           JMP [T5.LOOP]
U 08DE, 2040,15 ;8895   T5.2C0: INC WR[0]
U 08DF, 2040,15 ;8896           INC WR[0]
U 08E0, 2040,15 ;8897           INC WR[0]
U 08E1, 2040,15 ;8898           INC WR[0]
                ;8899           CMP WR[0] WITH WR[2],
U 08E2, 2641,35 ;8900            DT(LONG)&SET.ALU.CC
U 08E3, 888E,61 ;8901           JMP.IF[NEQ] TO [T5.2C4]        ;If not 2C0 then check for next bpw
U 08E4, 2045,15 ;8902           INC WR[2]
U 08E5, 888F,24 ;8903           JMP [T5.LOOP]
U 08E6, 4744,15 ;8904   T5.2C4: BIS LS[#4] TO WR[0]
                ;8905           CMP WR[0] WITH WR[2],
U 08E7, 2641,35 ;8906            DT(LONG)&SET.ALU.CC
U 08E8, 888E,C1 ;8907           JMP.IF[NEQ] TO [T5.2C9]        ;If not 2C4 then skip next 2 locations
U 08E9, 2045,15 ;8908           INC WR[2]                      ; and check next bpw
U 08EA, 2045,15 ;8909           INC WR[2]
U 08EB, 888F,24 ;8910           JMP [T5.LOOP]
U 08EC, C746,15 ;8911   T5.2C9: BIS LS[#8] TO WR[0]
U 08ED, C544,15 ;8912           BIC LS[#4] TO WR[0]
U 08EE, 2040,15 ;8913           INC WR[0]
                ;8914           CMP WR[0] WITH WR[2],
U 08EF, 2641,35 ;8915            DT(LONG)&SET.ALU.CC
U 08F0, 888F,21 ;8916           JMP.IF[NEQ] TO [T5.LOOP]       ;If not 2C9 then check then location
U 08F1, 2045,15 ;8917           INC WR[2]                      ; for bad parity
U 08F2, B652,15 ;8918   T5.LOOP:MOV LS[#200] TO WR[0]          ;Set up to force call incase of a return
U 08F3, 2100,15 ;8919           DEC WR[0]
U 08F4, B716,95 ;8920           MOV LS[#300] TO WR[1]          ;Setup to loop self
U 08F5, 90F6,15 ;8921           MISC2 [TRAP.ACC] WR[0]         ;Force call
U 08F6, 10F6,95 ;8922           MISC2 [TRAP.ACC] WR[1]         ;Force loop
U 08F7, 2004,15 ;8923           MOV WR[2] TO WR[0]             ;Force the location in WR[2]
U 08F8, 90F6,15 ;8924           MISC2 [TRAP.ACC] WR[0]
U 08F9, 10F8,15 ;8925           MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 08FA, 3A14,15 ;8926           MOV FPA TO LS[T10]             ;Enable the FPA BUS onto the Y BUS
U 08FB, DB00,15 ;8927           NOP
U 08FC, BA16,15 ;8928           MOV FPA TO LS[T11]
U 08FD, 10F6,95 ;8929           MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 08FE, 3614,15 ;8930           MOV LS[T10] TO WR[0]           ;Check to see that the NUA is not 0
U 08FF, 458A,15 ;8931           BIC LS[ERROR.MASK] TO WR[0]    ; after masking out all but the NUA
                ;8932           CMP WR[0] WITH LS[#0],
U 0900, 4F9C,35 ;8933            DT(LONG)&SET.ALU.CC
U 0901, 0890,59 ;8934           JMP.IF[EQL] TO [T5.ERROR]
U 0902, 2F80,15 ;8935           CLR WR[0]
U 0903, 2F82,95 ;8936           CLR WR[1]
U 0904, 0890,74 ;8937           JMP [CHECK.FOR.ERROR]
                ;8938   T5.ERROR:
U 0905, B616,15 ;8939           MOV LS[T11] TO WR[0]           ;Set up received data with the parity
U 0906, 2F82,95 ;8940           CLR WR[1]                      ;Set expected data
                ;8941   CHECK.FOR.ERROR:
U 0907, 3E89,15 ;8942           MOV WR[2] TO LS[ADDRESS.DATA]  ;Set up other data
U 0908, 0869,3C ;8943           JSR [CHECK.RESULT]             ;Check for error
```

```
U 0909, 888F,24 ;8944          JMP [T5.LOOP]                    ;Loop on error
U 090A, B654,15 ;8945          MOV LS[#400] TO WR[0]            ;Create the # 3FF
U 090B, 2100,15 ;8946          DEC WR[0]
                ;8947          CMP WR[2] WITH WR[0],            ;Have all FPA control store locations
U 090C, 2644,35 ;8948           DT(LONG)&SET.ALU.CC             ;Been checked?
U 090D, 888B,C1 ;8949          JMP.IF[NEQ] TO [T5.1]            ;If not get next location
                ;8950   T5.END:
```

```
                ;8951   .page
                ;8952   .TOC    "TEST 6 CLOCK SYNC TEST(M8389 FPA OR M8390 CPU MODULE)"
                ;8953   ;++
                ;8954   ;
                ;8955   ;
                ;8956   ; Test Description:
                ;8957   ;
                ;8958   ;       The purpose of this test is to verify that ACC SYNC is asserted in the
                ;8959   ;       FPA and skipped on in the CPU when SKIP.IF [SYNC] is present. This will
                ;8960   ;       be done by issuing a force from the CPU to a word in the FPA. Once the
                ;8961   ;       appropriate number of clock cycles have gone by, and ACC SYNC is
                ;8962   ;       asserted in the FPA, a SKIP.IF[SYNC] will be issued from the CPU. If
                ;8963   ;       there is no skip then an error will be issued.
                ;8964   ;
                ;8965   ; Assumptions:
                ;8966   ;
                ;8967   ;       It is assumed that a force to any Uaddress will work and that the
                ;8968   ;       word in the FPA that asserts ACC SYNC is corrrect.
                ;8969   ;
                ;8970   ; Test Steps:
                ;8971   ;
                ;8972   ;       1) Issue a FORCE from the CPU to the FPA to assert ACC SYNC.
                ;8973   ;       2) Wait two clock cycles for ACC SYNC to be asserted in the FPA.
                ;8974   ;       3) Issue a SKIP.IF[SYNC] in the CPU. If there is no skip then issue
                ;8975   ;          error 1.
                ;8976   ;
                ;8977   ; Error
                ;8978   ;
                ;8979   ;       Error1 - Either ACC SYNC failed to be asserted or the skip logic in
                ;8980   ;                in the CPU failed.
                ;8981   ;
                ;8982   ; Debug:
                ;8983   ;
                ;8984   ;       Error1: If this error occurs then both the FPA and the CPU are possible
                ;8985   ;               sources of error. If the FPA is in error then the latch in the
                ;8986   ;               middle left of page D should be checked for error. If the latch
                ;8987   ;               is not in error then  the error is probably in the CPU. If the
                ;8988   ;               CPU is thought to be the source of error then the USEQ.CTL PAL
                ;8989   ;               should be checked for error. ACC SYNC effects only three
                ;8990   ;               signals from the USEQ.CTL PAL, they are OR OUT 2 L, ENABLE UPC
                ;8991   ;               l, and ENABLE SP L. If these three signals are not asserted and
                ;8992   ;               ACC SYNC is asserted then the PAL is likely to be the cause of
                ;8993   ;               the error.
                ;8994   ;
                ;8995   ;
                ;8996   ;--
                ;8997   T.6:
U 090E, B65E,15 ;8998           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;8999                                           ; STATUS WORD
U 090F, 3E80,15 ;9000           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;9001                                           ; BIT 15 INDICATES START OF TEST TO
                ;9002                                           ; CONSOLE
U 0910, 10E0,15 ;9003           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;9004   WAIT.T6.0:
U 0911, 8891,14 ;9005           JMP [WAIT.T6.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
```

```
                    ;9006                                           ; RESPOND
U 0912, 087E,6C     ;9007           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;9008                                           ; AND SIZE BITS
U 0913, B646,15     ;9009           MOV LS[FPA] TO WR[0]             ;Store module code in LS to indicate
U 0914, 4742,15     ;9010           BIS LS[CPU] TO WR[0]             ; FPA and CPU modules.
U 0915, 3E8C,15     ;9011           MOV WR[0] TO LS[MODULE.NUM]
U 0916, B78E,15     ;9012           MOV LS[T6.POINTER] TO WR[0]      ;Initialize pointer for main memory
U 0917, BE12,15     ;9013           MOV WR[0] TO LS[T9]              ; references.
U 0918, B66E,15     ;9014           MOV LS[NA.EXP.REC] TO WR[0]      ;Set up phony data
U 0919, 3E80,15     ;9015           MOV WR[0] TO LS[CONTROL.STATUS]
U 091A, 8870,3C     ;9016           JSR [LO]                        ;Get Address to be forced
U 091B, BE14,15     ;9017           MOV WR[0] TO LS[T10]             ;Save addresses to force
U 091C, 3614,15     ;9018   T6.1:   MOV LS[T10] TO WR[0]            ;Restore registers
U 091D, 90F6,15     ;9019           MISC2 [TRAP.ACC] WR[0]          ;Force FPA word to assert ACC SYNC
U 091E, 2F82,95     ;9020           CLR WR[1]
U 091F, 2F80,15     ;9021           CLR WR[0]                       ;Set up expected and received data
U 0920, DB00,1F     ;9022           SKIP.IF[SYNC]                   ;Skip if ACC SYNC
U 0921, 2040,15     ;9023           INC WR[0]                       ;If no sync WR1=1, WR0=0
U 0922, 0869,3C     ;9024           JSR [CHECK.RESULT]              ; else WR1=0, WR0=0, and issue error
                    ;9025                                           ; if there is one.
U 0923, 0891,C4     ;9026           JMP [T6.1]                      ;Loop on error.
                    ;9027   T6.END:
```

```
;9028   .page
;9029   .TOC    "TEST 7 CPU FORCE TEST DURING FPA FAST CLOCK(M8389 FPA MODULE)"
;9030   ;++
;9031   ;
;9032   ;
;9033   ; Test Descrition:
;9034   ;
;9035   ;       The purpose of this test is to verify that the FPA will synchronize
;9036   ;       with the CPU when a FORCE instruction is issued from the CPU. Since
;9037   ;       all the microdiagnostic tests depend on the ability to force a
;9038   ;       microword at a specific address, if this test does not run without
;9039   ;       errors then it's likely to be the cause of error for all the other
;9040   ;       tests. This synchronization can take place on FPA PH0 or FPA PH1 and
;9041   ;       both of these cases will be tested. The testing will be done by forcing
;9042   ;       a word that will speed up the clock. The code following the word to
;9043   ;       speed up the clock will be to wait one cycle then issue a store of
;9044   ;       specific value. If there is never a CPU ATTN issued or the data
;9045   ;       that's stored is incorrect then an error will be issued. The second
;9046   ;       case will be tested by waiting two clock cycles instead of one.
;9047   ;
;9048   ; Assumptions:
;9049   ;
;9050   ;       It is assumed that all the tests previous to the control store test
;9051   ;       have been run without errors.
;9052   ;
;9053   ; Test Steps:
;9054   ;
;9055   ;       1) Issue a force to a routine that speeds up the FPA clock. Wait one
;9056   ;          clock cycle, then issue OPTION SYNC to initiate a store to the
;9057   ;          CPU. If there is no store or the data that is stored is incorrect,
;9058   ;          then issue error 1.
;9059   ;       2) Repeat step 1 except wait two clock cycles instead of one. If there
;9060   ;          is no store or the data that is stored is incorrect, then issue
;9061   ;          error 2.
;9062   ;
;9063   ; Error:
;9064   ;
;9065   ;       Error1 - Clock sync failed on FPA PH0.
;9066   ;       Error2 - Clock sync failed on FPA PH1.
;9067   ;
;9068   ; Debug:
;9069   ;
;9070   ;       Error1: If this error occurs then the CLOCK GENER'N PAL should be
;9071   ;               checked for error. When a force is being done then DAPK TRAP
;9072   ;               ACC L should be asserted. When the Clocks are running in fast
;9073   ;               mode then FPAC FAST CYCLE L and FPAC FAST PATH ENB H should
;9074   ;               both be asserted. FPAC FP PH0 L and FPAC FP PH1 H should be
;9075   ;               asserted alternating with each other in 90 second intervals
;9076   ;               with each other. If all of the outputs are correct then the
;9077   ;               gates above the CLOCK GENERATION PAL should be checked for
;9078   ;               error.
;9079   ;       Error2: Same as error 1 except that the sync failed of FPA PH1.
;9080   ;
;9081   ;
;9082   ;--
```

```
                    ;9083   T.7:
U 0924, B65E,15     ;9084           MOV LS[BEGIN.TEST] TO WR[0]      ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;9085                                            ; STATUS WORD
U 0925, 3E80,15     ;9086           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;9087                                            ; BIT 15 INDICATES START OF TEST TO
                    ;9088                                            ; CONSOLE
U 0926, 10E0,15     ;9089           MISC [SET.CP.ATTN]               ;ISSUE CPU ATTN TO CONSOLE
                    ;9090   WAIT.T7.0:
U 0927, 8892,74     ;9091           JMP [WAIT.T7.0]                  ;LOOP HERE WAITING FOR CONSOLE TO
                    ;9092                                            ; RESPOND
U 0928, 087E,6C     ;9093           JSR [SETUP]                      ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;9094                                            ; AND SIZE BITS
U 0929, B70C,15     ;9095           MOV LS[T7.POINTER] TO WR[0]      ;Initialize pointer for main memory
U 092A, BE12,15     ;9096           MOV WR[0] TO LS[T9]               ; references.
U 092B, B643,15     ;9097           MOV LS[#2] TO WR[2]              ;Load WR[2]  with 2 to check whether
                    ;9098                                            ; error 2 has been checked for
U 092C, B66E,15     ;9099           MOV LS[NA.EXP.REC] TO WR[0]      ;Set up phony data
U 092D, 3E80,15     ;9100           MOV WR[0] TO LS[CONTROL.STATUS]
U 092E, 8870,3C     ;9101           JSR [L0]                         ;Get address to be forced.
U 092F, 0870,8C     ;9102           JSR [L1]
U 0930, BE14,15     ;9103           MOV WR[0] TO LS[T10]              ;Save registers
U 0931, BE16,95     ;9104           MOV WR[1] TO LS[T11]
                    ;9105   T7.2:   CMP WR[2] WITH LS[ERROR.NUMBER],;Set the CC's for branching on one
U 0932, CF83,35     ;9106            DT(LONG)&SET.ALU.CC             ; cycle wait
U 0933, 3614,15     ;9107   T7.1:   MOV LS[T10] TO WR[0]             ;Restore registers
U 0934, 3616,95     ;9108           MOV LS[T11] TO WR[1]
U 0935, 90F6,15     ;9109           MISC2 [TRAP.ACC] WR[0]           ;Issue force to FPA routine that speeds
                    ;9110                                            ; up the clock and loops on self
                    ;9111                                            ; waiting for the next force from the
                    ;9112                                            ; CPU.
U 0936, DB00,01     ;9113           SKIP.IF[NEQ]                     ;If error 1 then no skip
U 0937, DB00,15     ;9114           NOP
U 0938, 10F6,95     ;9115           MISC2 [TRAP.ACC] WR[1]           ;Force to an FPA word which asserts ACC
                    ;9116                                            ; SYNC and then goes to a word which
                    ;9117                                            ; initiates a store of the FIRST.FLT
U 0939, 2F82,95     ;9118           CLR WR[1]                        ;Set up expected and received data.
U 093A, 2F80,15     ;9119           CLR WR[0]
U 093B, DB00,1F     ;9120           SKIP.IF[SYNC]                    ;Skip if SYNC is asserted
U 093C, 2042,95     ;9121           INC WR[1]                        ;If not asserted then there's an error
                    ;9122                                            ; so WR1=1 and WR0=0
U 093D, 0869,3C     ;9123           JSR [CHECK.RESULT]               ;If there's an error report it.
U 093E, 8893,34     ;9124           JMP [T7.1]                       ;Loop on error.
                    ;9125           CMP WR[2] WITH LS[ERROR.NUMBER],;If error 2 in test 7 has been checked
U 093F, CF83,35     ;9126            DT(LONG)&SET.ALU.CC             ; for then go to the end of test 7
U 0940, 8894,39     ;9127           JMP.IF[EQL] TO [T7.END]          ; else set up for error 2 and jump to
                    ;9128                                            ; T7.2
U 0941, FF82,15     ;9129           INC LS[ERROR.NUMBER]
U 0942, 0893,24     ;9130           JMP [T7.2]
                    ;9131   T7.END:
```

```
                ;9132   .page
                ;9133   .TOC    "TEST 8 CPU DATA AVAIL AND ZERO BRANCH TEST(M8389 FPA OR M8390 CPU MODULE)"
                ;9134   ;++
                ;9135   ;
                ;9136   ;
                ;9137   ; Test Description:
                ;9138   ;
                ;9139   ;       The purpose of this test is to check the branch conditions, CPU DATA
                ;9140   ;       AVAIL and ZERO. The data path that will be used is the following: A
                ;9141   ;       MISC instruction will be issued from the CPU to force a nop in the
                ;9142   ;       FPA with the branch field set to 01010. Then a MISC instruction will
                ;9143   ;       be issued to assert CPU DATA AVAIL, this should cause the branch.
                ;9144   ;       This procedure will be repeated for CPU  DATA AVAIL unasserted and for
                ;9145   ;       the next branch condition, 01011.
                ;9146   ;
                ;9147   ; Assumptions:
                ;9148   ;
                ;9149   ;       It is assumed that the branch control logic has been tested previously
                ;9150   ;       and works.
                ;9151   ;
                ;9152   ; Test Steps:
                ;9153   ;
                ;9154   ;       1) Issue a MISC instruction to force a nop in the FPA with the branch
                ;9155   ;          field set to 01010. Then issue a MISC instruction to assert CPU
                ;9156   ;          DATA AVAIL. If there is no branch or an incorrect branch then issue
                ;9157   ;          error1.
                ;9158   ;       2) Issue a nop with the branch field set to 01010 and CPU DATA AVAIL
                ;9159   ;          unasserted. If there is any branch then issue error2.
                ;9160   ;
                ;9161   ; Error:
                ;9162   ;
                ;9163   ;       Error1 - Branch failed when branch field 01010 and CPU DATA AVAIL
                ;9164   ;                asserted.
                ;9165   ;       Error2 - Branch failed when branch field 01010 and CPU DATA AVAIL
                ;9166   ;                unasserted.
                ;9167   ;
                ;9168   ; Debug:
                ;9169   ;
                ;9170   ;       Error1: If this error occurs then the BRANCH 1 PAL should be checked
                ;9171   ;               for error. If the PAL isn't the cause of the error then the
                ;9172   ;               then the UBCTL bits should be checked to be 01010.If the
                ;9173   ;               UBCTL bits are in error then the latch that they come from
                ;9174   ;               should be checked for errors. If all the logic on the FPA
                ;9175   ;               is alright then DAPK CPU DATA AVAIL from the CPU should be
                ;9176   ;               checked to be asserted. If the problem is in the CPU then
                ;9177   ;               the MISC INSTR DECODER should be checked for error.
                ;9178   ;       Error2: Same as error1 except CPU DATA AVAIL should not be asserted.
                ;9179   ;
                ;9180   ;
                ;9181   ;--
                ;9182   T.8:
U 0943, B65E,15 ;9183           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;9184                                           ; STATUS WORD
U 0944, 3E80,15 ;9185           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;9186                                           ; BIT 15 INDICATES START OF TEST TO
```

```
                 ;9187                                  ; CONSOLE
U 0945, 10E0,15  ;9188          MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE
                 ;9189  WAIT.T8.0:
U 0946, 0894,64  ;9190          JMP [WAIT.T8.0]         ;LOOP HERE WAITING FOR CONSOLE TO
                 ;9191                                  ; RESPOND
U 0947, 887E,EC  ;9192          JSR [SETUP.10]          ;SET UP ERROR INFO AND CLEAR INSTRUC
                 ;9193                                  ; AND SIZE BITS. CHECK LOWER 10 BITS
U 0948, B646,15  ;9194          MOV LS[FPA] TO WR[0]    ;Store module code in LS to indicate
U 0949, 4742,15  ;9195          BIS LS[CPU] TO WR[0]    ; the FPA or the CPU module
U 094A, 3E8C,15  ;9196          MOV WR[0] TO LS[MODULE.NUM]
U 094B, 370E,15  ;9197          MOV LS[T8.POINTER] TO WR[0]  ;Initialize pointer for main memory
U 094C, BE12,15  ;9198          MOV WR[0] TO LS[T9]     ; references.
U 094D, 0870,8C  ;9199          JSR [L1]                ;Get expected data.
U 094E, 3E14,95  ;9200          MOV WR[1] TO LS[T10]    ;Save expected data
U 094F, 8871,2C  ;9201          JSR [T84]               ;Set address to be forced.
U 0950, 3708,15  ;9202  T8.1:   MOV LS[T84] TO WR[0]
U 0951, 90F6,15  ;9203          MISC2 [TRAP.ACC] WR[0]  ;Force the BUT
U 0952, 10F4,15  ;9204          MISC2 [ASSERT.DA.AND.PASS.WR]  ;Assert CPU DATA AVAIL and branch
U 0953, 10F8,15  ;9205          MISC2 [READ.ACC.UPC]    ;Enable the NUA BUS onto the FPA BUS
U 0954, 3A18,15  ;9206          MOV FPA TO LS[T12]      ;Enable the FPA BUS onto the Y BUS
U 0955, 3618,15  ;9207          MOV LS[T12] TO WR[0]    ;Set up received data
U 0956, B614,95  ;9208          MOV LS[T10] TO WR[1]    ;Set up expected data
U 0957, 0869,3C  ;9209          JSR [CHECK.RESULT]      ;If there's an error report it
U 0958, 8895,04  ;9210          JMP [T8.1]              ;Loop on error
U 0959, FF82,15  ;9211          INC LS[ERROR.NUMBER]    ;Go on to next error
U 095A, 0870,8C  ;9212          JSR [L1]                ;Get expected data
U 095B, 3E14,95  ;9213          MOV WR[1] TO LS[T10]    ;Save expected data
U 095C, 3708,15  ;9214  T8.2:   MOV LS[T84] TO WR[0]
U 095D, 90F6,15  ;9215          MISC2 [TRAP.ACC] WR[0]  ;Force the BUT
U 095E, DB00,15  ;9216          NOP
U 095F, 10F8,15  ;9217          MISC2 [READ.ACC.UPC]    ;Enable the NUA BUS onto the FPA BUS
U 0960, 3A18,15  ;9218          MOV FPA TO LS[T12]      ;Enable the FPA BUS onto the Y BUS
U 0961, 3618,15  ;9219          MOV LS[T12] TO WR[0]    ;Set up recieved data
U 0962, B614,95  ;9220          MOV LS[T10] TO WR[1]    ;Setup expected data
U 0963, 0869,3C  ;9221          JSR [CHECK.RESULT]      ;If there's an error report it
U 0964, 8895,C4  ;9222          JMP [T8.2]              ;Loop on error
                 ;9223  T8.END:
```

```
;9224   .page
;9225   .TOC    "TEST 9 ZEROS FROM THE 2901 TEST(M8389 FPA MODULE)"
;9226   ;++
;9227   ;
;9228   ;
;9229   ; Test Description:
;9230   ;
;9231   ;       The purpose of this test is to check to see if zeros can be stored from
;9232   ;       the 2901's before we try and load data into and read it out again.
;9233   ;       The fraction data path control signals will be A and 0 for the operands
;9234   ;       R AND S for the function and F -> B for the destination. The exponent
;9235   ;       data path control signals will be 0 and the SCRATCH PADS will be the
;9236   ;       destination. A STORE of the WR addressed by the B register is used to
;9237   ;       STORE that WR into the CPU. If the contents of the WR aren't zeros then
;9238   ;       an error statement will be issued.
;9239   ;
;9240   ; Assumptions:
;9241   ;
;9242   ;       It is assumed that the NUA lines and the Control Store have been
;9243   ;       checked for errors and are free of them.
;9244   ;
;9245   ; Test Steps:
;9246   ;
;9247   ;       1) Issue a Force to an instruction that AND's the D inputs and 0. Read
;9248   ;          the contents of the resultant register back to the CPU and if the
;9249   ;          contents aren't all zeros then issue error1.
;9250   ;
;9251   ; Error:
;9252   ;
;9253   ;       NOTE: When other data is 1 the bits 31 - 55 of the fraction data path
;9254   ;             and bits 0 - 7 of the exponent path are being checked. When other
;9255   ;             data is 2 then bits 0 - 31 of the fraction path are being checked
;9256   ;
;9257   ;       Error1 - D 'ANDED' 0 failed to give zeros.
;9258   ;
;9259   ; Debug:
;9260   ;
;9261   ;       Error1: Since this test is the first test to use the data path through
;9262   ;               the 2901's there are numerous errors that can occur. The errors
;9263   ;               fall into three basic categories. They are direct input failure
;9264   ;               2901 enable failure, and control line failure.
;9265   ;
;9266   ;               DIRECT INPUT FAILURE
;9267   ;
;9268   ;               The data path from the CPU to the direct inputs of the 2901's
;9269   ;               are considered to be a direct input failure. If the data fails
;9270   ;               only in bytes, the tranceiver associated with that byte is a
;9271   ;               possible cause of the error. If both the first and second
;9272   ;               floats of data are failing then the control logic to the
;9273   ;               tranceivers should be checked during the load cycle in the FPA.
;9274   ;               FPAB ENBCP LOAD L is the signal likely to be in error.
;9275   ;               IF FPAB ENBCP LOAD L is high then the INSTRUCTION PAL should be
;9276   ;               checked for error. FPAD PAR ERR H and FPAA UADDR(1)H (inputs to
;9277   ;               the INSTRUCTION PAL) should be low and FPAE LOAD H should be
;9278   ;               high. If these signals aren't in error INSTRUCTION PAL is
```

```
                ;9279   ;               likely to be the cause of error. If FPAE LOAD H is low then the
                ;9280   ;               EXT FUNC CTL PAL may be in error. The clock inputs should be
                ;9281   ;               011 and EXTEND CLK(1) should be high. CPU DATA AVAIL should be
                ;9282   ;               low. If bits 0 - 7 are the only bits failing then the DATA IN
                ;9283   ;               CTL PAL should be checked for error. The LO SEL inputs to the
                ;9284   ;               PAL should both be low. If the LO SEL inputs are correct then
                ;9285   ;               the DATA IN CTL PAL is probably the cause of error. If LO SEL 0
                ;9286   ;               is in error then SIZE 1 should be checked to be low. If SIZE 1
                ;9287   ;               is low then the EXT FUNC PAL should be checked for error. If LO
                ;9288   ;               SEL 1 is high and SIZE 1 is low then the BRANCH 0 PAL is prob-
                ;9289   ;               ably the cause of error.
                ;9290   ;
                ;9291   ;               2901 ENABLE
                ;9292   ;
                ;9293   ;               If the first eight bits are in error then FPA L ENB FRAC
                ;9294   ;               <55:48> L should be checked to be low. If the second eight
                ;9295   ;               bits are in error the FPAL EXP<7:0> ENB L should be checked
                ;9296   ;               to be low. If bits 16 - 31 are in error then FPAL ENB <47:32> L
                ;9297   ;               should be checked to be low. If any of the enable signals are
                ;9298   ;               in error the STORE CTL PAL is likely to be the cause. The MOD
                ;9299   ;               inputs to the PAL should be 11 and the SHF inputs to the PAL
                ;9300   ;               should be 00. If the inputs are correct the STORE CTL PAL is
                ;9301   ;               probably the cause of the error.
                ;9302   ;
                ;9303   ;               CONTROL LINE FAILURE
                ;9304   ;
                ;9305   ;               Refer to the 2901 ALU CONTROL DESCRIPTION section of this doc-
                ;9306   ;               ument in order to determine what the control bits should be.
                ;9307   ;
                ;9308   ;
                ;9309   ;--
                ;9310   T.9:
U 0965, B65E,15 ;9311           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;9312                                           ; STATUS WORD
U 0966, 3E80,15 ;9313           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;9314                                           ; BIT 15 INDICATES START OF TEST TO
                ;9315                                           ; CONSOLE
U 0967, 10E0,15 ;9316           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;9317   WAIT.T9.0:
U 0968, 0896,84 ;9318           JMP [WAIT.T9.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;9319                                           ; RESPOND
U 0969, 087E,6C ;9320           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;9321                                           ; AND SIZE BITS
U 096A, B65E,15 ;9322           MOV LS[BIT15] TO WR[0]          ;Set up error mask to check lower 10
U 096B, 3E8A,15 ;9323           MOV WR[0] TO LS[ERROR.MASK]     ; bits.
U 096C, 0874,3C ;9324           JSR [CLR.INST.AND.SIZE.BITS]
U 096D, B79C,15 ;9325           MOV LS[T9.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 096E, BE12,15 ;9326           MOV WR[0] TO LS[T9]
U 096F, B670,15 ;9327           MOV LS[OTHER.DATA] TO WR[0]     ;Set up other data fiald
U 0970, 3E80,15 ;9328           MOV WR[0] TO LS[CONTROL.STATUS]
U 0971, E5A2,15 ;9329           CLR LS[DATA.1]
U 0972, E5A4,15 ;9330           CLR LS[DATA.2]
U 0973, 8870,3C ;9331           JSR [LO]                        ;Get address of CLR FWR and CLR EWR
U 0974, BE1E,15 ;9332           MOV WR[0] TO LS[T15]
                ;9333   T9.LOOP1:
```

```
U 0975, 361E,15 ;9334          MOV LS[T15] TO WR[0]          ;Set up to CLR FWR and CLR EWR
U 0976, B716,95 ;9335          MOV LS[#300] TO WR[1]         ;Setup to Loop self
U 0977, 90F6,15 ;9336          MISC2 [TRAP.ACC] WR[0]        ;Clear FWR and EWR
U 0978, 10F6,95 ;9337          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 0979, 087B,5C ;9338          JSR [STORE.0.TRIPLE]
U 097A, B640,15 ;9339          MOV LS[BIT0] TO WR[0]         ;Set up other data
U 097B, BE88,15 ;9340          MOV WR[0] TO LS[ADDRESS.DATA]
U 097C, B65E,15 ;9341          MOV LS[BIT15] TO WR[0]        ;Change error mask back
U 097D, 3E8A,15 ;9342          MOV WR[0] TO LS[ERROR.MASK]
U 097E, 36A2,95 ;9343          MOV LS[DATA.1] TO WR[1]       ;Set up expected data
U 097F, B6A8,15 ;9344          MOV LS[FIRST.FLT] TO WR[0]    ;Set up received data
U 0980, 0869,3C ;9345          JSR [CHECK.RESULT]
U 0981, 0897,54 ;9346          JMP [T9.LOOP1]
U 0982, 3642,15 ;9347          MOV LS[BIT1] TO WR[0]         ;Set up other data
U 0983, BE88,15 ;9348          MOV WR[0] TO LS[ADDRESS.DATA]
U 0984, E58A,15 ;9349          CLR LS[ERROR.MASK]            ;Change the error mask
U 0985, 36A4,95 ;9350          MOV LS[DATA.2] TO WR[1]       ;Set up expected data
U 0986, 36AA,15 ;9351          MOV LS[SEC.FLT] TO WR[0]      ;Set up received data
U 0987, 0869,3C ;9352          JSR [CHECK.RESULT]            ;If an error report it
U 0988, 0897,54 ;9353          JMP [T9.LOOP1]                ;Loop on error
                ;9354  T9.END:
```

```
;9355    .page
;9356    .TOC    "TEST A WORKING REGISTER TEST (M8389 FPA MODULE)"
;9357    ;++
;9358    ;
;9359    ; Test Description:
;9360    ;
;9361    ;       The data paths of the FPA to the 2901's can be manipulated by loading
;9362    ;       data onto the FPA BUS through a load routine in the FPA Control Store.
;9363    ;       The load routine in the FPA Control Store will consist of an instruc-
;9364    ;       tion to toggle the load bit. The toggle will occur if there's a 1 in
;9365    ;       the MOD field and a 4 in the CLK field. The next micro instruction
;9366    ;       will load the FPA WR from the Y BUS of the CPU. The first cycle of data
;9367    ;       must come from WR[0] or WR[1] but the three cycles following (for a
;9368    ;       huge load) will be whatever is loaded onto the Y BUS by the CPU. On a
;9369    ;       huge load the first cycle will load the sign bit, the exponent bits
;9370    ;       and some of the fraction bits. The second and third cycles will load
;9371    ;       more of the fraction bits and the extention bits, the fourth cycle
;9372    ;       will load the remaining fraction bits . At the end of the load
;9373    ;       routine there will be a jump to self. A force to an FPA micro address
;9374    ;       that will load the WR of the 2901's onto the FPA BUS is executed by
;9375    ;       the CPU code. The jump field in the FPA micro word will be to a jump
;9376    ;       to self. The next CPU instruction will be to read the data from the
;9377    ;       FPA BUS onto the Y BUS and into a location in local store. The data
;9378    ;       will be read into four LS locations where it can be checked against
;9379    ;       the data that originally loaded onto the FPA BUS. The WR of the FPA
;9380    ;       that can't be directly tested in this manner are the upper two 2901's
;9381    ;       of the exponent data path and the odd WR of the extention data path.
;9382    ;       These will be tested later with the shift mechanism.
;9383    ;
;9384    ; Assumptions:
;9385    ;
;9386    ;       It is assumed that the next micro address logic has been checked and
;9387    ;       the control logic associated with enabling the Y BUS, the FPA BUS and
;9388    ;       the NUA BUS has been tested and works. The parity logic and the parity
;9389    ;       of the micro words in the FPA control Store have also been tested and
;9390    ;       been found to be correct.
;9391    ;
;9392    ; Test Steps:
;9393    ;
;9394    ;       1) Get the address of the double load routine
;9395    ;       2) Set up and save the pointer into memory.
;9396    ;       3) Get the data patterns to be passed to the FPA to test the first and
;9397    ;          second floats of the FPA data path and put them in DATA.1 and DATA.2
;9398    ;       4) Put the second float in a WR so it can be put on the Y BUS
;9399    ;          immediately following the first float load.
;9400    ;       5) Restore the pointer into memory in order to be able to loop on error.
;9401    ;       6) Force the FPA routine to do a double load.
;9402    ;       7) Load the first float of data into WR[0]. Then move the second
;9403    ;          float of data to LS putting it on the Y BUS so it will be picked up
;9404    ;          by the FPA.
;9405    ;       8) Get the address of the store routine. Then force that address.
;9406    ;       9) Store the first float of data in an LS location called FIRST.FLT
;9407    ;          and the second in a location called SEC.FLT.
;9408    ;      10) Check for error in the first float and report if there is one.
;9409    ;      11) Check for error in the second float and report if there is one.
```

```
;9410   ;       13) If all patterns are tested then repeat the procedure for WR[2]
;9411   ;           through WR[F] and Q, if not then go back and get the next pattern.
;9412   ;
;9413   ; Error:
;9414   ;
;9415   ;       NOTE: Other data in this case refers to whether it is the first or
;9416   ;             second float of data that is being tested.
;9417   ;
;9418   ;       Error1 - WR[0] or the associated logic is in error.
;9419   ;       Error2 - WR[1] or the associated logic in in error.
;9420   ;       Error3 - WR[2] or the associated logic in in error.
;9421   ;       Error4 - WR[3] or the associated logic in in error.
;9422   ;       Error5 - WR[4] or the associated logic in in error.
;9423   ;       Error6 - WR[5] or the associated logic in in error.
;9424   ;       Error7 - WR[6] or the associated logic in in error.
;9425   ;       Error8 - WR[7] or the associated logic in in error.
;9426   ;       Error9 - WR[8] or the associated logic in in error.
;9427   ;       ErrorA - WR[9] or the associated logic in in error.
;9428   ;       ErrorB - WR[A] or the associated logic in in error.
;9429   ;       ErrorC - WR[B] or the associated logic in in error.
;9430   ;       ErrorD - WR[C] or the associated logic in in error.
;9431   ;       ErrorE - WR[D] or the associated logic in in error.
;9432   ;       ErrorF - WR[E] or the associated logic in in error.
;9433   ;       Error10 - WR[F] or the associated logic in in error.
;9434   ;       Error11 - QR or the associated logic in in error.
;9435   ;
;9436   ; Debug:
;9437   ;
;9438   ;       Error1: Since this test is the first test to use the data path through
;9439   ;               the 2901's there are numerous errors that can occur. The errors
;9440   ;               fall into three basic categories. They are direct input failure
;9441   ;               2901 enable failure, and control line failure.
;9442   ;
;9443   ;               DIRECT INPUT FAILURE
;9444   ;
;9445   ;               The data path from the CPU to the direct inputs of the 2901's
;9446   ;               considered to direct input failure. If the data fails only in
;9447   ;               bytes the tranceiver associated with that byte is a possible
;9448   ;               cause of the error. If both the first and second floats of data
;9449   ;               are failing then the control logic to the tranceivers should be
;9450   ;               checked. FPAB ENBCP LOAD L is the signal likely to be in error.
;9451   ;               IF FPAB ENBCP LOAD L is high then the INSTRUCTION PAL should be
;9452   ;               checked for error. FPAD PAR ERR H and FPAA UADDR(1)H (inputs to
;9453   ;               the INSTRUCTION PAL) should be low and FPAE LAOD H should be
;9454   ;               high. If these signals aren't in error INSTRUCTION PAL is
;9455   ;               likely to be the cause of error. If FPAE LOAD H is low then the
;9456   ;               EXT FUNC CTL PAL may be in error. The clock inputs should be
;9457   ;               011 and EXTEND CLK(1) should be high. CPU DATA AVAIL should be
;9458   ;               low. If bits 0 - 7 are the only bits failing then the DATA IN
;9459   ;               CTL PAL should be checked for error. The LO SEL inputs to the
;9460   ;               PAL should both be low. If the LO SEL inputs are correct then
;9461   ;               the DATA IN CTL PAL is probably the cause of error. If LO SEL 0
;9462   ;               is in error then SIZE 1 should be checked to be low. If SIZE 1
;9463   ;               is low then the EXT FUNC PAL should be checked for error. If LO
;9464   ;               SEL 1 is high and SIZE 1 is low then the BRANCH 0 PAL is prob-
```

```
                ;9465   ;               ably the cause of error.
                ;9466   ;
                ;9467   ;               2901 ENABLE
                ;9468   ;
                ;9469   ;               If the first eight bits are in error then FPA L ENB FRAC
                ;9470   ;               <55:48> L should be checked to be low. If the second eight
                ;9471   ;               bits are in error the FPAL EXP<7:0> ENB L should be checked
                ;9472   ;               to be low. If bits 16 - 31 are in error then FPAL ENB <47:32> L
                ;9473   ;               should be checked to be low. If any of the enable signals are
                ;9474   ;               in error the STORE CTL PAL is likely to be the cause. The MOD
                ;9475   ;               inputs to the PAL should be 11 and the SHF inputs to the PAL
                ;9476   ;               should be 00. If the inputs are correct the STORE CTL PAL is
                ;9477   ;               probably the cause of the error.
                ;9478   ;
                ;9479   ;               CONTROL LINE FAILURE
                ;9480   ;
                ;9481   ;               Refer to the 2901 ALU CONTROL DESCRIPTION section of this doc-
                ;9482   ;               ument in order to determine what the control bits should be.
                ;9483   ;       Error2: Same as error 1.
                ;9484   ;       Error3: Same as error 1.
                ;9485   ;       Error4: Same as error 1.
                ;9486   ;       Error5: Same as error 1.
                ;9487   ;       Error6: Same as error 1.
                ;9488   ;       Error7: Same as error 1.
                ;9489   ;       Error8: Same as error 1.
                ;9490   ;       Error9: Same as error 1.
                ;9491   ;       ErrorA: Same as error 1.
                ;9492   ;       ErrorB: Same as error 1.
                ;9493   ;       ErrorC: Same as error 1.
                ;9494   ;       ErrorD: Same as error 1.
                ;9495   ;       ErrorE: Same as error 1.
                ;9496   ;       ErrorF: Same as error 1.
                ;9497   ;       Error10: Same as error 1.
                ;9498   ;       Error11: Same as error 1.
                ;9499   ;
                ;9500   ;
                ;9501   ;-
                ;9502   T.A:
U 0989, B65E,15 ;9503           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;9504                                           ; STATUS WORD
U 098A, 3E80,15 ;9505           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;9506                                           ; BIT 15 INDICATES START OF TEST TO
                ;9507                                           ; CONSOLE
U 098B, 10E0,15 ;9508           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;9509   WAIT.TA.0:
U 098C, 0898,C4 ;9510           JMP [WAIT.TA.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;9511                                           ; RESPOND
U 098D, 087E,6C ;9512           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;9513                                           ; AND SIZE BITS
U 098E, B65E,15 ;9514           MOV LS[BIT15] TO WR[0]          ;Set up error mask
U 098F, 3E8A,15 ;9515           MOV WR[0] TO LS[ERROR.MASK]
U 0990, B670,15 ;9516           MOV LS[OTHER.DATA] TO WR[0]     ;Set up other data
U 0991, 3E80,15 ;9517           MOV WR[0] TO LS[CONTROL.STATUS]
U 0992, 3710,15 ;9518           MOV LS[TA.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 0993, BE12,15 ;9519           MOV WR[0] TO LS[T9]             ; references.
```

```
U 0994, 2F85,15 ;9520            CLR WR[2]                          ;Clear pointer in GET DATA subroutine
U 0995, 0870,8C ;9521            JSR [L1]
U 0996, BE10,95 ;9522            MOV WR[1] TO LS[T8]
U 0997, 8870,3C ;9523            JSR [L0]                           ;Get the address of STORE FF FWR[FTO]
U 0998, 3E16,15 ;9524            MOV WR[0] TO LS[T11]
U 0999, 8870,3C ;9525            JSR [L0]                           ;Get the address of STORE SF FWR[FTO]
U 099A, BE18,15 ;9526            MOV WR[0] TO LS[T12]
                ;9527    TA.LOOP0:
U 099B, 8871,7C ;9528            JSR [GET.DATA]                     ;Get data to test WR.
U 099C, 36A4,95 ;9529    TA.0:   MOV LS[DATA.2] TO WR[1]            ;Put second float of data in WR 1
U 099D, B610,15 ;9530            MOV LS[T8] TO WR[0]                ;Get address of load routine
U 099E, 90F6,15 ;9531            MISC2 [TRAP.ACC] WR[0]             ;Force address of load routine
U 099F, B6A2,15 ;9532            MOV LS[DATA.1] TO WR[0]            ;Wait for toggle load and set up
                ;9533                                               ; data to be passed to the FPA.
U 09A0, 10F4,15 ;9534            MISC2 [ASSERT.DA.AND.PASS.WR]      ;CPU syncs with FPA and passes WR 0
U 09A1, 3E14,95 ;9535            MOV WR[1] TO LS[T10]               ; and a longword of 0's to the FPA
U 09A2, B616,15 ;9536            MOV LS[T11] TO WR[0]               ;Get address of STORE FIRST FLOAT
U 09A3, B618,95 ;9537            MOV LS[T12] TO WR[1]               ;Get address of STORE SECOND FLOAT
U 09A4, 90F6,15 ;9538            MISC2 [TRAP.ACC] WR[0]             ;Force word to store first float
U 09A5, BAA8,15 ;9539            MOV FPA TO LS[FIRST.FLT]
U 09A6, 10F6,95 ;9540            MISC2 [TRAP.ACC] WR[1]             ;Get second float
U 09A7, 3AAA,15 ;9541            MOV FPA TO LS[SEC.FLT]
U 09A8, B640,15 ;9542            MOV LS[BIT0] TO WR[0]              ;Set up other data
U 09A9, BE88,15 ;9543            MOV WR[0] TO LS[ADDRESS.DATA]
U 09AA, B65E,15 ;9544            MOV LS[BIT15] TO WR[0]             ;Setup error mask for first float
U 09AB, 3E8A,15 ;9545            MOV WR[0] TO LS[ERROR.MASK]
U 09AC, 36A2,95 ;9546            MOV LS[DATA.1] TO WR[1]            ;Set up expected data
U 09AD, B6A8,15 ;9547            MOV LS[FIRST.FLT] TO WR[0]         ;Set up received data
U 09AE, 0869,3C ;9548            JSR [CHECK.RESULT]
U 09AF, 8899,C4 ;9549            JMP [TA.0]
U 09B0, 3642,15 ;9550            MOV LS[BIT1] TO WR[0]              ;Set up other data
U 09B1, BE88,15 ;9551            MOV WR[0] TO LS[ADDRESS.DATA]
U 09B2, E58A,15 ;9552            CLR LS[ERROR.MASK]                 ;Setup error mask for second float
U 09B3, 36A4,95 ;9553            MOV LS[DATA.2] TO WR[1]            ;Set up expected data
U 09B4, 36AA,15 ;9554            MOV LS[SEC.FLT] TO WR[0]           ;Set up received data
U 09B5, 0869,3C ;9555            JSR [CHECK.RESULT]                 ;If an error report it
U 09B6, 8899,C4 ;9556            JMP [TA.0]                         ;Loop on error
U 09B7, 3714,95 ;9557            MOV LS[#6] TO WR[1]
                ;9558            CMP WR[2] WITH WR[1],              ;All data tested?
U 09B8, A644,B5 ;9559             DT(LONG)&SET.ALU.CC
U 09B9, 0899,B1 ;9560            JMP.IF[NEQ] TO [TA.LOOP0]          ;If not then continue looping
                ;9561                                               ;Else go on to next register
                ;9562    TA.NEXT.REGISTER:
U 09BA, FF82,15 ;9563            INC LS[ERROR.NUMBER]               ;Go on to next error
U 09BB, 2F85,15 ;9564            CLR WR[2]                          ;Clear pointer to GET DATA subroutine
U 09BC, 8870,3C ;9565            JSR [L0]                           ;Get address of MOV instruction
U 09BD, 3E1A,15 ;9566            MOV WR[0] TO LS[T13]
U 09BE, 8870,3C ;9567            JSR [L0]                           ;Get address of FIRST FLOAT STORE
U 09BF, 3E16,15 ;9568            MOV WR[0] TO LS[T11]
U 09C0, 8870,3C ;9569            JSR [L0]                           ;Get address of SECOND FLOAT STORE
U 09C1, BE18,15 ;9570            MOV WR[0] TO LS[T12]
                ;9571    TA.LOOP1:
U 09C2, 8871,7C ;9572            JSR [GET.DATA]                     ;Get data to test WR.
U 09C3, 36A4,95 ;9573    TA.1:   MOV LS[DATA.2] TO WR[1]            ;Put second float of data in WR 1
U 09C4, B610,15 ;9574            MOV LS[T8] TO WR[0]                ;Get address of load routine
```

```
U 09C5, 90F6,15 ;9575           MISC2 [TRAP.ACC] WR[0]          ;Force address of load routine
U 09C6, B6A2,15 ;9576           MOV LS[DATA.1] TO WR[0]         ;Wait for toggle load and set up
                ;9577                                           ; data to be passed to the FPA.
U 09C7, 10F4,15 ;9578           MISC2 [ASSERT.DA.AND.PASS.WR]   ;CPU syncs with FPA and passes WR 0
U 09C8, 3E14,95 ;9579           MOV WR[1] TO LS[T10]            ; and a longword of 0's to the FPA
U 09C9, B61A,15 ;9580           MOV LS[T13] TO WR[0]
U 09CA, B716,95 ;9581           MOV LS[#300] TO WR[1]           ;Get loop address
U 09CB, 90F6,15 ;9582           MISC2 [TRAP.ACC] WR[0]          ;Force move instruction
U 09CC, 10F6,95 ;9583           MISC2 [TRAP.ACC] WR[1]          ;Force loop instruction
U 09CD, B616,15 ;9584           MOV LS[T11] TO WR[0]            ;Get address of FIRST FLOAT STORE
U 09CE, B618,95 ;9585           MOV LS[T12] TO WR[1]            ;Get address of SECOND FLOAT STORE
U 09CF, 90F6,15 ;9586           MISC2 [TRAP.ACC] WR[0]          ;Force word to store first float
U 09D0, BAA8,15 ;9587           MOV FPA TO LS[FIRST.FLT]
U 09D1, 10F6,95 ;9588           MISC2 [TRAP.ACC] WR[1]          ;Get second float
U 09D2, 3AAA,15 ;9589           MOV FPA TO LS[SEC.FLT]
U 09D3, B640,15 ;9590           MOV LS[BIT0] TO WR[0]           ;Set up other data
U 09D4, BE88,15 ;9591           MOV WR[0] TO LS[ADDRESS.DATA]
U 09D5, B65E,15 ;9592           MOV LS[BIT15] TO WR[0]          ;Setup error mask for first float
U 09D6, 3E8A,15 ;9593           MOV WR[0] TO LS[ERROR.MASK]
U 09D7, 36A2,95 ;9594           MOV LS[DATA.1] TO WR[1]         ;Set up expected data
U 09D8, B6A8,15 ;9595           MOV LS[FIRST.FLT] TO WR[0]      ;Set up received data
U 09D9, 0869,3C ;9596           JSR [CHECK.RESULT]
U 09DA, 889C,34 ;9597           JMP [TA.1]
U 09DB, 3642,15 ;9598           MOV LS[BIT1] TO WR[0]           ;Set up other data
U 09DC, BE88,15 ;9599           MOV WR[0] TO LS[ADDRESS.DATA]
U 09DD, E58A,15 ;9600           CLR LS[ERROR.MASK]              ;Setup error mask for second float
U 09DE, 36A4,95 ;9601           MOV LS[DATA.2] TO WR[1]         ;Set up expected data
U 09DF, 36AA,15 ;9602           MOV LS[SEC.FLT] TO WR[0]        ;Set up received data
U 09E0, 0869,3C ;9603           JSR [CHECK.RESULT]              ;If an error report it
U 09E1, 889C,34 ;9604           JMP [TA.1]                      ;Loop on error
U 09E2, 3714,95 ;9605           MOV LS[#6] TO WR[1]
                ;9606           CMP WR[2] WITH WR[1],           ;All data tested?
U 09E3, A644,B5 ;9607            DT(LONG)&SET.ALU.CC
U 09E4, 089C,21 ;9608           JMP.IF[NEQ] TO [TA.LOOP1]       ;If not then continue looping
                ;9609                                           ;Else go on to next register
U 09E5, 3648,15 ;9610           MOV LS[#10] TO WR[0]
                ;9611           CMP WR[0] WITH LS[ERROR.NUMBER],
U 09E6, 4F82,35 ;9612            DT(LONG)&SET.ALU.CC
U 09E7, 089B,A1 ;9613           JMP.IF[NEQ] TO [TA.NEXT.REGISTER]
U 09E8, FF82,15 ;9614           INC LS[ERROR.NUMBER]            ;Go on to next error
U 09E9, 2F85,15 ;9615           CLR WR[2]                       ;Clear pointer for GET DATA subroutine
U 09EA, 8870,3C ;9616           JSR [L0]                        ;Get address of MOV FWR[FT0] TO FQ
U 09EB, 3E1A,15 ;9617           MOV WR[0] TO LS[T13]
U 09EC, 8870,3C ;9618           JSR [L0]                        ;Get address of MOV FQ TO FWR[FT1]
U 09ED, 3E1C,15 ;9619           MOV WR[0] TO LS[T14]
U 09EE, 8870,3C ;9620           JSR [L0]                        ;Get address of STORE FIRST FLOAT
U 09EF, 3E16,15 ;9621           MOV WR[0] TO LS[T11]
U 09F0, 8870,3C ;9622           JSR [L0]                        ;Get address of STORE SECOND FLOAT
U 09F1, BE18,15 ;9623           MOV WR[0] TO LS[T12]
                ;9624   TA.LOOPQ:
U 09F2, 8871,7C ;9625           JSR [GET.DATA]                  ;Get data to test WR.
U 09F3, 36A4,95 ;9626   TA.Q:   MOV LS[DATA.2] TO WR[1]         ;Put second float of data in WR 1
U 09F4, B610,15 ;9627           MOV LS[T8] TO WR[0]             ;Get address of load routine
U 09F5, 90F6,15 ;9628           MISC2 [TRAP.ACC] WR[0]          ;Force address of load routine
U 09F6, B6A2,15 ;9629           MOV LS[DATA.1] TO WR[0]         ;Wait for toggle load and set up
```

```
                ;9630                                   ; data to be passed to the FPA.
U 09F7, 10F4,15 ;9631       MISC2 [ASSERT.DA.AND.PASS.WR]   ;CPU syncs with FPA and passes WR 0
U 09F8, 3E14,95 ;9632       MOV WR[1] TO LS[T10]            ; and a longword of 0's to the FPA
U 09F9, B61A,15 ;9633       MOV LS[T13] TO WR[0]
U 09FA, B716,95 ;9634       MOV LS[#300] TO WR[1]           ;Get loop address
U 09FB, 90F6,15 ;9635       MISC2 [TRAP.ACC] WR[0]          ;Force move instruction
U 09FC, 10F6,95 ;9636       MISC2 [TRAP.ACC] WR[1]          ;Force loop instruction
U 09FD, B61C,15 ;9637       MOV LS[T14] TO WR[0]
U 09FE, 90F6,15 ;9638       MISC2 [TRAP.ACC] WR[0]          ;Force second MOV instruction
U 09FF, B616,15 ;9639       MOV LS[T11] TO WR[0]
U 0A00, B618,95 ;9640       MOV LS[T12] TO WR[1]
U 0A01, 90F6,15 ;9641       MISC2 [TRAP.ACC] WR[0]          ;Force word to store first float
U 0A02, BAA8,15 ;9642       MOV FPA TO LS[FIRST.FLT]
U 0A03, 10F6,95 ;9643       MISC2 [TRAP.ACC] WR[1]          ;Get second float
U 0A04, 3AAA,15 ;9644       MOV FPA TO LS[SEC.FLT]
U 0A05, B640,15 ;9645       MOV LS[BIT0] TO WR[0]           ;Set up other data
U 0A06, BE88,15 ;9646       MOV WR[0] TO LS[ADDRESS.DATA]
U 0A07, B65E,15 ;9647       MOV LS[BIT15] TO WR[0]          ;Setup error mask for first float
U 0A08, 3E8A,15 ;9648       MOV WR[0] TO LS[ERROR.MASK]
U 0A09, 36A2,95 ;9649       MOV LS[DATA.1] TO WR[1]         ;Set up expected data
U 0A0A, B6A8,15 ;9650       MOV LS[FIRST.FLT] TO WR[0]      ;Set up received data
U 0A0B, 0869,3C ;9651       JSR [CHECK.RESULT]
U 0A0C, 889F,34 ;9652       JMP [TA.Q]
U 0A0D, 3642,15 ;9653       MOV LS[BIT1] TO WR[0]           ;Set up other data
U 0A0E, BE88,15 ;9654       MOV WR[0] TO LS[ADDRESS.DATA]
U 0A0F, E58A,15 ;9655       CLR LS[ERROR.MASK]              ;Setup error mask for second float
U 0A10, 36A4,95 ;9656       MOV LS[DATA.2] TO WR[1]         ;Set up expected data
U 0A11, 36AA,15 ;9657       MOV LS[SEC.FLT] TO WR[0]        ;Set up received data
U 0A12, 0869,3C ;9658       JSR [CHECK.RESULT]              ;If an error report it
U 0A13, 889F,34 ;9659       JMP [TA.Q]                      ;Loop on error
U 0A14, 3714,95 ;9660       MOV LS[#6] TO WR[1]
                ;9661       CMP WR[2] WITH WR[1],           ;All data tested?
U 0A15, A644,B5 ;9662        DT(LONG)&SET.ALU.CC
U 0A16, 089F,21 ;9663       JMP.IF[NEQ] TO [TA.LOOPQ]       ;If not then continue looping
                ;9664                                       ;Else go on to next test
                ;9665   TA.END:
```

```
;9666    .page
;9667    .TOC    "TEST B EXPONENT DATA PATH TEST (M8389 FPA MODULE)"
;9668    ;++
;9669    ;
;9670    ;
;9671    ; Test Description:
;9672    ;
;9673    ;       The upper eight bits of the 16 bit exponent data path can't be address-
;9674    ;       ed by the CPU since there are no direct input lines. In order to check
;9675    ;       the Working Registers, data must be shifted in and data must be shifted
;9676    ;       out so that it can be moved back to the CPU where it can be tested. The
;9677    ;       data patterns that will be used to test all 16 Working registers are
;9678    ;       shift eight zeros left, then shift four right, and load the exponent
;9679    ;       data path back into the CPU to check.  Shift eight ones in the lower
;9680    ;       eight bits in the exponent data path left, then shift right, and check.
;9681    ;       These shifts can be done using the macro SHF LEFT EWR[] AND EQ IN[],
;9682    ;       with EQ IN[0] for zero and EQ IN[1] for one. To shift the pattern 0001
;9683    ;       from the lower four bits, that pattern must be loaded into the lower
;9684    ;       eight bits of the exponent data path then shifted. Using this same
;9685    ;       manner of testing, the exponent data path will be loaded with 1110 and
;9686    ;       shifted into the upper bits of the exponent data path. The last test
;9687    ;       that will be done will be to shift ones left 12 times, which should
;9688    ;       shift the ones past the eight upper bits of the exponent data path.
;9689    ;       Shifting right 8 times should result in 4 bits of zero's being shifted
;9690    ;       into the lower eight bits of the exponent data path because zero's are
;9691    ;       always shifted in from the right.
;9692    ;
;9693    ; Assumptions:
;9694    ;
;9695    ;       It is assumed that the Working register that can be loaded directly
;9696    ;       from the CPU has been tested.
;9697    ;
;9698    ; Test Description:
;9699    ;
;9700    ;       1) Load ET0 with the LSB of the exponent set. Shift left the contents
;9701    ;          of ET0. Read ET0 back to the CPU and check for error.
;9702    ;       2) Repeat step 1 until all of the lower 8 bit positions of ET0 can be
;9703    ;          checked.
;9704    ;       3) For the remaining steps each step will be performed for the patterns
;9705    ;          0, FFFFFFFF, 4000, and FFFFBFFF. Load the data pattern into ET0.
;9706    ;       4) Shift left ET0 four times, shift right ET0 four times, and store
;9707    ;          ET0 to the CPU and check for error.
;9708    ;       5) Load the data pattern into ET0, shift left eight times, shift right
;9709    ;          eight times, store ET0 to the CPU and check for error.
;9710    ;       6) Repeat steps 3 - 5 for ET6, ET2, and EQ.
;9711    ;
;9712    ; Error:
;9713    ;
;9714    ;       Error1 - Error in the shift mechanism of the 2901.
;9715    ;       Error2 - Failure in shifting left four times in ET0
;9716    ;       Error3 - Failure in shifting left eight times in ET0
;9717    ;       Error4 - Failure in shifting left four times in ET6
;9718    ;       Error5 - Failure in shifting left eight times in ET6
;9719    ;       Error6 - Failure in shifting left four times in ET2
;9720    ;       Error7 - Failure in shifting left eight times in ET2
```

```
                ;9721   ;       Error8 - Failure in shifting left four times in EQ
                ;9722   ;       Error9 - Failure in shifting left eight times in EQ
                ;9723   ; Debug:
                ;9724   ;
                ;9725   ;       Error1: If this error occurs then it should be noted what the expected
                ;9726   ;               and received data are. If the shift is failing in the first
                ;9727   ;               four bits then the least significant 2901 in the exponent data
                ;9728   ;               path is probably the cause of error. If the error is in the
                ;9729   ;               last four bits then the error is probably in the second 2901
                ;9730   ;               in the exponent data path. Since these registers have been
                ;9731   ;               tested previously it's likely that the error is with the
                ;9732   ;               shifting part of the 2901. If the 2901 is not the cause of
                ;9733   ;               error the inputs to the 2901 should be checked.
                ;9734   ;       Error2: If error one doesn't occur then the cause of the error is very
                ;9735   ;               likely the third 2901 in the exponent data path. If this is not
                ;9736   ;               not the case then the inputs to the 2901 should be checked
                ;9737   ;               during the right shift.
                ;9738   ;       Error3: Same as error 2 except that the most significant 2901 in the
                ;9739   ;               exponent data path is probably the cause of error.
                ;9740   ;       Error4: Same as error 2.
                ;9741   ;       Error5: Same as error 3.
                ;9742   ;       Error6: Same as error 2.
                ;9743   ;       Error7: Same as error 3.
                ;9744   ;       Error8: Same as error 2.
                ;9745   ;       Error9: Same as error 3.
                ;9746   ;
                ;9747   ;
                ;9748   ;--
                ;9749   T.B:
U 0A17, B65E,15 ;9750           MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;9751                                           ; STATUS WORD
U 0A18, 3E80,15 ;9752           MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;9753                                           ; BIT 15 INDICATES START OF TEST TO
                ;9754                                           ; CONSOLE
U 0A19, 10E0,15 ;9755           MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;9756   WAIT.TB.0:
U 0A1A, 08A1,A4 ;9757           JMP [WAIT.TB.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;9758                                           ; RESPOND
U 0A1B, 087E,6C ;9759           JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;9760                                           ; AND SIZE BITS
U 0A1C, 371A,15 ;9761           MOV LS[#FFFF807F] TO WR[0]      ;Set up error mask
U 0A1D, 3E8A,15 ;9762           MOV WR[0] TO LS[ERROR.MASK]
U 0A1E, 371C,15 ;9763           MOV LS[TB.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 0A1F, BE12,15 ;9764           MOV WR[0] TO LS[T9]             ; references.
U 0A20, 8870,3C ;9765           JSR [L0]                        ;Store addresses to force
U 0A21, BE14,15 ;9766           MOV WR[0] TO LS[T10]            ;Get the address of the Load routine
U 0A22, 8870,3C ;9767           JSR [L0]
U 0A23, 3E16,15 ;9768           MOV WR[0] TO LS[T11]            ;Get the address of SHIFT LEFT ETO
U 0A24, 8870,3C ;9769           JSR [L0]
U 0A25, BE18,15 ;9770           MOV WR[0] TO LS[T12]            ;Get the address of STORE ETO
U 0A26, 8870,3C ;9771           JSR [L0]
U 0A27, 3E10,15 ;9772           MOV WR[0] TO LS[T8]             ;Get the address of SIFHT RIGHT ETO
U 0A28, B64F,15 ;9773           MOV LS[#80] TO WR[2]
U 0A29, 3651,95 ;9774           MOV LS[#100] TO WR[3]
U 0A2A, 474F,95 ;9775   TB.LOOP:BIS LS[BIT7] TO WR[3]
```

```
U 0A2B, 3614,15 ;9776          MOV LS[T10] TO WR[0]              ;Load up WR 0 for forcing
U 0A2C, 90F6,15 ;9777          MISC2 [TRAP.ACC] WR[0]            ;Force the load routine
U 0A2D, 2004,15 ;9778          MOV WR[2] TO WR[0]                ;Set up data to pass
U 0A2E, 10F4,15 ;9779          MISC2 [ASSERT.DA.AND.PASS.WR]     ;Pass data
U 0A2F, B616,15 ;9780          MOV LS[T11] TO WR[0]              ;Load WR[0] for force
U 0A30, B618,95 ;9781          MOV LS[T12] TO WR[1]              ;Load WR[1] for force
U 0A31, 90F6,15 ;9782          MISC2 [TRAP.ACC] WR[0]            ;Force shift ET0
U 0A32, 10F6,95 ;9783          MISC2 [TRAP.ACC] WR[1]            ;Force store
U 0A33, BA1A,15 ;9784          MOV FPA TO LS[T13]                ;load the FPA data into t13
U 0A34, B61A,15 ;9785          MOV LS[T13] TO WR[0]              ;Set up received data
U 0A35, 2006,95 ;9786          MOV WR[3] TO WR[1]                ;Set up expected data
U 0A36, 0869,3C ;9787          JSR [CHECK.RESULT]                ;Report error if there is one
U 0A37, 08A2,A4 ;9788          JMP [TB.LOOP]                     ;Loop on error
U 0A38, A3D1,15 ;9789          SHL WR[2]                         ;Shift WR[2] left one for expected data
U 0A39, C54F,95 ;9790          BIC LS[BIT7] TO WR[3]
U 0A3A, 23D1,95 ;9791          SHL WR[3]
                ;9792          CMP WR[2] WITH LS[#8000],         ;Shifted through all 8 bits
U 0A3B, 4F5F,35 ;9793           DT(LONG)&SET.ALU.CC
U 0A3C, 08A2,A1 ;9794          JMP.IF[NEQ] TO [TB.LOOP]          ;If not then loop
U 0A3D, FF82,15 ;9795          INC LS[ERROR.NUMBER]              ;If yes then go on to next error
U 0A3E, 2F85,15 ;9796          CLR WR[2]                         ;clear counter
                ;9797    NEXT.PATTERN.ET0.4:
U 0A3F, 0874,6C ;9798          JSR [GET.PATTERN]
                ;9799    LOOP.ET0.4:
U 0A40, 8877,DC ;9800          JSR [LOAD.DATA]                   ;Load the data into the FPA
U 0A41, 8875,9C ;9801          JSR [SHIFT.LEFT.4]                ;shift the data left four times
U 0A42, 0876,BC ;9802          JSR [SHIFT.RIGHT.4]               ;shift the data right four times
U 0A43, 0878,6C ;9803          JSR [STORE.DATA.2]                ;store the data in LS[T13]
U 0A44, B61A,15 ;9804          MOV LS[T13] TO WR[0]              ;Set up received data
U 0A45, 36A2,95 ;9805          MOV LS[DATA.1] TO WR[1]           ;Set up expected data
U 0A46, 0869,3C ;9806          JSR [CHECK.RESULT]                ;Report error if there is one
U 0A47, 08A4,04 ;9807          JMP [LOOP.ET0.4]                  ;Loop on error
                ;9808          CMP WR[2] WITH LS[#4],
U 0A48, CF45,35 ;9809           DT(LONG)&SET.ALU.CC
U 0A49, 88A3,F1 ;9810          JMP.IF[NEQ] TO [NEXT.PATTERN.ET0.4]
                ;9811                                            ;If all patterns checked then
                ;9812                                            ;go on else continue looping
U 0A4A, FF82,15 ;9813          INC LS[ERROR.NUMBER]              ;If yes then go on to next error
U 0A4B, 2F85,15 ;9814          CLR WR[2]                         ;clear counter
                ;9815    NEXT.PATTERN.ET0.8:
U 0A4C, 0874,6C ;9816          JSR [GET.PATTERN]
                ;9817    LOOP.ET0.8:
U 0A4D, 8877,DC ;9818          JSR [LOAD.DATA]                   ;Load the data into the FPA
U 0A4E, 0876,2C ;9819          JSR [SHIFT.LEFT.8]                ;shift the data left four times
U 0A4F, 8877,4C ;9820          JSR [SHIFT.RIGHT.8]               ;shift the data right four times
U 0A50, 0878,6C ;9821          JSR [STORE.DATA.2]                ;store the data in LS[T13]
U 0A51, B61A,15 ;9822          MOV LS[T13] TO WR[0]              ;Set up received data
U 0A52, 36A2,95 ;9823          MOV LS[DATA.1] TO WR[1]           ;Set up expected data
U 0A53, 0869,3C ;9824          JSR [CHECK.RESULT]                ;Report error if there is one
U 0A54, 88A4,D4 ;9825          JMP [LOOP.ET0.8]                  ;Loop on error
                ;9826          CMP WR[2] WITH LS[#4],
U 0A55, CF45,35 ;9827           DT(LONG)&SET.ALU.CC
U 0A56, 08A4,C1 ;9828          JMP.IF[NEQ] TO [NEXT.PATTERN.ET0.8]
                ;9829                                            ;If all patterns checked then
                ;9830                                            ;go on else continue looping
```

```
U 0A57, 8870,3C ;9831   ET6:    JSR [LO]                          ;Shift left
U 0A58, 3E16,15 ;9832           MOV WR[0] TO LS[T11]
U 0A59, 8870,3C ;9833           JSR [LO]                          ;Shift right
U 0A5A, 3E10,15 ;9834           MOV WR[0] TO LS[T8]
U 0A5B, 8870,3C ;9835           JSR [LO]                          ;MOVE
U 0A5C, 3E1C,15 ;9836           MOV WR[0] TO LS[T14]
U 0A5D, 8870,3C ;9837           JSR [LO]                          ;STORE
U 0A5E, BE18,15 ;9838           MOV WR[0] TO LS[T12]
U 0A5F, FF82,15 ;9839           INC LS[ERROR.NUMBER]              ;If yes then go on to next error
U 0A60, 2F85,15 ;9840           CLR WR[2]                         ;clear counter
                ;9841   NEXT.PATTERN.ET2&ET6.4:
U 0A61, 0874,6C ;9842           JSR [GET.PATTERN]
                ;9843   LOOP.ET2&ET6.4:
U 0A62, 8877,DC ;9844           JSR [LOAD.DATA]                   ;Load the data into the FPA
U 0A63, B61C,15 ;9845           MOV LS[T14] TO WR[0]              ;MOV data to ET2/ET6
U 0A64, B716,95 ;9846           MOV LS[#300] TO WR[1]
U 0A65, 90F6,15 ;9847           MISC2 [TRAP.ACC] WR[0]
U 0A66, 10F6,95 ;9848           MISC2 [TRAP.ACC] WR[1]
U 0A67, 8875,9C ;9849           JSR [SHIFT.LEFT.4]                ;shift the data left four times
U 0A68, 0876,BC ;9850           JSR [SHIFT.RIGHT.4]               ;shift the data right four times
U 0A69, 0878,6C ;9851           JSR [STORE.DATA.2]                ;store the data in LS[T13]
U 0A6A, B61A,15 ;9852           MOV LS[T13] TO WR[0]              ;Set up received data
U 0A6B, 36A2,95 ;9853           MOV LS[DATA.1] TO WR[1]           ;Set up expected data
U 0A6C, 0869,3C ;9854           JSR [CHECK.RESULT]                ;Report error if there is one
U 0A6D, 08A6,24 ;9855           JMP [LOOP.ET2&ET6.4]              ;Loop on error
                ;9856           CMP WR[2] WITH LS[#4],
U 0A6E, CF45,35 ;9857            DT(LONG)&SET.ALU.CC
U 0A6F, 08A6,11 ;9858           JMP.IF[NEQ] TO [NEXT.PATTERN.ET2&ET6.4]
                ;9859                                             ;If all patterns checked then
                ;9860                                             ;go on else continue looping
U 0A70, FF82,15 ;9861           INC LS[ERROR.NUMBER]              ;If yes then go on to next error
U 0A71, 2F85,15 ;9862           CLR WR[2]                         ;clear counter
                ;9863   NEXT.PATTERN.ET2&ET6.8:
U 0A72, 0874,6C ;9864           JSR [GET.PATTERN]
                ;9865   LOOP.ET2&ET6.8:
U 0A73, 8877,DC ;9866           JSR [LOAD.DATA]                   ;Load the data into the FPA
U 0A74, B61C,15 ;9867           MOV LS[T14] TO WR[0]              ;MOV data to ET2&ET6
U 0A75, B716,95 ;9868           MOV LS[#300] TO WR[1]
U 0A76, 90F6,15 ;9869           MISC2 [TRAP.ACC] WR[0]
U 0A77, 10F6,95 ;9870           MISC2 [TRAP.ACC] WR[1]
U 0A78, 0876,2C ;9871           JSR [SHIFT.LEFT.8]                ;shift the data left four times
U 0A79, 8877,4C ;9872           JSR [SHIFT.RIGHT.8]               ;shift the data right four times
U 0A7A, 0878,6C ;9873           JSR [STORE.DATA.2]                ;store the data in LS[T13]
U 0A7B, B61A,15 ;9874           MOV LS[T13] TO WR[0]              ;Set up received data
U 0A7C, 36A2,95 ;9875           MOV LS[DATA.1] TO WR[1]           ;Set up expected data
U 0A7D, 0869,3C ;9876           JSR [CHECK.RESULT]                ;Report error if there is one
U 0A7E, 08A7,34 ;9877           JMP [LOOP.ET2&ET6.8]              ;Loop on error
                ;9878           CMP WR[2] WITH LS[#4],
U 0A7F, CF45,35 ;9879            DT(LONG)&SET.ALU.CC
U 0A80, 88A7,21 ;9880           JMP.IF[NEQ] TO [NEXT.PATTERN.ET2&ET6.8]
                ;9881                                             ;If all patterns checked then
                ;9882                                             ;go on else continue looping
U 0A81, 3720,15 ;9883           MOV LS[#7] TO WR[0]
                ;9884           CMP LS[ERROR.NUMBER] WITH WR[0],
U 0A82, CE82,35 ;9885            DT(LONG)&SET.ALU.CC
```

```
U 0A83, 08A5,71 ;9886            JMP.IF[NEQ] TO [ET6]
U 0A84, 8870,3C ;9887            JSR [LO]                       ;Shift left
U 0A85, 3E16,15 ;9888            MOV WR[0] TO LS[T11]
U 0A86, 8870,3C ;9889            JSR [LO]                       ;Shift right
U 0A87, 3E10,15 ;9890            MOV WR[0] TO LS[T8]
U 0A88, 8870,3C ;9891            JSR [LO]                       ;MOVE
U 0A89, 3E1C,15 ;9892            MOV WR[0] TO LS[T14]
U 0A8A, 8870,3C ;9893            JSR [LO]                       ;STORE
U 0A8B, BE18,15 ;9894            MOV WR[0] TO LS[T12]
U 0A8C, 8870,3C ;9895            JSR [LO]                       ;MOVE
U 0A8D, BE1E,15 ;9896            MOV WR[0] TO LS[T15]
U 0A8E, FF82,15 ;9897            INC LS[ERROR.NUMBER]           ;If yes then go on to next error
U 0A8F, 2F85,15 ;9898            CLR WR[2]                      ;clear counter
                ;9899    NEXT.PATTERN.EQ.4:
U 0A90, 0874,6C ;9900            JSR [GET.PATTERN]
                ;9901    LOOP.EQ.4:
U 0A91, 8877,DC ;9902            JSR [LOAD.DATA]                ;Load the data into the FPA
U 0A92, B61C,15 ;9903            MOV LS[T14] TO WR[0]           ;MOV data to EQ
U 0A93, B716,95 ;9904            MOV LS[#300] TO WR[1]
U 0A94, 90F6,15 ;9905            MISC2 [TRAP.ACC] WR[0]
U 0A95, 10F6,95 ;9906            MISC2 [TRAP.ACC] WR[1]
U 0A96, 8875,9C ;9907            JSR [SHIFT.LEFT.4]             ;shift the data left four times
U 0A97, 0876,BC ;9908            JSR [SHIFT.RIGHT.4]            ;shift the data right four times
U 0A98, 361E,15 ;9909            MOV LS[T15] TO WR[0]           ;MOV data from EQ to ET1
U 0A99, B716,95 ;9910            MOV LS[#300] TO WR[1]
U 0A9A, 90F6,15 ;9911            MISC2 [TRAP.ACC] WR[0]
U 0A9B, 10F6,95 ;9912            MISC2 [TRAP.ACC] WR[1]
U 0A9C, 0878,6C ;9913            JSR [STORE.DATA.2]             ;store the data in LS[T13]
U 0A9D, B61A,15 ;9914            MOV LS[T13] TO WR[0]           ;Set up received data
U 0A9E, 36A2,95 ;9915            MOV LS[DATA.1] TO WR[1]        ;Set up expected data
U 0A9F, 0869,3C ;9916            JSR [CHECK.RESULT]             ;Report error if there is one
U 0AA0, 08A9,14 ;9917            JMP [LOOP.EQ.4]                ;Loop on error
                ;9918            CMP WR[2] WITH LS[#4],
U 0AA1, CF45,35 ;9919             DT(LONG)&SET.ALU.CC
U 0AA2, 88A9,01 ;9920            JMP.IF[NEQ] TO [NEXT.PATTERN.EQ.4]
                ;9921                                           ;If all patterns checked then
                ;9922                                           ;go on else continue looping
U 0AA3, FF82,15 ;9923            INC LS[ERROR.NUMBER]           ;If yes then go on to next error
U 0AA4, 2F85,15 ;9924            CLR WR[2]                      ;clear counter
                ;9925    NEXT.PATTERN.EQ.8:
U 0AA5, 0874,6C ;9926            JSR [GET.PATTERN]
                ;9927    LOOP.EQ.8:
U 0AA6, 8877,DC ;9928            JSR [LOAD.DATA]                ;Load the data into the FPA
U 0AA7, B61C,15 ;9929            MOV LS[T14] TO WR[0]           ;MOV data to EQ
U 0AA8, B716,95 ;9930            MOV LS[#300] TO WR[1]
U 0AA9, 90F6,15 ;9931            MISC2 [TRAP.ACC] WR[0]
U 0AAA, 10F6,95 ;9932            MISC2 [TRAP.ACC] WR[1]
U 0AAB, 0876,2C ;9933            JSR [SHIFT.LEFT.8]             ;shift the data left four times
U 0AAC, 8877,4C ;9934            JSR [SHIFT.RIGHT.8]            ;shift the data right four times
U 0AAD, 361E,15 ;9935            MOV LS[T15] TO WR[0]           ;MOV data from EQ to ET1
U 0AAE, B716,95 ;9936            MOV LS[#300] TO WR[1]
U 0AAF, 90F6,15 ;9937            MISC2 [TRAP.ACC] WR[0]
U 0AB0, 10F6,95 ;9938            MISC2 [TRAP.ACC] WR[1]
U 0AB1, 0878,6C ;9939            JSR [STORE.DATA.2]             ;store the data in LS[T13]
U 0AB2, B61A,15 ;9940            MOV LS[T13] TO WR[0]           ;Set up received data
```

```
U 0AB3, 36A2,95 ;9941          MOV LS[DATA.1] TO WR[1]          ;Set up expected data
U 0AB4, 0869,3C ;9942          JSR [CHECK.RESULT]               ;Report error if there is one
U 0AB5, 88AA,64 ;9943          JMP [LOOP.EQ.8]                  ;Loop on error
                ;9944          CMP WR[2] WITH LS[#4],
U 0AB6, CF45,35 ;9945           DT(LONG)&SET.ALU.CC
U 0AB7, 88AA,51 ;9946          JMP.IF[NEQ] TO [NEXT.PATTERN.EQ.8]
                ;9947                                           ;If all patterns checked then
                ;9948                                           ;go on else continue looping
                ;9949  TB.END:
```

```
;9950    .page
;9951    .TOC    "TEST C EVEN EXTENSION WORKING REGISTER TEST(M8389 FPA MODULE)"
;9952    ;++
;9953    ;
;9954    ;
;9955    ; Test Description:
;9956    ;
;9957    ;       The purpose of this test is to check the even working registers in
;9958    ;       the extension part of the fraction data path. The even part of the
;9959    ;       extension data path can only be written to on a third huge load
;9960    ;       and the odd part of the extension data path can't be written to
;9961    ;       directly at all and will be tested in a later test. The even part
;9962    ;       of the extension data path will be tested by loading data into WR
;9963    ;       and reading it back to the CPU and checking it. Since there is a
;9964    ;       shortage of ROM code only one register will be loaded and verified.
;9965    ;       Once this register is verified other registers will be tested by
;9966    ;       moving data from the load register to the register to be tested
;9967    ;       and storing the register that's being tested. In this way only one
;9968    ;       load routine is required instead of eight.
;9969    ;
;9970    ; Assumptions:
;9971    ;
;9972    ;       It is assumed that the all WR's in the FPA have been verified except
;9973    ;       for the registers in the extension data path.
;9974    ;
;9975    ; Test Steps:
;9976    ;
;9977    ;       1) Get the data that 's going to be loaded into the register to be
;9978    ;          tested.
;9979    ;       2) Load the data from the CPU into the FPA
;9980    ;       3) MOV the data into the register to be tested if it's not already
;9981    ;          there.
;9982    ;       4) Read the data back from the register under test to the CPU and
;9983    ;          compare it with the data you got in step 1 and if they are not the
;9984    ;          same then issue and error.
;9985    ;
;9986    ; Error:
;9987    ;
;9988    ;       Error1 - ET0 failed
;9989    ;       Error2 - ET2 failed
;9990    ;       Error3 - ET4 failed
;9991    ;       Error4 - ET6 failed
;9992    ;       Error5 - ET8 failed
;9993    ;       Error6 - ETA failed
;9994    ;       Error7 - ETC failed
;9995    ;       Error8 - ETE failed
;9996    ;       Error9 - EQ failed
;9997    ;
;9998    ; Debug:
;9999    ;
;10000   ;       Error1: If this error occurs then one of the bit or bits in one of the
;10001   ;               2901's in the extension data path is probably failing. If the
;10002   ;               expected and received data are different in the 5th place then
;10003   ;               the error is with the most significant 2901. If the expected
;10004   ;               and received data are different in the 6th place then the least
```

```
                ;10005  ;               significant 2901 is in error.
                ;10006  ;       Error2: Same as error 1 except that ET2 is in error.
                ;10007  ;       Error3: Same as error 1 except that ET4 is in error.
                ;10008  ;       Error4: Same as error 1 except that ET6 is in error.
                ;10009  ;       Error5: Same as error 1 except that ET8 is in error.
                ;10010  ;       Error6: Same as error 1 except that ETA is in error.
                ;10011  ;       Error7: Same as error 1 except that ETC is in error.
                ;10012  ;       Error8: Same as error 1 except that ETE is in error.
                ;10013  ;       Error9: Same as error 1 except that ETQ is in error.
                ;10014  ;
                ;10015  ;
                ;10016  ;--
                ;10017  T.C:
U 0AB8, B65E,15 ;10018          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;10019                                          ; STATUS WORD
U 0AB9, 3E80,15 ;10020          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;10021                                          ; BIT 15 INDICATES START OF TEST TO
                ;10022                                          ; CONSOLE
U 0ABA, 10E0,15 ;10023          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;10024  WAIT.TC.0:
U 0ABB, 88AB,B4 ;10025          JMP [WAIT.TC.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;10026                                          ; RESPOND
U 0ABC, 087E,6C ;10027          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;10028                                          ; AND SIZE BITS
U 0ABD, B748,15 ;10029          MOV LS[#70300] TO WR[0]         ;Set size bits to huge
U 0ABE, 90F6,15 ;10030          MISC2 [TRAP.ACC] WR[0]
U 0ABF, B722,15 ;10031          MOV LS[TC.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 0AC0, BE12,15 ;10032          MOV WR[0] TO LS[T9]              ; references.
U 0AC1, 8870,3C ;10033          JSR [LO]
U 0AC2, BE14,15 ;10034          MOV WR[0] TO LS[T10]             ;LOAD
U 0AC3, 8870,3C ;10035          JSR [LO]
U 0AC4, BE18,15 ;10036          MOV WR[0] TO LS[T12]             ;STORE
U 0AC5, E5F8,15 ;10037          CLR LS[OS]
U 0AC6, 7CF8,15 ;10038          DEC LS[OS]
                ;10039  NEXT.PATTERN.EXT:
U 0AC7, 08B0,1C ;10040          JSR [TEST.DATA]                 ;Get data to test
                ;10041  LOOP.EXT:
U 0AC8, 8877,DC ;10042          JSR [LOAD.DATA]                 ;Load the data into the FPA
U 0AC9, 0878,6C ;10043          JSR [STORE.DATA.2]              ;Read the data back to the CPU
U 0ACA, B61A,15 ;10044          MOV LS[T13] TO WR[0]            ;Set up received data
U 0ACB, 36A2,95 ;10045          MOV LS[DATA.1] TO WR[1]         ;Set up expected data
U 0ACC, 0869,3C ;10046          JSR [CHECK.RESULT]              ;Report an error in there is one
U 0ACD, 08AC,84 ;10047          JMP [LOOP.EXT]                  ;Loop on error
U 0ACE, 374C,15 ;10048          MOV LS[#BFFFFFFF] TO WR[0]      ;Setup to see if all patterns checked
                ;10049          CMP WR[2] WITH WR[0],           ;Are all patterns checked?
U 0ACF, 2644,35 ;10050           DT(LONG)&SET.ALU.CC
U 0AD0, 08AC,71 ;10051          JMP.IF[NEQ] TO [NEXT.PATTERN.EXT]
                ;10052                                          ;If not then get next pattern
                ;10053                                          ; else continue
U 0AD1, B724,15 ;10054  REST:   MOV LS[#FFFF00FF] TO WR[0]      ;Reset error mask
U 0AD2, 3E8A,15 ;10055          MOV WR[0] TO LS[ERROR.MASK]
U 0AD3, FF82,15 ;10056          INC LS[ERROR.NUMBER]            ;Go onto error 2, 3, 4, 5, 6, 7, 8
U 0AD4, 8870,3C ;10057          JSR [LO]
U 0AD5, BE18,15 ;10058          MOV WR[0] TO LS[T12]             ;STORE
U 0AD6, 8870,3C ;10059          JSR [LO]
```

```
U 0AD7, 3E1C,15 ;10060          MOV WR[0] TO LS[T14]             ;MOV
U 0AD8, 2F85,15 ;10061          CLR WR[2]
U 0AD9, 2105,15 ;10062          DEC WR[2]
                ;10063  NEXT.PATTERN.EXT.REST:
U 0ADA, 8871,7C ;10064          JSR [GET.DATA]                   ;Get data to test
                ;10065  LOOP.EXT.REST:
U 0ADB, 8877,DC ;10066          JSR [LOAD.DATA]                  ;Load the data into the FPA
U 0ADC, 0878,AC ;10067          JSR [MOV.DATA]                   ;MOV the data to the register to be
                ;10068                                           ; tested
U 0ADD, 0878,6C ;10069          JSR [STORE.DATA.2]               ;Read the data back to the CPU
U 0ADE, B61A,15 ;10070          MOV LS[T13] TO WR[0]             ;Set up received data
U 0ADF, 36A2,95 ;10071          MOV LS[DATA.1] TO WR[1]          ;Set up expected data
U 0AE0, 0869,3C ;10072          JSR [CHECK.RESULT]               ;Report an error in there is one
U 0AE1, 88AD,B4 ;10073          JMP [LOOP.EXT.REST]              ;Loop on error
U 0AE2, 36C0,15 ;10074          MOV LS[#3(H)] TO WR[0]
                ;10075          CMP WR[2] WITH WR[0],
U 0AE3, 2644,35 ;10076           DT(LONG)&SET.ALU.CC
U 0AE4, 08AD,A1 ;10077          JMP.IF[NEQ] TO [NEXT.PATTERN.EXT.REST]
                ;10078                                           ;If all patterns not tested then loop
U 0AE5, 3682,15 ;10079          MOV LS[ERROR.NUMBER] TO WR[0]
                ;10080          CMP WR[0] WITH LS[#8],
U 0AE6, CF46,35 ;10081           DT(LONG)&SET.ALU.CC
U 0AE7, 88AD,11 ;10082          JMP.IF[NEQ] TO [REST]            ;If all registers not test then loop
U 0AE8, FF82,15 ;10083          INC LS[ERROR.NUMBER]             ;Go on to error 9
U 0AE9, 8870,3C ;10084          JSR [L0]
U 0AEA, BE18,15 ;10085          MOV WR[0] TO LS[T12]             ;STORE
U 0AEB, 8870,3C ;10086          JSR [L0]
U 0AEC, 3E1C,15 ;10087          MOV WR[0] TO LS[T14]             ;MOV
U 0AED, 8870,3C ;10088          JSR [L0]
U 0AEE, BE1E,15 ;10089          MOV WR[0] TO LS[T15]             ;MOV from Q
U 0AEF, 2F85,15 ;10090          CLR WR[2]
U 0AF0, 2105,15 ;10091          DEC WR[2]
                ;10092  NEXT.PATTERN.EXT.Q:
U 0AF1, 8871,7C ;10093          JSR [GET.DATA]                   ;Get data to test
                ;10094  LOOP.EXT.Q:
U 0AF2, 8877,DC ;10095          JSR [LOAD.DATA]                  ;Load the data into the FPA
U 0AF3, 0878,AC ;10096          JSR [MOV.DATA]                   ;MOV the data to the register to be
                ;10097                                           ; tested
U 0AF4, 361E,15 ;10098          MOV LS[T15] TO WR[0]             ;move from Q to 2
U 0AF5, B716,95 ;10099          MOV LS[#300] TO WR[1]
U 0AF6, 90F6,15 ;10100          MISC2 [TRAP.ACC] WR[0]
U 0AF7, 10F6,95 ;10101          MISC2 [TRAP.ACC] WR[1]
U 0AF8, 0878,6C ;10102          JSR [STORE.DATA.2]               ;Read the data back to the CPU
U 0AF9, B61A,15 ;10103          MOV LS[T13] TO WR[0]             ;Set up received data
U 0AFA, 36A2,95 ;10104          MOV LS[DATA.1] TO WR[1]          ;Set up expected data
U 0AFB, 0869,3C ;10105          JSR [CHECK.RESULT]               ;Report an error in there is one
U 0AFC, 08AF,24 ;10106          JMP [LOOP.EXT.Q]                 ;Loop on error
U 0AFD, 36C0,15 ;10107          MOV LS[#3(H)] TO WR[0]
                ;10108          CMP WR[2] WITH WR[0],
U 0AFE, 2644,35 ;10109           DT(LONG)&SET.ALU.CC
U 0AFF, 08AF,11 ;10110          JMP.IF[NEQ] TO [NEXT.PATTERN.EXT.Q]
U 0B00, 08B0,F4 ;10111          JMP [TC.END]
                ;10112
                ;10113  ; SHIFTING 32 BIT PATTERN OF ONES THROUGH ZEROS AND ZEROS THROUGH ONES
                ;10114  ;
```

```
                ;10115  ;       The subrouitne generates a shifting pattern of one through zeros
                ;10116  ;       and zero through ones for 32 bits
                ;10117
                ;10118  TEST.DATA:
                ;10119          CMP WR[2] WITH LS[BIT31],       ;Has the bit been shifted thru 32 bits?
U 0B01, CF7F,35 ;10120           DT(LONG)&SET.ALU.CC
U 0B02, 08B0,89 ;10121          JMP.IF[EQL] TO [HALF]           ;If it's starting the 2nd 1/2 go to half
U 0B03, 88B0,A3 ;10122          JMP.IF[GEQU] TO [GTR.HALF]      ;If it's the 2nd 1/2 go to gtr.half
U 0B04, 7FF8,15 ;10123          INC LS[OS]                      ; else add one to OS pointer
U 0B05, B6F7,15 ;10124          MOV LS[SHIFT.OS(4-0)] TO WR[2]  ;Move the data pointed to by OS to WR 2
U 0B06, BEA3,15 ;10125          MOV WR[2] TO LS[DATA.1]         ;Move WR 2 to DATA.1
U 0B07, 5B00,14 ;10126          RETURN                          ;Return to calling routine
U 0B08, E5F8,15 ;10127  HALF:   CLR LS[OS]                      ;Clear the pointrer
U 0B09, 7CF8,15 ;10128          DEC LS[OS]                      ;Decrement pointer
                ;10129  GTR.HALF:
U 0B0A, 7FF8,15 ;10130          INC LS[OS]                      ;Add one to pointer
U 0B0B, B6F7,15 ;10131          MOV LS[SHIFT.OS(4-0)] TO WR[2]  ;Move the data pointed to by OS to WR 2
U 0B0C, A0C5,15 ;10132          COM WR[2]                       ;Complement WR 2
U 0B0D, BEA3,15 ;10133          MOV WR[2] TO LS[DATA.1]         ;Move WR 2 to DATA.1
U 0B0E, 5B00,14 ;10134          RETURN                          ;Return to calling routine
                ;10135
                ;10136  TC.END:
```

```
;10137  .page
;10138  .TOC    "TEST D ODD EXTENSION WORKING REGISTER TEST (M8389 FPA MODULE)"
;10139  ;++
;10140  ;
;10141  ;
;10142  ; Test Description:
;10143  ;
;10144  ;       The odd working registers of the extension data path can't be loaded
;10145  ;       from the CPU so they will be tested by loading an even working reg,
;10146  ;       moving to an odd working reg, moving to the Q reg and then back to
;10147  ;       an even working reg. The move through the Q reg is to save fpa micro-
;10148  ;       code. in this manner we can test all of the odd extension working
;10149  ;       registers.
;10150  ;
;10151  ; Assumptions:
;10152  ;
;10153  ;       It is assumed that the Working registers of all the other 2901's have
;10154  ;       been tested and work, the ones not having been tested are the Working
;10155  ;       registers being tested here the extension data path Working registers.
;10156  ;
;10157  ; Test Steps:
;10158  ;
;10159  ;       1) For all the steps in test the patterns in GET DATA will be used.
;10160  ;          Load FT0 with the data pattern.
;10161  ;       2) Move FWR[FT0] TO FWR[FT1].
;10162  ;       3) Move FWR[FT1] TO FQ.
;10163  ;       4) Move FQ TO FWR[FT2].
;10164  ;       5) Check for error.
;10165  ;       6) Repeat steps 1-5 for FWR[FT3], FWR[FT5], FWR[FT7], FWR[FT9], FWR[FTB]
;10166  ;          FWR[FTD], and FWR[FTF].
;10167  ;
;10168  ; Error:
;10169  ;
;10170  ;       Error1 - FWR[FT1] failed.
;10171  ;       Error2 - FWR[FT3] failed.
;10172  ;       Error3 - FWR[FT5] failed.
;10173  ;       Error4 - FWR[FT7] failed.
;10174  ;       Error5 - FWR[FT9] failed.
;10175  ;       Error6 - FWR[FTB] failed.
;10176  ;       Error7 - FWR[FTD] failed.
;10177  ;       Error8 - FWR[FTF] failed.
;10178  ;
;10179  ; Debug:
;10180  ;
;10181  ;       Error1: There are four likely causes of this error, they are the
;10182  ;               control lines to the 2901, a short in the etch between the
;10183  ;               2901's, the 2901 itself or the failure of FPAD EXT B ADDR0.
;10184  ;               If FPAD EXT B ADDR0 is not high then the latch it came from
;10185  ;               should be checked for error. FPAD EXT B ADDR0 should be checked
;10186  ;               during either the load of the data or the store. There are a
;10187  ;               variety of instructions used in testing the extension registers
;10188  ;               Depending on the instruction the control lines will vary. In
;10189  ;               the case where the control lines are suspect the 2901 Control
;10190  ;               Decoding table under tables should be checked.
;10191  ;       Error2: Same as Error1 except that since the control lines have been
```

```
                ;10192  ;               tested already it's likely the error is in the 2901 or the etch
                ;10193  ;       Error3: Same as error2.
                ;10194  ;       Error4: Same as error2.
                ;10195  ;       Error5: Same as error2.
                ;10196  ;       Error6: Same as error2.
                ;10197  ;       Error7: Same as error2.
                ;10198  ;       Error8: Same as error2.
                ;10199  ;
                ;10200  ;
                ;10201  ;--
                ;10202  T.D:
U 0B0F, B65E,15 ;10203          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;10204                                          ; STATUS WORD
U 0B10, 3E80,15 ;10205          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;10206                                          ; BIT 15 INDICATES START OF TEST TO
                ;10207                                          ; CONSOLE
U 0B11, 10E0,15 ;10208          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;10209  WAIT.TD.0:
U 0B12, 08B1,24 ;10210          JMP [WAIT.TD.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                ;10211                                          ; RESPOND
U 0B13, 087E,6C ;10212          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;10213                                          ; AND SIZE BITS
U 0B14, B724,15 ;10214          MOV LS[#FFFF00FF] TO WR[0]      ;Set up error mask
U 0B15, 3E8A,15 ;10215          MOV WR[0] TO LS[ERROR.MASK]
U 0B16, 3726,15 ;10216          MOV LS[TD.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 0B17, BE12,15 ;10217          MOV WR[0] TO LS[T9]             ; references.
U 0B18, 8870,3C ;10218          JSR [LO]
U 0B19, BE14,15 ;10219          MOV WR[0] TO LS[T10]            ;Load FTO
U 0B1A, 8870,3C ;10220          JSR [LO]
U 0B1B, 3E16,15 ;10221          MOV WR[0] TO LS[T11]            ;MOV FQ TO FT2
U 0B1C, 8870,3C ;10222          JSR [LO]
U 0B1D, BE18,15 ;10223          MOV WR[0] TO LS[T12]            ;STORE FT2
U 0B1E, 8870,3C ;10224  EVEN:   JSR [LO]
U 0B1F, 3E1C,15 ;10225          MOV WR[0] TO LS[T14]            ;MOV FTO TO FTX
U 0B20, 8870,3C ;10226          JSR [LO]
U 0B21, BE1E,15 ;10227          MOV WR[0] TO LS[T15]            ;MOV FTX TO FQ
U 0B22, 2F85,15 ;10228          CLR WR[2]
U 0B23, 2105,15 ;10229          DEC WR[2]
                ;10230  NEXT.PATTERN.EXT.EVEN:
U 0B24, 8871,7C ;10231          JSR [GET.DATA]                  ;Get data to test
                ;10232  LOOP.EXT.EVEN:
U 0B25, 8877,DC ;10233          JSR [LOAD.DATA]                 ;Load the data into the FPA
U 0B26, 0878,AC ;10234          JSR [MOV.DATA]                  ;MOV the data to the register to be
                ;10235                                          ; tested
U 0B27, 361E,15 ;10236          MOV LS[T15] TO WR[0]            ;move FTX to FQ
U 0B28, B716,95 ;10237          MOV LS[#300] TO WR[1]
U 0B29, 90F6,15 ;10238          MISC2 [TRAP.ACC] WR[0]
U 0B2A, 10F6,95 ;10239          MISC2 [TRAP.ACC] WR[1]
U 0B2B, B616,15 ;10240          MOV LS[T11] TO WR[0]            ;move from Q to 2
U 0B2C, B716,95 ;10241          MOV LS[#300] TO WR[1]
U 0B2D, 90F6,15 ;10242          MISC2 [TRAP.ACC] WR[0]
U 0B2E, 10F6,95 ;10243          MISC2 [TRAP.ACC] WR[1]
U 0B2F, 0878,6C ;10244          JSR [STORE.DATA.2]              ;Read the data back to the CPU
U 0B30, B51A,15 ;10245          MOV LS[T13] TO WR[0]            ;Set up received data
U 0B31, 36A2,95 ;10246          MOV LS[DATA.1] TO WR[1]         ;Set up expected data
```

```
U 0B32, 0869,3C ;10247          JSR [CHECK.RESULT]              ;Report an error in there is one
U 0B33, 88B2,54 ;10248          JMP [LOOP.EXT.EVEN]             ;Loop on error
                ;10249          CMP WR[2] WITH LS[#3(H)],
U 0B34, CFC1,35 ;10250           DT(LONG)&SET.ALU.CC
U 0B35, 08B2,41 ;10251          JMP.IF[NEQ] TO [NEXT.PATTERN.EXT.EVEN]
                ;10252                                          ;If all patterns not tested then loop
U 0B36, FF82,15 ;10253          INC LS[ERROR.NUMBER]
U 0B37, B728,15 ;10254          MOV LS[#9] TO WR[0]
                ;10255          CMP WR[0] WITH LS[ERROR.NUMBER],
U 0B38, 4F82,35 ;10256           DT(LONG)&SET.ALU.CC
U 0B39, 08B1,E1 ;10257          JMP.IF[NEQ] TO [EVEN]           ;If all registers not test then loop
                ;10258  TD.END:
```

```
;10259  .page
;10260  .TOC    "TEST E REMAINING EXPONENT REGISTER TEST(M8389 FPA MODULE)"
;10261  ;++
;10262  ;
;10263  ;
;10264  ; Test Description:
;10265  ;
;10266  ;       The purpose of this test is to check the upper bits of the exponent
;10267  ;       working registers that couldn't be shifted in and out of directly.
;10268  ;       In order to check these registers, they are EWR[ET1], EWR[ET3],EWR[ET4]
;10269  ;       EWR[ET5], EWR[ET7], EWR[ET8], EWR[ET9], EWR[ETA], EWR[ETB], EWR[ETC],
;10270  ;       EWR[ETD], EWR[ETE], and EWR[ETF] the data pattern will have to be
;10271  ;       shifted into EWR[ET0], moved to the register that's being tested, moved
;10272  ;       to EWR[ET6], shifted right into the bits that can be stored to the
;10273  ;       CPU, and stored to the CPU. This will be done for all the registers
;10274  ;       mentioned and for all the patterns in GET PATTERN.
;10275  ;
;10276  ; Assumptions:
;10277  ;
;10278  ;       It is assumed that the exponent data path test has been run success-
;10279  ;       fully.
;10280  ;
;10281  ; Test Steps:
;10282  ;
;10283  ;       1) Get the data pattern and load the data pattern into EWR[ET0].
;10284  ;       2) Shift EWR[ET0] left eight times to put the data in the upper
;10285  ;          bits.
;10286  ;       3) Move EWR[ET0] to EWR[ET1].
;10287  ;       4) Move EWR[ET1] to EWR[ET6].
;10288  ;       5) Shift EWR[ET6] right eight times to put the data in the lower
;10289  ;          exponent bits.
;10290  ;       6) Store EWR[ET6] to the CPU and check for error.
;10291  ;       7) Repeat steps 1 - 6 for EWR[ET3], EWR[ET4], EWR[ET5], EWR[ET7],
;10292  ;          EWR[ET8], EWR[ET9], EWR[ETA], EWR[ETB], EWR[ETC], EWR[ETD],EWR[ETE],
;10293  ;          and EWR[ETF].
;10294  ;
;10295  ; Error:
;10296  ;
;10297  ;       Error1 - EWR[ET1] failed in the first four bits.
;10298  ;       Error2 - EWR[ET1] failed in the last four bits.
;10299  ;       Error3 - EWR[ET3] failed in the first four bits.
;10300  ;       Error4 - EWR[ET3] failed in the last four bits.
;10301  ;       Error5 - EWR[ET4] failed in the first four bits.
;10302  ;       Error6 - EWR[ET4] failed in the last four bits.
;10303  ;       Error7 - EWR[ET5] failed in the first four bits.
;10304  ;       Error8 - EWR[ET5] failed in the last four bits.
;10305  ;       Error9 - EWR[ET7] failed in the first four bits.
;10306  ;       ErrorA - EWR[ET7] failed in the last four bits.
;10307  ;       ErrorB - EWR[ET8] failed in the first four bits.
;10308  ;       ErrorC - EWR[ET8] failed in the last four bits.
;10309  ;       ErrorD - EWR[ET9] failed in the first four bits.
;10310  ;       ErrorE - EWR[ET9] failed in the last four bits.
;10311  ;       ErrorF - EWR[ETA] failed in the first four bits.
;10312  ;       Error10 - EWR[ETA] failed in the last four bits.
;10313  ;       Error11 - EWR[ETB] failed in the first four bits.
```

```
                ;10314  ;       Error12 - EWR[ET8] failed in the last four bits.
                ;10315  ;       Error13 - EWR[ET9] failed in the first four bits.
                ;10316  ;       Error14 - EWR[ET9] failed in the last four bits.
                ;10317  ;       Error15 - EWR[ETD] failed in the first four bits.
                ;10318  ;       Error16 - EWR[ETD] failed in the last four bits.
                ;10319  ;       Error17 - EWR[ETE] failed in the first four bits.
                ;10320  ;       Error18 - EWR[ETE] failed in the last four bits.
                ;10321  ;       Error19 - EWR[ETF] failed in the first four bits.
                ;10322  ;       Error1A - EWR[ETF] failed in the last four bits.
                ;10323  ;
                ;10324  ; Debug:
                ;10325  ;
                ;10326  ;       Error1: If this error occurs then it's very likely that the third
                ;10327  ;               2901 in the exponent data path is in error. Since EWR[ET0]
                ;10328  ;               and EWR[ET6] have been checked in an earlier test the
                ;10329  ;               only other cause of error could be the control bits to the
                ;10330  ;               2901 durring the move to and from the register being tested.
                ;10331  ;               Therefore if the 2901 is not the cause of error then the
                ;10332  ;               control line inputs to the 2901 should be checked during the
                ;10333  ;               moves to and from the register.
                ;10334  ;       Error2: Same as error one except the failing part is likely to be the
                ;10335  ;               fourth 2901 in the exponent data path.
                ;10336  ;       Error3: Same as error 1.
                ;10337  ;       Error4: Same as error 2.
                ;10338  ;       Error5: Same as error 1.
                ;10339  ;       Error6: Same as error 2.
                ;10340  ;       Error7: Same as error 1.
                ;10341  ;       Error8: Same as error 2.
                ;10342  ;       Error9: Same as error 1.
                ;10343  ;       ErrorA: Same as error 2.
                ;10344  ;       ErrorB: Same as error 1.
                ;10345  ;       ErrorC: Same as error 2.
                ;10346  ;       ErrorD: Same as error 1.
                ;10347  ;       ErrorE: Same as error 2.
                ;10348  ;       ErrorF: Same as error 1.
                ;10349  ;       Error10: Same as error 2.
                ;10350  ;       Error11: Same as error 1.
                ;10351  ;       Error12: Same as error 2.
                ;10352  ;       Error13: Same as error 1.
                ;10353  ;       Error14: Same as error 2.
                ;10354  ;       Error15: Same as error 1.
                ;10355  ;       Error16: Same as error 2.
                ;10356  ;       Error17: Same as error 1.
                ;10357  ;       Error18: Same as error 2.
                ;10358  ;       Error19: Same as error 1.
                ;10359  ;       Error1A: Same as error 2.
                ;10360  ;
                ;10361  ;
                ;10362  ;--
                ;10363  T.E:
U 0B3A, B65E,15 ;10364          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;10365                                          ; STATUS WORD
U 0B3B, 3E80,15 ;10366          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;10367                                          ; BIT 15 INDICATES START OF TEST TO
                ;10368                                          ; CONSOLE
```

```
U 0B3C, 10E0,15 ;10369          MISC [SET.CP.ATTN]                  ;ISSUE CPU ATTN TO CONSOLE
                ;10370  WAIT.TE.0:
U 0B3D, 88B3,D4 ;10371          JMP [WAIT.TE.0]                     ;LOOP HERE WAITING FOR CONSOLE TO
                ;10372                                              ; RESPOND
U 0B3E, 087E,6C ;10373          JSR [SETUP]                         ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;10374                                              ; AND SIZE BITS
U 0B3F, 371A,15 ;10375          MOV LS[#FFFF807F] TO WR[0]          ;Set up error mask
U 0B40, 3E8A,15 ;10376          MOV WR[0] TO LS[ERROR.MASK]
U 0B41, 372A,15 ;10377          MOV LS[TE.POINTER] TO WR[0]         ;Initialize pointer for main memory
U 0B42, BE12,15 ;10378          MOV WR[0] TO LS[T9]                 ; references.
U 0B43, 8870,3C ;10379          JSR [L0]                            ;LOAD
U 0B44, BE14,15 ;10380          MOV WR[0] TO LS[T10]
U 0B45, 8870,3C ;10381          JSR [L0]                            ;Shift left
U 0B46, 3E16,15 ;10382          MOV WR[0] TO LS[T11]
U 0B47, 8870,3C ;10383          JSR [L0]                            ;Shift right
U 0B48, 3E10,15 ;10384          MOV WR[0] TO LS[T8]
U 0B49, 8870,3C ;10385          JSR [L0]                            ;STORE
U 0B4A, BE18,15 ;10386          MOV WR[0] TO LS[T12]
                ;10387  NEXT.REG:
U 0B4B, 8870,3C ;10388          JSR [L0]                            ;MOVE
U 0B4C, 3E1C,15 ;10389          MOV WR[0] TO LS[T14]
U 0B4D, 8870,3C ;10390          JSR [L0]                            ;MOVE
U 0B4E, BE1E,15 ;10391          MOV WR[0] TO LS[T15]
U 0B4F, 2F85,15 ;10392          CLR WR[2]                           ;clear counter
                ;10393  NEXT.PATTERN.REST.4:
U 0B50, 0874,6C ;10394          JSR [GET.PATTERN]
                ;10395  LOOP.REST.4:
U 0B51, 8877,DC ;10396          JSR [LOAD.DATA]                     ;Load the data into the FPA
U 0B52, B61C,15 ;10397          MOV LS[T14] TO WR[0]                ;MOV data to the register to be
U 0B53, B716,95 ;10398          MOV LS[#300] TO WR[1]               ; tested.
U 0B54, 90F6,15 ;10399          MISC2 [TRAP.ACC] WR[0]
U 0B55, 10F6,95 ;10400          MISC2 [TRAP.ACC] WR[1]
U 0B56, 8875,9C ;10401          JSR [SHIFT.LEFT.4]                  ;shift the data left four times
U 0B57, 0876,BC ;10402          JSR [SHIFT.RIGHT.4]                 ;shift the data right four times
U 0B58, 361E,15 ;10403          MOV LS[T15] TO WR[0]                ;MOV data from ETX to ET1
U 0B59, B716,95 ;10404          MOV LS[#300] TO WR[1]
U 0B5A, 90F6,15 ;10405          MISC2 [TRAP.ACC] WR[0]
U 0B5B, 10F6,95 ;10406          MISC2 [TRAP.ACC] WR[1]
U 0B5C, 0878,6C ;10407          JSR [STORE.DATA.2]                  ;store the data in LS[T13]
U 0B5D, B61A,15 ;10408          MOV LS[T13] TO WR[0]                ;Set up received data
U 0B5E, 36A2,95 ;10409          MOV LS[DATA.1] TO WR[1]             ;Set up expected data
U 0B5F, 0869,3C ;10410          JSR [CHECK.RESULT]                  ;Report error if there is one
U 0B60, 88B5,14 ;10411          JMP [LOOP.REST.4]                   ;Loop on error
                ;10412          CMP WR[2] WITH LS[#4],
U 0B61, CF45,35 ;10413           DT(LONG)&SET.ALU.CC
U 0B62, 08B5,01 ;10414          JMP.IF[NEQ] TO [NEXT.PATTERN.REST.4]
                ;10415                                              ;If all patterns checked then
                ;10416                                              ;go on else continue looping
U 0B63, FF82,15 ;10417          INC LS[ERROR.NUMBER]                ;If yes then go on to next error
U 0B64, 2F85,15 ;10418          CLR WR[2]                           ;clear counter
                ;10419  NEXT.PATTERN.REST.8:
U 0B65, 0874,6C ;10420          JSR [GET.PATTERN]
                ;10421  LOOP.REST.8:
U 0B66, 8877,DC ;10422          JSR [LOAD.DATA]                     ;Load the data into the FPA
U 0B67, B61C,15 ;10423          MOV LS[T14] TO WR[0]                ;MOV data to the register to be
```

```
U 0B68, B716,95 ;10424          MOV LS[#300] TO WR[1]           ; tested
U 0B69, 90F6,15 ;10425          MISC2 [TRAP.ACC] WR[0]
U 0B6A, 10F6,95 ;10426          MISC2 [TRAP.ACC] WR[1]
U 0B6B, 0876,2C ;10427          JSR [SHIFT.LEFT.8]              ;shift the data left four times
U 0B6C, 8877,4C ;10428          JSR [SHIFT.RIGHT.8]             ;shift the data right four times
U 0B6D, 361E,15 ;10429          MOV LS[T15] TO WR[0]            ;MOV data from ETX to ET1
U 0B6E, B716,95 ;10430          MOV LS[#300] TO WR[1]
U 0B6F, 90F6,15 ;10431          MISC2 [TRAP.ACC] WR[0]
U 0B70, 10F6,95 ;10432          MISC2 [TRAP.ACC] WR[1]
U 0B71, 0878,6C ;10433          JSR [STORE.DATA.2]              ;store the data in LS[T13]
U 0B72, B61A,15 ;10434          MOV LS[T13] TO WR[0]            ;Set up received data
U 0B73, 36A2,95 ;10435          MOV LS[DATA.1] TO WR[1]         ;Set up expected data
U 0B74, 0869,3C ;10436          JSR [CHECK.RESULT]              ;Report error if there is one
U 0B75, 08B6,64 ;10437          JMP [LOOP.REST.8]               ;Loop on error
                ;10438          CMP WR[2] WITH LS[#4],
U 0B76, CF45,35 ;10439           DT(LONG)&SET.ALU.CC
U 0B77, 08B6,51 ;10440          JMP.IF[NEQ] TO [NEXT.PATTERN.REST.8]
                ;10441                                          ;If all patterns checked then
                ;10442                                          ;go on else continue looping
U 0B78, 372C,15 ;10443          MOV LS[#1A] TO WR[0]            ;If all the registers are tested
                ;10444          CMP WR[0] WITH LS[ERROR.NUMBER],
U 0B79, 4F82,35 ;10445           DT(LONG)&SET.ALU.CC            ; then done else test remaining regs
U 0B7A, FF82,15 ;10446          INC LS[ERROR.NUMBER]            ;set up for next error
U 0B7B, 08B4,B1 ;10447          JMP.IF[NEQ] TO [NEXT.REG]       ;check other registers if not done
                ;10448  TE.END:
```

```
;10449  .page
;10450  .TOC    "TEST F ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)"
;10451  ;++
;10452  ;
;10453  ;
;10454  ; Test Description:
;10455  ;
;10456  ;       The purpose of this test is to verify that none of the Address are
;10457  ;       tied together. All sixteen locations will be initialized to zero. Then
;10458  ;       starting at 0 and working towards F read the resgister, write the reg-
;10459  ;       ister, and then read then register again, then go onto the next regis-
;10460  ;       ter. When register F is reached then start at F and work towards 0
;10461  ;       writing zeros. This will verify that the address lines are all correct.
;10462  ;
;10463  ; Assumptions:
;10464  ;
;10465  ;       It is assumed that parity, the NUA lines and the Control Store have
;10466  ;       been checked for and found to be errorless.
;10467  ;
;10468  ; Test Steps:
;10469  ;
;10470  ;       1) Load all registers with zeros.
;10471  ;       2) Read register 0, write ones to register 0 and read register 0 again.
;10472  ;          Then check for error.
;10473  ;       3) Repeat step 2 until this has been done for all registers.
;10474  ;       4) Starting at register F and working toward 0, and writing zeros i
;10475  ;          instead of ones repeat steps 2 and 3.
;10476  ;
;10477  ; Error:
;10478  ;
;10479  ;       NOTE: If other data is one then bits 32 - 56 of the fraction data
;10480  ;             path and bits 0 - 7 of the exponent data path are being checked.
;10481  ;             If other data is two then bits 0 - 31 of the fraction data path
;10482  ;             are being checked.
;10483  ;       Error1 - A or B address lines failed when FT1 was loaded with ones
;10484  ;       Error2 - A or B address lines failed when FT2 was loaded with ones
;10485  ;       Error3 - A or B address lines failed when FT3 was loaded with ones
;10486  ;       Error4 - A or B address lines failed when FT4 was loaded with ones
;10487  ;       Error5 - A or B address lines failed when FT5 was loaded with ones
;10488  ;       Error6 - A or B address lines failed when FT6 was loaded with ones
;10489  ;       Error7 - A or B address lines failed when FT7 was loaded with ones
;10490  ;       Error8 - A or B address lines failed when FT8 was loaded with ones
;10491  ;       Error9 - A or B address lines failed when FT9 was loaded with ones
;10492  ;       ErrorA - A or B address lines failed when FTA was loaded with ones
;10493  ;       ErrorB - A or B address lines failed when FTB was loaded with ones
;10494  ;       ErrorC - A or B address lines failed when FTC was loaded with ones
;10495  ;       ErrorD - A or B address lines failed when FTD was loaded with ones
;10496  ;       ErrorE - A or B address lines failed when FTE was loaded with ones
;10497  ;       ErrorF - A or B address lines failed when FTF was loaded with ones
;10498  ;       Error10 - A or B address lines failed when FT1 was loaded with zeros
;10499  ;       Error11 - A or B address lines failed when FT2 was loaded with zeros
;10500  ;       Error12 - A or B address lines failed when FT3 was loaded with zeros
;10501  ;       Error13 - A or B address lines failed when FT4 was loaded with zeros
;10502  ;       Error14 - A or B address lines failed when FT5 was loaded with zeros
;10503  ;       Error15 - A or B address lines failed when FT6 was loaded with zeros
```

```
                    ;10504  ;       Error16 - A or B address lines failed when FT7 was loaded with zeros
                    ;10505  ;       Error17 - A or B address lines failed when FT8 was loaded with zeros
                    ;10506  ;       Error18 - A or B address lines failed when FT9 was loaded with zeros
                    ;10507  ;       Error19 - A or B address lines failed when FTA was loaded with zeros
                    ;10508  ;       Error1A - A or B address lines failed when FTB was loaded with zeros
                    ;10509  ;       Error1B - A or B address lines failed when FTC was loaded with zeros
                    ;10510  ;       Error1C - A or B address lines failed when FTD was loaded with zeros
                    ;10511  ;       Error1D - A or B address lines failed when FTE was loaded with zeros
                    ;10512  ;       Error1E - A or B address lines failed when FTF was loaded with zeros
                    ;10513  ;
                    ;10514  ; Debug:
                    ;10515  ;
                    ;10516  ;       Error1: If an error occurs and there were no errors in previous tests
                    ;10517  ;               then there are two likely causes of error. The first is that
                    ;10518  ;               two of the B Address lines are tied togehter, the second is
                    ;10519  ;               that the A Address lines failed in transfering the data from
                    ;10520  ;               WR[0]. In either case the error means replacement of the bit
                    ;10521  ;               slice(2901) that's in error.
                    ;10522  ;       Error2 - Error1E: Same as error1.
                    ;10523  ;
                    ;10524  ;--
                    ;10525  T.F:
U 0B7C, B65E,15     ;10526          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                    ;10527                                          ; STATUS WORD
U 0B7D, 3E80,15     ;10528          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;10529                                          ; BIT 15 INDICATES START OF TEST TO
                    ;10530                                          ; CONSOLE
U 0B7E, 10E0,15     ;10531          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;10532  WAIT.TF.0:
U 0B7F, 88B7,F4     ;10533          JMP [WAIT.TF.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                    ;10534                                          ; RESPOND
U 0B80, 087E,6C     ;10535          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;10536                                          ; AND SIZE BITS
U 0B81, B65E,15     ;10537          MOV LS[BIT15] TO WR[0]          ;Set up error mask to check lower 10
U 0B82, 3E8A,15     ;10538          MOV WR[0] TO LS[ERROR.MASK]     ; bits.
U 0B83, B670,15     ;10539          MOV LS[OTHER.DATA] TO WR[0]     ;Set up other data fiald
U 0B84, 3E80,15     ;10540          MOV WR[0] TO LS[CONTROL.STATUS]
U 0B85, 379E,15     ;10541          MOV LS[TF.POINTER] TO WR[0]     ;Initialize pointer for main memory
U 0B86, BE12,15     ;10542          MOV WR[0] TO LS[T9]
U 0B87, 2F85,15     ;10543          CLR WR[2]
U 0B88, 2045,15     ;10544          INC WR[2]
U 0B89, E5A2,15     ;10545          CLR LS[DATA.1]                  ;Set up data to 0
U 0B8A, E5A4,15     ;10546          CLR LS[DATA.2]                  ;Set up data to 0
U 0B8B, 8879,AC     ;10547          JSR [LOAD.DOUBLE]               ;Load zeros into FWR[FT0] and EWR[ET0]
                    ;10548  TF.LOAD:
U 0B8C, 8870,3C     ;10549          JSR [L0]                        ;Get address of MOV
U 0B8D, 3E1C,15     ;10550          MOV WR[0] TO LS[T14]
U 0B8E, 8870,3C     ;10551          JSR [L0]                        ;Get address of FIRST FLOAT STORE
U 0B8F, 3E16,15     ;10552          MOV WR[0] TO LS[T11]
U 0B90, 8870,3C     ;10553          JSR [L0]                        ;Get address of SECOND FLOAT STORE
U 0B91, BE18,15     ;10554          MOV WR[0] TO LS[T12]
U 0B92, 0878,AC     ;10555          JSR [MOV.DATA]                  ;Load zeros into all registers
U 0B93, 2045,15     ;10556          INC WR[2]                       ;add one to counter
                    ;10557          CMP WR[2] WITH LS[#10],         ;Have all registes been loaded?
U 0B94, CF49,35     ;10558           DT(LONG)&SET.ALU.CC
```

```
U 0B95, 88B8,C1 ;10559          JMP.IF[NEQ] TO [TF.LOAD]          ;If not then continue loading
U 0B96, 379E,15 ;10560          MOV LS[TF.POINTER] TO WR[0]       ;else restart pointer at FT1
U 0B97, BE12,15 ;10561          MOV WR[0] TO LS[T9]
U 0B98, 2F85,15 ;10562          CLR WR[2]                         ;Clear counter
U 0B99, 2045,15 ;10563          INC WR[2]                         ; and start it at one
                ;10564  TF.LOOP1:
U 0B9A, 8870,3C ;10565          JSR [LO]                          ;Get address of MOV
U 0B9B, 3E1C,15 ;10566          MOV WR[0] TO LS[T14]
U 0B9C, 8870,3C ;10567          JSR [LO]                          ;Get address of STORE FIRST FLOAT
U 0B9D, 3E16,15 ;10568          MOV WR[0] TO LS[T11]
U 0B9E, 8870,3C ;10569          JSR [LO]                          ;Get address of STORE SECOND FLOAT
U 0B9F, BE18,15 ;10570          MOV WR[0] TO LS[T12]
U 0BA0, 8878,2C ;10571  TF.1:   JSR [STORE.DATA.1]                ;STORE the first float
U 0BA1, B640,15 ;10572          MOV LS[#1] TO WR[0]               ;Set up other data
U 0BA2, BE88,15 ;10573          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BA3, B65E,15 ;10574          MOV LS[BIT15] TO WR[0]            ;Setup error mask
U 0BA4, 3E8A,15 ;10575          MOV WR[0] TO LS[ERROR.MASK]
U 0BA5, B69C,95 ;10576          MOV LS[#0] TO WR[1]               ;Set up expected data
U 0BA6, B61A,15 ;10577          MOV LS[T13] TO WR[0]              ;Set up received data
U 0BA7, 0869,3C ;10578          JSR [CHECK.RESULT]                ;Check for error
U 0BA8, 08BA,04 ;10579          JMP [TF.1]
U 0BA9, 0878,6C ;10580  TF.2:   JSR [STORE.DATA.2]                ;STORE the second float
U 0BAA, 3642,15 ;10581          MOV LS[#2] TO WR[0]               ;Set up other data
U 0BAB, BE88,15 ;10582          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BAC, E58A,15 ;10583          CLR LS[ERROR.MASK]                ;Setup mask for second float
U 0BAD, B69C,95 ;10584          MOV LS[#0] TO WR[1]               ;Set up expected data
U 0BAE, B61A,15 ;10585          MOV LS[T13] TO WR[0]              ;Set up received data
U 0BAF, 0869,3C ;10586          JSR [CHECK.RESULT]                ;Check for error
U 0BB0, 08BA,94 ;10587          JMP [TF.2]                        ;Loop on error
U 0BB1, B69E,15 ;10588          MOV LS[ONES] TO WR[0]             ;Set up data 1 and data 2 with ones
U 0BB2, 3EA2,15 ;10589          MOV WR[0] TO LS[DATA.1]
U 0BB3, 3EA4,15 ;10590          MOV WR[0] TO LS[DATA.2]
U 0BB4, 8879,AC ;10591          JSR [LOAD.DOUBLE]                 ;Load ONES into FT0 and ET0
U 0BB5, 0878,AC ;10592          JSR [MOV.DATA]                    ;MOV the data into the Current register
U 0BB6, 8878,2C ;10593  TF.3:   JSR [STORE.DATA.1]                ;STORE the first float
U 0BB7, B640,15 ;10594          MOV LS[#1] TO WR[0]               ;Set up other data
U 0BB8, BE88,15 ;10595          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BB9, B65E,15 ;10596          MOV LS[BIT15] TO WR[0]            ;Setup error mask for first float
U 0BBA, 3E8A,15 ;10597          MOV WR[0] TO LS[ERROR.MASK]
U 0BBB, 369E,95 ;10598          MOV LS[ONES] TO WR[1]             ;Set up expected data
U 0BBC, B61A,15 ;10599          MOV LS[T13] TO WR[0]              ;Set up received data
U 0BBD, 0869,3C ;10600          JSR [CHECK.RESULT]                ;Check for error
U 0BBE, 88BB,64 ;10601          JMP [TF.3]
U 0BBF, 0878,6C ;10602  TF.4:   JSR [STORE.DATA.2]                ;STORE the second float
U 0BC0, 3642,15 ;10603          MOV LS[#2] TO WR[0]               ;Set up other data
U 0BC1, BE88,15 ;10604          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BC2, E58A,15 ;10605          CLR LS[ERROR.MASK]                ;Setup error mask for the second float
U 0BC3, 369E,95 ;10606          MOV LS[ONES] TO WR[1]             ;Set up expected data
U 0BC4, B61A,15 ;10607          MOV LS[T13] TO WR[0]              ;Set up received data
U 0BC5, 0869,3C ;10608          JSR [CHECK.RESULT]                ;Check for error
U 0BC6, 88BB,F4 ;10609          JMP [TF.4]                        ;Loop on error
U 0BC7, FF82,15 ;10610          INC LS[ERROR.NUMBER]
U 0BC8, 2045,15 ;10611          INC WR[2]                         ;Add one the counter
                ;10612          CMP WR[2] WITH LS[#10],           ;Have all registers been tested ?
U 0BC9, CF49,35 ;10613           DT(LONG)&SET.ALU.CC
```

```
U 0BCA, 08B9,A1 ;10614          JMP.IF[NEQ] TO [TF.LOOP1]       ;If not then check next register
U 0BCB, 2F85,15 ;10615          CLR WR[2]                       ;Set up counter
U 0BCC, 2045,15 ;10616          INC WR[2]
U 0BCD, 3716,15 ;10617          MOV LS[#300] TO WR[0]           ;Set up pointer to backup
U 0BCE, C74A,15 ;10618          BIS LS[#20] TO WR[0]
U 0BCF, 4748,15 ;10619          BIS LS[#10] TO WR[0]
U 0BD0, 4744,15 ;10620          BIS LS[#4] TO WR[0]
U 0BD1, 4742,15 ;10621          BIS LS[#2] TO WR[0]
U 0BD2, BE12,15 ;10622          MOV WR[0] TO LS[T9]
                ;10623  TF.LOOP2:
U 0BD3, 0870,DC ;10624          JSR [-LO]                       ;Get address of STORE SECOND FLOAT
U 0BD4, BE18,15 ;10625          MOV WR[0] TO LS[T12]
U 0BD5, 0870,DC ;10626          JSR [-LO]                       ;Get address of STORE FIRST FLOAT
U 0BD6, 3E16,15 ;10627          MOV WR[0] TO LS[T11]
U 0BD7, 0870,DC ;10628          JSR [-LO]                       ;Get address of MOV
U 0BD8, 3E1C,15 ;10629          MOV WR[0] TO LS[T14]
U 0BD9, 8878,2C ;10630  TF.5:   JSR [STORE.DATA.1]              ;STORE the first float
U 0BDA, B640,15 ;10631          MOV LS[#1] TO WR[0]             ;Set up other data
U 0BDB, BE88,15 ;10632          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BDC, B65E,15 ;10633          MOV LS[BIT15] TO WR[0]          ;Setup error mask for first float
U 0BDD, 3E8A,15 ;10634          MOV WR[0] TO LS[ERROR.MASK]
U 0BDE, 369E,95 ;10635          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0BDF, B61A,15 ;10636          MOV LS[T13] TO WR[0]            ;Set up received data
U 0BE0, 0869,3C ;10637          JSR [CHECK.RESULT]              ;Check for error
U 0BE1, 88BD,94 ;10638          JMP [TF.5]
U 0BE2, 0878,6C ;10639  TF.6:   JSR [STORE.DATA.2]              ;STORE the second float
U 0BE3, 3642,15 ;10640          MOV LS[#2] TO WR[0]             ;Set up other data
U 0BE4, BE88,15 ;10641          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BE5, E58A,15 ;10642          CLR LS[ERROR.MASK]              ;Setup error mask for second float
U 0BE6, 369E,95 ;10643          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0BE7, B61A,15 ;10644          MOV LS[T13] TO WR[0]            ;Set up received data
U 0BE8, 0869,3C ;10645          JSR [CHECK.RESULT]              ;Check for error
U 0BE9, 08BE,24 ;10646          JMP [TF.6]                      ;Loop on error
U 0BEA, 369C,15 ;10647          MOV LS[#0] TO WR[0]             ;Set up data 1 and data 2 with zero
U 0BEB, 3EA2,15 ;10648          MOV WR[0] TO LS[DATA.1]
U 0BEC, 3EA4,15 ;10649          MOV WR[0] TO LS[DATA.2]
U 0BED, 8879,AC ;10650          JSR [LOAD.DOUBLE]               ;Load ZERO into FTO and ETO
U 0BEE, 0878,AC ;10651          JSR [MOV.DATA]                  ;MOV the data into the Current register
U 0BEF, 8878,2C ;10652  TF.7:   JSR [STORE.DATA.1]              ;STORE the first float
U 0BF0, B640,15 ;10653          MOV LS[#1] TO WR[0]             ;Set up other data
U 0BF1, BE88,15 ;10654          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BF2, B65E,15 ;10655          MOV LS[BIT15] TO WR[0]          ;Setup error mask for second float
U 0BF3, 3E8A,15 ;10656          MOV WR[0] TO LS[ERROR.MASK]
U 0BF4, B69C,95 ;10657          MOV LS[#0] TO WR[1]             ;Set up expected data
U 0BF5, B61A,15 ;10658          MOV LS[T13] TO WR[0]            ;Set up received data
U 0BF6, 0869,3C ;10659          JSR [CHECK.RESULT]              ;Check for error
U 0BF7, 88BE,F4 ;10660          JMP [TF.7]
U 0BF8, 0878,6C ;10661  TF.8:   JSR [STORE.DATA.2]              ;STORE the second float
U 0BF9, 3642,15 ;10662          MOV LS[#2] TO WR[0]             ;Set up other data
U 0BFA, BE88,15 ;10663          MOV WR[0] TO LS[ADDRESS.DATA]
U 0BFB, E58A,15 ;10664          CLR LS[ERROR.MASK]              ;Setup error mask for second float
U 0BFC, B69C,95 ;10665          MOV LS[#0] TO WR[1]             ;Set up expected data
U 0BFD, B61A,15 ;10666          MOV LS[T13] TO WR[0]            ;Set up received data
U 0BFE, 0869,3C ;10667          JSR [CHECK.RESULT]              ;Check for error
U 0BFF, 88BF,84 ;10668          JMP [TF.8]                      ;Loop on error
```

```
U 0C00, FF82,15 ;10669          INC LS[ERROR.NUMBER]
U 0C01, 2045,15 ;10670          INC WR[2]                       ;Add one the counter
                ;10671          CMP WR[2] WITH LS[#10],         ;Have all registers been tested ?
U 0C02, CF49,35 ;10672           DT(LONG)&SET.ALU.CC
U 0C03, 88BD,31 ;10673          JMP.IF[NEQ] TO [TF.LOOP2]       ;If not then check next register
                ;10674  TF.END:
```

```
;10675  .page
;10676  .TOC    "TEST 10 EXTENSION DATA PATH BIT ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)"
;10677  ;++
;10678  ;
;10679  ;
;10680  ; Test Description:
;10681  ;
;10682  ;       The purpose of this test is to verify that none of the Address are
;10683  ;       tied together. This will be done for the odd extension working regis-
;10684  ;       ters. All registers will be loaded with zeros then the contents of one
;10685  ;       register will be read, then written with ones then read again. This
;10686  ;       will be done for all registers then starting at FTF read ones, write
;10687  ;       zeros, then read zeros.
;10688  ;
;10689  ; Assumptions:
;10690  ;
;10691  ;       It is assumed that parity, the NUA lines and the Control Store have
;10692  ;       been checked for and found to be errorless.
;10693  ;
;10694  ; Test Steps:
;10695  ;
;10696  ;       1) Load zeros into all the extension bit locations
;10697  ;       2) Read the contents of the FWR. If they are not zero then issue error1
;10698  ;       3) Write ones to the FWR
;10699  ;       4) Read the contents of the FWR. If they are not ones then issue error1
;10700  ;       5) Then starting at FWR[FTF] and working down read ones, write zeros
;10701  ;          then read zeros. If there is an error then issue error 1.
;10702  ;
;10703  ; Error:
;10704  ;
;10705  ;       Error1 - Address lines failed in the extension data path.
;10706  ;
;10707  ; Debug:
;10708  ;
;10709  ;       Error1: If an error occurs and there were no errors in previous tests
;10710  ;               then there are two likely causes of error. The first is that
;10711  ;               two of the B Address lines are tied together, the second is
;10712  ;               that the A Address lines failed in transfering the data from
;10713  ;               WR[0]. In either case the error means replacement of the bit
;10714  ;               slice(2901) that's in error.
;10715  ;       Error2: Same as error 1 except FWR[FT2] failed.
;10716  ;       Error3: Same as error 1 except FWR[FT3] failed.
;10717  ;       Error4: Same as error 1 except FWR[FT4] failed.
;10718  ;       Error5: Same as error 1 except FWR[FT5] failed.
;10719  ;       Error6: Same as error 1 except FWR[FT6] failed.
;10720  ;       Error7: Same as error 1 except FWR[FT7] failed.
;10721  ;       Error8: Same as error 1 except FWR[FT8] failed.
;10722  ;       Error9: Same as error 1 except FWR[FT9] failed.
;10723  ;       ErrorA: Same as error 1 except FWR[FTA] failed.
;10724  ;       ErrorB: Same as error 1 except FWR[FTB] failed.
;10725  ;       ErrorC: Same as error 1 except FWR[FTC] failed.
;10726  ;       ErrorD: Same as error 1 except FWR[FTD] failed.
;10727  ;       ErrorE: Same as error 1 except FWR[FTE] failed.
;10728  ;       ErrorF: Same as error 1 except FWR[FTF] failed.
;10729  ;       Error10: Same as error1 except FWR[FTF] failed.
```

```
                ;10730  ;       Error11: Same as error1 except FWR[FTE] failed.
                ;10731  ;       Error12: Same as error1 except FWR[FTD] failed.
                ;10732  ;       Error13: Same as error1 except FWR[FTC] failed.
                ;10733  ;       Error14: Same as error1 except FWR[FTB] failed.
                ;10734  ;       Error15: Same as error1 except FWR[FTA] failed.
                ;10735  ;       Error16: Same as error1 except FWR[FT9] failed.
                ;10736  ;       Error17: Same as error1 except FWR[FT8] failed.
                ;10737  ;       Error18: Same as error1 except FWR[FT7] failed.
                ;10738  ;       Error19: Same as error1 except FWR[FT6] failed.
                ;10739  ;       Error1A: Same as error1 except FWR[FT5] failed.
                ;10740  ;       Error1B: Same as error1 except FWR[FT4] failed.
                ;10741  ;       Error1C: Same as error1 except FWR[FT3] failed.
                ;10742  ;       Error1D: Same as error1 except FWR[FT2] failed.
                ;10743  ;       Error1E: Same as error1 except FWR[FT1] failed.
                ;10744  ;
                ;10745  ;
                ;10746  ;--
                ;10747  T.10:
U 0C04, B65E,15 ;10748          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;10749                                          ; STATUS WORD
U 0C05, 3E80,15 ;10750          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;10751                                          ; BIT 15 INDICATES START OF TEST TO
                ;10752                                          ; CONSOLE
U 0C06, 10E0,15 ;10753          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;10754  WAIT.T10.0:
U 0C07, 08C0,74 ;10755          JMP [WAIT.T10.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;10756                                          ; RESPOND
U 0C08, 087E,6C ;10757          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;10758                                          ; AND SIZE BITS
U 0C09, B724,15 ;10759          MOV LS[#FFFF00FF] TO WR[0]      ;Set up error mask to check lower 10
U 0C0A, 3E8A,15 ;10760          MOV WR[0] TO LS[ERROR.MASK]     ; bits.
U 0C0B, B7A0,15 ;10761          MOV LS[T10.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 0C0C, BE12,15 ;10762          MOV WR[0] TO LS[T9]
U 0C0D, 8870,3C ;10763          JSR [LO]                        ;Get address of MOV FQ TO FWR[FT0]
U 0C0E, BE1E,15 ;10764          MOV WR[0] TO LS[T15]
U 0C0F, 8870,3C ;10765          JSR [LO]                        ;Get address of STORE FWR[FT0]
U 0C10, BE18,15 ;10766          MOV WR[0] TO LS[T12]
U 0C11, 2F85,15 ;10767          CLR WR[2]                       ;Set pointer to 0
U 0C12, 087C,1C ;10768          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 0C13, 8870,3C ;10769  T10.1:  JSR [LO]                        ;Get address of EVEN MOV
U 0C14, 3E1C,15 ;10770          MOV WR[0] TO LS[T14]
U 0C15, 8870,3C ;10771          JSR [LO]                        ;Get address of STORE THIRD FLOAT
U 0C16, 3E16,15 ;10772          MOV WR[0] TO LS[T11]
U 0C17, 8870,3C ;10773          JSR [LO]                        ;Get address of ODD MOV
U 0C18, 3E10,15 ;10774          MOV WR[0] TO LS[T8]
U 0C19, 8870,3C ;10775          JSR [LO]                        ;Get address of MOV ODD TO Q
U 0C1A, 3EAE,15 ;10776          MOV WR[0] TO LS[T57]
U 0C1B, 0878,AC ;10777          JSR [MOV.DATA]                  ;Load zeros into the extension WR
U 0C1C, B610,15 ;10778          MOV LS[T8] TO WR[0]             ;Setup to force MOV ODD
U 0C1D, B716,95 ;10779          MOV LS[#300] TO WR[1]           ;Setup to loop self
U 0C1E, 90F6,15 ;10780          MISC2 [TRAP.ACC] WR[0]          ;Force MOV ODD
U 0C1F, 10F6,95 ;10781          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0C20, 2045,15 ;10782          INC WR[2]                       ;Add one to the counter
U 0C21, B646,15 ;10783          MOV LS[#8] TO WR[0]             ;Create 7
U 0C22, 2100,15 ;10784          DEC WR[0]
```

```
                 ;10785          CMP WR[2] WITH WR[0],          ;Have all registers been loaded with 0?
U 0C23, 2644,35 ;10786           DT(LONG)&SET.ALU.CC
U 0C24, 08C1,31 ;10787          JMP.IF[NEQ] TO [T10.1]         ;If not then continue loading zeros
U 0C25, 8870,3C ;10788          JSR [L0]                       ;Get address of MOV FT0 TO FTF
U 0C26, 3E1C,15 ;10789          MOV WR[0] TO LS[T14]
U 0C27, 0878,AC ;10790          JSR [MOV.DATA]                 ;Load zeros into FTF
U 0C28, B7A0,15 ;10791          MOV LS[T10.POINTER] TO WR[0]   ;Reset memory pointer to start of test
U 0C29, BE12,15 ;10792          MOV WR[0] TO LS[T9]
U 0C2A, 8870,3C ;10793          JSR [L0]                       ;Move pointer ahead two
U 0C2B, 8870,3C ;10794          JSR [L0]                       ;Move pointer ahead two
U 0C2C, B69E,15 ;10795          MOV LS[ONES] TO WR[0]          ;Load the DATA with ones
U 0C2D, 3EA2,15 ;10796          MOV WR[0] TO LS[DATA.1]
U 0C2E, 3EA4,15 ;10797          MOV WR[0] TO LS[DATA.2]
U 0C2F, BEA6,15 ;10798          MOV WR[0] TO LS[DATA.3]
U 0C30, 0878,FC ;10799          JSR [LOAD.TRIPLE]              ;Load the ones into the FPA
                 ;10800  T10.LOOP:
U 0C31, 8870,3C ;10801          JSR [L0]                       ;Get address of EVEN MOV
U 0C32, 3E1C,15 ;10802          MOV WR[0] TO LS[T14]
U 0C33, 8870,3C ;10803          JSR [L0]                       ;Get address of STORE THIRD FLOAT
U 0C34, 3E16,15 ;10804          MOV WR[0] TO LS[T11]
U 0C35, 8870,3C ;10805          JSR [L0]                       ;Get address of ODD MOV
U 0C36, 3E10,15 ;10806          MOV WR[0] TO LS[T8]
U 0C37, 8870,3C ;10807          JSR [L0]                       ;Get address of MOV ODD TO Q
U 0C38, 3EAE,15 ;10808          MOV WR[0] TO LS[T57]
U 0C39, B6AE,15 ;10809  T10.2:  MOV LS[T57] TO WR[0]           ;Setup to force MOV TO FQ
U 0C3A, B716,95 ;10810          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 0C3B, 90F6,15 ;10811          MISC2 [TRAP.ACC] WR[0]         ;Force MOV TO FQ
U 0C3C, 10F6,95 ;10812          MISC2 [TRAP.ACC] WR[1]         ;Force loop on self
U 0C3D, 361E,15 ;10813          MOV LS[T15] TO WR[0]           ;Setup to force MOV TO FT0
U 0C3E, B716,95 ;10814          MOV LS[#300] TO WR[1]          ;Setup to force loop on self
U 0C3F, 90F6,15 ;10815          MISC2 [TRAP.ACC] WR[0]         ;Force MOV TO FT0
U 0C40, 10F6,95 ;10816          MISC2 [TRAP.ACC] WR[1]         ;Force to loop on self
U 0C41, 0878,6C ;10817          JSR [STORE.DATA.2]             ;Store data from FT0 to CPU
U 0C42, B61A,15 ;10818          MOV LS[T13] TO WR[0]           ;Setup received data
U 0C43, B69C,95 ;10819          MOV LS[#0] TO WR[1]            ;Setup expected data
U 0C44, 0869,3C ;10820          JSR [CHECK.RESULT]             ;Check for error
U 0C45, 88C3,94 ;10821          JMP [T10.2]                    ;Loop on error
U 0C46, 0878,FC ;10822  T10.3:  JSR [LOAD.TRIPLE]              ;Load ONES into extension FT0
U 0C47, B610,15 ;10823          MOV LS[T8] TO WR[0]            ;Setup to MOV ODD
U 0C48, B716,95 ;10824          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 0C49, 90F6,15 ;10825          MISC2 [TRAP.ACC] WR[0]         ;Force MOV ODD
U 0C4A, 10F6,95 ;10826          MISC2 [TRAP.ACC] WR[1]         ;Force loop on self
U 0C4B, B6AE,15 ;10827          MOV LS[T57] TO WR[0]           ;Setup to force MOV TO FQ
U 0C4C, B716,95 ;10828          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 0C4D, 90F6,15 ;10829          MISC2 [TRAP.ACC] WR[0]         ;Force MOV TO FQ
U 0C4E, 10F6,95 ;10830          MISC2 [TRAP.ACC] WR[1]         ;Force loop on self
U 0C4F, 361E,15 ;10831          MOV LS[T15] TO WR[0]           ;Setup to force MOV TO FT0
U 0C50, B716,95 ;10832          MOV LS[#300] TO WR[1]          ;Setup to force loop on self
U 0C51, 90F6,15 ;10833          MISC2 [TRAP.ACC] WR[0]         ;Force MOV TO FT0
U 0C52, 10F6,95 ;10834          MISC2 [TRAP.ACC] WR[1]         ;Force to loop on self
U 0C53, 0878,6C ;10835          JSR [STORE.DATA.2]             ;Store data from FT0 to CPU
U 0C54, B61A,15 ;10836          MOV LS[T13] TO WR[0]           ;Setup received data
U 0C55, 369E,95 ;10837          MOV LS[ONES] TO WR[1]          ;Setup expected data
U 0C56, 0869,3C ;10838          JSR [CHECK.RESULT]             ;Check for error
U 0C57, 08C4,64 ;10839          JMP [T10.3]                    ;Loop on error
```

```
U 0C58, FF82,15 ;10840          INC LS[ERROR.NUMBER]              ;Add one to the error number
U 0C59, 8878,2C ;10841  T10.4:  JSR [STORE.DATA.1]                ;Read the extension WR back to the CPU
U 0C5A, B61A,15 ;10842          MOV LS[T13] TO WR[0]              ;Load the received data into WR[0]
U 0C5B, B69C,95 ;10843          MOV LS[#0] TO WR[1]               ;Set up expected data
U 0C5C, 0869,3C ;10844          JSR [CHECK.RESULT]                ;Check for error
U 0C5D, 88C5,94 ;10845          JMP [T10.4]                       ;Loop on error
U 0C5E, 0878,FC ;10846  T10.5:  JSR [LOAD.TRIPLE]
U 0C5F, 0878,AC ;10847          JSR [MOV.DATA]                    ;Load ones into the extension register
U 0C60, 8878,2C ;10848          JSR [STORE.DATA.1]                ;Store data in the extension register
U 0C61, B61A,15 ;10849          MOV LS[T13] TO WR[0]              ;Setup received data
U 0C62, 369E,95 ;10850          MOV LS[ONES] TO WR[1]             ;Setup expected data
U 0C63, 0869,3C ;10851          JSR [CHECK.RESULT]                ;Check for error
U 0C64, 08C5,E4 ;10852          JMP [T10.5]                       ;Loop on error
U 0C65, FF82,15 ;10853          INC LS[ERROR.NUMBER]
U 0C66, 3648,15 ;10854          MOV LS[#10] TO WR[0]              ;Set up to compare if all regs tested
U 0C67, 2100,15 ;10855          DEC WR[0]
                ;10856          CMP WR[0] WITH LS[ERROR.NUMBER],;Have all registers been tested?
U 0C68, 4F82,35 ;10857           DT(LONG)&SET.ALU.CC
U 0C69, 08C3,11 ;10858          JMP.IF[NEQ] TO [T10.LOOP]         ;If not then go onto next register
U 0C6A, 8870,3C ;10859          JSR [L0]                          ;Get address of MOV FTO TO FTF
U 0C6B, 3E10,15 ;10860          MOV WR[0] TO LS[T8]
U 0C6C, 8870,3C ;10861          JSR [L0]                          ;Get address of MOV FTF TO FQ
U 0C6D, 3EAE,15 ;10862          MOV WR[0] TO LS[T57]
U 0C6E, B6AE,15 ;10863  T10.6:  MOV LS[T57] TO WR[0]              ;Setup to force MOV TO FQ
U 0C6F, B716,95 ;10864          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 0C70, 90F6,15 ;10865          MISC2 [TRAP.ACC] WR[0]            ;Force MOV TO FQ
U 0C71, 10F6,95 ;10866          MISC2 [TRAP.ACC] WR[1]            ;Force loop on self
U 0C72, 361E,15 ;10867          MOV LS[T15] TO WR[0]              ;Setup to force MOV TO FTO
U 0C73, B716,95 ;10868          MOV LS[#300] TO WR[1]             ;Setup to force loop on self
U 0C74, 90F6,15 ;10869          MISC2 [TRAP.ACC] WR[0]            ;Force MOV TO FTO
U 0C75, 10F6,95 ;10870          MISC2 [TRAP.ACC] WR[1]            ;Force to loop on self
U 0C76, 0878,6C ;10871          JSR [STORE.DATA.2]                ;Store data from FTO to CPU
U 0C77, B61A,15 ;10872          MOV LS[T13] TO WR[0]              ;Setup received data
U 0C78, B69C,95 ;10873          MOV LS[#0] TO WR[1]               ;Setup expected data
U 0C79, 0869,3C ;10874          JSR [CHECK.RESULT]                ;Check for error
U 0C7A, 08C6,E4 ;10875          JMP [T10.6]                       ;Loop on error
U 0C7B, 0878,FC ;10876  T10.7:  JSR [LOAD.TRIPLE]                 ;Load ONES into extension FTO
U 0C7C, B610,15 ;10877          MOV LS[T8] TO WR[0]               ;Setup to MOV ODD
U 0C7D, B716,95 ;10878          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 0C7E, 90F6,15 ;10879          MISC2 [TRAP.ACC] WR[0]            ;Force MOV ODD
U 0C7F, 10F6,95 ;10880          MISC2 [TRAP.ACC] WR[1]            ;Force loop on self
U 0C80, B6AE,15 ;10881          MOV LS[T57] TO WR[0]              ;Setup to force MOV TO FQ
U 0C81, B716,95 ;10882          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 0C82, 90F6,15 ;10883          MISC2 [TRAP.ACC] WR[0]            ;Force MOV TO FQ
U 0C83, 10F6,95 ;10884          MISC2 [TRAP.ACC] WR[1]            ;Force loop on self
U 0C84, 361E,15 ;10885          MOV LS[T15] TO WR[0]              ;Setup to force MOV TO FTO
U 0C85, B716,95 ;10886          MOV LS[#300] TO WR[1]             ;Setup to force loop on self
U 0C86, 90F6,15 ;10887          MISC2 [TRAP.ACC] WR[0]            ;Force MOV TO FTO
U 0C87, 10F6,95 ;10888          MISC2 [TRAP.ACC] WR[1]            ;Force to loop on self
U 0C88, 0878,6C ;10889          JSR [STORE.DATA.2]                ;Store data from FTO to CPU
U 0C89, B61A,15 ;10890          MOV LS[T13] TO WR[0]              ;Setup received data
U 0C8A, 369E,95 ;10891          MOV LS[ONES] TO WR[1]             ;Setup expected data
U 0C8B, 0869,3C ;10892          JSR [CHECK.RESULT]                ;Check for error
U 0C8C, 88C7,B4 ;10893          JMP [T10.7]                       ;Loop on error
U 0C8D, FF82,15 ;10894          INC LS[ERROR.NUMBER]              ;Go onto error 10
```

```
U 0C8E, 087C.1C ;10895          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 0C8F, 3734.15 ;10896          MOV LS[#370] TO WR[0]           ;Set the memory pointer to 372
U 0C90, 47-4.15 ;10897          BIS LS[#4] TO WR[0]
U 0C91, 4742.15 ;10898          BIS LS[#2] TO WR[0]
U 0C92, BE12.15 ;10899          MOV WR[0] TO LS[T9]
U 0C93, 0870.DC ;10900          JSR [-L0]                       ;Get address of MOV FTF TO FQ
U 0C94, 3EAE.15 ;10901          MOV WR[0] TO LS[T57]
U 0C95, 0870.DC ;10902          JSR [-L0]                       ;Get address of MOV FT0 TO FTF
U 0C96, 3E10.15 ;10903          MOV WR[0] TO LS[T8]
U 0C97, B6AE.15 ;10904  T10.8:  MOV LS[T57] TO WR[0]            ;Setup to force MOV TO FQ
U 0C98, B716.95 ;10905          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0C99, 90F6.15 ;10906          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FQ
U 0C9A, 10F6.95 ;10907          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0C9B, 361E.15 ;10908          MOV LS[T15] TO WR[0]            ;Setup to force MOV TO FT0
U 0C9C, B716.95 ;10909          MOV LS[#300] TO WR[1]           ;Setup to force loop on self
U 0C9D, 90F6.15 ;10910          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FT0
U 0C9E, 10F6.95 ;10911          MISC2 [TRAP.ACC] WR[1]          ;Force to loop on self
U 0C9F, 0878.6C ;10912          JSR [STORE.DATA.2]              ;Store data from FT0 to CPU
U 0CA0, B61A.15 ;10913          MOV LS[T13] TO WR[0]            ;Setup received data
U 0CA1, 369E.95 ;10914          MOV LS[ONES] TO WR[1]           ;Setup expected data
U 0CA2, 0869.3C ;10915          JSR [CHECK.RESULT]              ;Check for error
U 0CA3, 08C9.74 ;10916          JMP [T10.8]                     ;Loop on error
U 0CA4, 0878.FC ;10917  T10.9:  JSR [LOAD.TRIPLE]               ;Load ONES into extension FT0
U 0CA5, B610.15 ;10918          MOV LS[T8] TO WR[0]             ;Setup to MOV ODD
U 0CA6, B716.95 ;10919          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0CA7, 90F6.15 ;10920          MISC2 [TRAP.ACC] WR[0]          ;Force MOV ODD
U 0CA8, 10F6.95 ;10921          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0CA9, B6AE.15 ;10922          MOV LS[T57] TO WR[0]            ;Setup to force MOV TO FQ
U 0CAA, B716.95 ;10923          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0CAB, 90F6.15 ;10924          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FQ
U 0CAC, 10F6.95 ;10925          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0CAD, 361E.15 ;10926          MOV LS[T15] TO WR[0]            ;Setup to force MOV TO FT0
U 0CAE, B716.95 ;10927          MOV LS[#300] TO WR[1]           ;Setup to force loop on self
U 0CAF, 90F6.15 ;10928          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FT0
U 0CB0, 10F6.95 ;10929          MISC2 [TRAP.ACC] WR[1]          ;Force to loop on self
U 0CB1, 0878.6C ;10930          JSR [STORE.DATA.2]              ;Store data from FT0 to CPU
U 0CB2, B61A.15 ;10931          MOV LS[T13] TO WR[0]            ;Setup received data
U 0CB3, B69C.95 ;10932          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0CB4, 0869.3C ;10933          JSR [CHECK.RESULT]              ;Check for error
U 0CB5, 08CA.44 ;10934          JMP [T10.9]                     ;Loop on error
U 0CB6, FF82.15 ;10935          INC LS[ERROR.NUMBER]            ;Go onto error 11
                ;10936  T10.LOOP1:
U 0CB7, 0870.DC ;10937          JSR [-L0]                       ;Get address of MOV ODD TO Q
U 0CB8, 3EAE.15 ;10938          MOV WR[0] TO LS[T57]
U 0CB9, 0870.DC ;10939          JSR [-L0]                       ;Get address of ODD MOV
U 0CBA, 3E10.15 ;10940          MOV WR[0] TO LS[T8]
U 0CBB, 0870.DC ;10941          JSR [-L0]                       ;Get address of STORE THIRD FLOAT
U 0CBC, 3E16.15 ;10942          MOV WR[0] TO LS[T11]
U 0CBD, 0870.DC ;10943          JSR [-L0]                       ;Get address of EVEN MOV
U 0CBE, 3E1C.15 ;10944          MOV WR[0] TO LS[T14]
U 0CBF, 8878.2C ;10945  T10.C:  JSR [STORE.DATA.1]              ;Read the extension WR back to the CPU
U 0CC0, B61A.15 ;10946          MOV LS[T13] TO WR[0]            ;Load the received data into WR[0]
U 0CC1, 369E.95 ;10947          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0CC2, 0869.3C ;10948          JSR [CHECK.RESULT]              ;Check for error
U 0CC3, 08CB.F4 ;10949          JMP [T10.C]                     ;Loop on error
```

```
U 0CC4, 0878,FC ;10950  T10.D:  JSR [LOAD.TRIPLE]
U 0CC5, 0878,AC ;10951          JSR [MOV.DATA]                  ;Load zero into the extension register
U 0CC6, 8878,2C ;10952          JSR [STORE.DATA.1]              ;Store data in the extension register
U 0CC7, B61A,15 ;10953          MOV LS[T13] TO WR[0]            ;Setup received data
U 0CC8, B69C,95 ;10954          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0CC9, 0869,3C ;10955          JSR [CHECK.RESULT]              ;Check for error
U 0CCA, 08CC,44 ;10956          JMP [T10.D]                     ;Loop on error
U 0CCB, FF82,15 ;10957          INC LS[ERROR.NUMBER]
U 0CCC, B6AE,15 ;10958  T10.A:  MOV LS[T57] TO WR[0]            ;Setup to force MOV TO FQ
U 0CCD, B716,95 ;10959          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0CCE, 90F6,15 ;10960          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FQ
U 0CCF, 10F6,95 ;10961          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0CD0, 361E,15 ;10962          MOV LS[T15] TO WR[0]            ;Setup to force MOV TO FTO
U 0CD1, B716,95 ;10963          MOV LS[#300] TO WR[1]           ;Setup to force loop on self
U 0CD2, 90F6,15 ;10964          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FTO
U 0CD3, 10F6,95 ;10965          MISC2 [TRAP.ACC] WR[1]          ;Force to loop on self
U 0CD4, 0878,6C ;10966          JSR [STORE.DATA.2]              ;Store data from FTO to CPU
U 0CD5, B61A,15 ;10967          MOV LS[T13] TO WR[0]            ;Setup received data
U 0CD6, 369E,95 ;10968          MOV LS[ONES] TO WR[1]           ;Setup expected data
U 0CD7, 0869,3C ;10969          JSR [CHECK.RESULT]              ;Check for error
U 0CD8, 88CC,C4 ;10970          JMP [T10.A]                     ;Loop on error
U 0CD9, 0878,FC ;10971  T10.B:  JSR [LOAD.TRIPLE]               ;Load zero into extension FTO
U 0CDA, B610,15 ;10972          MOV LS[T8] TO WR[0]             ;Setup to MOV ODD
U 0CDB, B716,95 ;10973          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0CDC, 90F6,15 ;10974          MISC2 [TRAP.ACC] WR[0]          ;Force MOV ODD
U 0CDD, 10F6,95 ;10975          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0CDE, B6AE,15 ;10976          MOV LS[T57] TO WR[0]            ;Setup to force MOV TO FQ
U 0CDF, B716,95 ;10977          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0CE0, 90F6,15 ;10978          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FQ
U 0CE1, 10F6,95 ;10979          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0CE2, 361E,15 ;10980          MOV LS[T15] TO WR[0]            ;Setup to force MOV TO FTO
U 0CE3, B716,95 ;10981          MOV LS[#300] TO WR[1]           ;Setup to force loop on self
U 0CE4, 90F6,15 ;10982          MISC2 [TRAP.ACC] WR[0]          ;Force MOV TO FTO
U 0CE5, 10F6,95 ;10983          MISC2 [TRAP.ACC] WR[1]          ;Force to loop on self
U 0CE6, 0878,6C ;10984          JSR [STORE.DATA.2]              ;Store data from FTO to CPU
U 0CE7, B61A,15 ;10985          MOV LS[T13] TO WR[0]            ;Setup received data
U 0CE8, B69C,95 ;10986          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0CE9, 0869,3C ;10987          JSR [CHECK.RESULT]              ;Check for error
U 0CEA, 08CD,94 ;10988          JMP [T10.B]                     ;Loop on error
U 0CEB, FF82,15 ;10989          INC LS[ERROR.NUMBER]            ;Add one to the error number
U 0CEC, B64A,15 ;10990          MOV LS[#20] TO WR[0]            ;Setup #1F
U 0CED, 2100,15 ;10991          DEC WR[0]
                ;10992          CMP WR[0] WITH LS[ERROR.NUMBER],;Have all registers been tested?
U 0CEE, 4F82,35 ;10993           DT(LONG)&SET.ALU.CC
U 0CEF, 88CB,71 ;10994          JMP.IF[NEQ] TO [T10.LOOP1]      ;If not then go onto next register
                ;10995  T10.END:
```

```
;10996  .page
;10997  .TOC    "TEST 11 UPPER BIT EXPONENT DATA PATH BIT ADDRESS LINE INTEGRITY TEST(M8389 FPA MODULE)"
;10998  ;++
;10999  ;
;11000  ;
;11001  ; Test Description:
;11002  ;
;11003  ;       The purpose of this test is to verify that none of the Address are
;11004  ;       tied together. This will be done for the upper bits of the working reg-
;11005  ;       isters. All registers will be loaded with zeros then the contents of
;11006  ;       one register will be read, then written with ones then read again. This
;11007  ;       will be done for all registers then starting at FTF read ones, write
;11008  ;       zeros, then read zeros.
;11009  ;
;11010  ; Assumptions:
;11011  ;
;11012  ;       It is assumed that parity, the NUA lines and the Control Store have
;11013  ;       been checked for and found to be errorless.
;11014  ;
;11015  ; Test Steps:
;11016  ;
;11017  ;       1) Load zeros into all the upper bits of the exponentlocations
;11018  ;       2) Sshift the upper bits down, and read the exponent bits back to the
;11019  ;          CPU.
;11020  ;       3) Write ones to the exponent bits and shift them to the upper bits
;11021  ;       4) Read the contents of the upper bits of the exponent data path. If
;11022  ;          they are not ones then issue error1.
;11023  ;       5) Repeat this for each register.
;11024  ;
;11025  ; Error:
;11026  ;
;11027  ;       Error1 - Address lines failed in the upper bits of the exponent data
;11028  ;                path.
;11029  ;
;11030  ; Debug:
;11031  ;
;11032  ;       Error1: If an error occurs and there were no errors in previous tests
;11033  ;               then there are two likely causes of error. The first is that
;11034  ;               two of the B Address lines are tied togehter, the second is
;11035  ;               that the A Address lines failed in transfering the data from
;11036  ;               WR[0]. In either case the error means replacement of the bit
;11037  ;               slice that's in error. For EWR[ET2]
;11038  ;       Error2: Same as error 1 except EWR[ET6] failed.
;11039  ;       Error3: Same as error 1 except EWR[ET1] failed.
;11040  ;       Error4: Same as error 1 except EWR[ET3] failed.
;11041  ;       Error5: Same as error 1 except EWR[ET4] failed.
;11042  ;       Error6: Same as error 1 except EWR[ET5] failed.
;11043  ;       Error7: Same as error 1 except EWR[ET7] failed.
;11044  ;       Error8: Same as error 1 except EWR[ET8] failed.
;11045  ;       Error9: Same as error 1 except EWR[ET9] failed.
;11046  ;       ErrorA: Same as error 1 except EWR[ETA] failed.
;11047  ;       ErrorB: Same as error 1 except EWR[ETB] failed.
;11048  ;       ErrorC: Same as error 1 except EWR[ETC] failed.
;11049  ;       ErrorD: Same as error 1 except EWR[ETD] failed.
;11050  ;       ErrorE: Same as error 1 except EWR[ETE] failed.
```

```
                ;11051  ;       ErrorF: Same as error 1 except EWR[ETF] failed.
                ;11052  ;       Error10: Same as error 1 except EWR[ETF] failed.
                ;11053  ;       Error11: Same as error 1 except EWR[ETE] failed.
                ;11054  ;       Error12: Same as error 1 except EWR[ETD] failed.
                ;11055  ;       Error13: Same as error 1 except EWR[ETC] failed.
                ;11056  ;       Error14: Same as error 1 except EWR[ETB] failed.
                ;11057  ;       Error15: Same as error 1 except EWR[ETA] failed.
                ;11058  ;       Error16: Same as error 1 except EWR[ET9] failed.
                ;11059  ;       Error17: Same as error 1 except EWR[ET8] failed.
                ;11060  ;       Error18: Same as error 1 except EWR[ET7] failed.
                ;11061  ;       Error19: Same as error 1 except EWR[ET5] failed.
                ;11062  ;       Error1A: Same as error 1 except EWR[ET4] failed.
                ;11063  ;       Error1B: Same as error 1 except EWR[ET3] failed.
                ;11064  ;       Error1C: Same as error 1 except EWR[ET1] failed.
                ;11065  ;       Error1D: Same as error 1 except EWR[ET2] failed.
                ;11066  ;
                ;11067  ;
                ;11068  ;--
                ;11069  T.11:
U 0CF0, B65E,15 ;11070          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;11071                                          ; STATUS WORD
U 0CF1, 3E80,15 ;11072          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;11073                                          ; BIT 15 INDICATES START OF TEST TO
                ;11074                                          ; CONSOLE
U 0CF2, 10E0,15 ;11075          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;11076  WAIT.T11.0:
U 0CF3, 88CF,34 ;11077          JMP [WAIT.T11.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;11078                                          ; RESPOND
U 0CF4, 087E,6C ;11079          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;11080                                          ; AND SIZE BITS
U 0CF5, 371A,15 ;11081          MOV LS[#FFFF807F] TO WR[0]
U 0CF6, 3E8A,15 ;11082          MOV WR[0] TO LS[ERROR.MASK]     ;Set up new error mask
U 0CF7, 37A2,15 ;11083          MOV LS[T11.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 0CF8, BE12,15 ;11084          MOV WR[0] TO LS[T9]
U 0CF9, 8870,3C ;11085          JSR [LO]                        ;Get address of SHIFT ET0 LEFT 8
U 0CFA, 3E16,15 ;11086          MOV WR[0] TO LS[T11]
U 0CFB, E5A2,15 ;11087          CLR LS[DATA.1]                  ;Load zeros into the DATA
U 0CFC, E5A4,15 ;11088          CLR LS[DATA.2]                  ;Load zeros into the DATA
U 0CFD, 8879,AC ;11089          JSR [LOAD.DOUBLE]               ;Load the exponent data path
U 0CFE, 0876,2C ;11090          JSR [SHIFT.LEFT.8]              ;Load zeros into the upper bits
U 0CFF, 8870,3C ;11091          JSR [LO]                        ;Get address of MOV ET0 TO ET2
U 0D00, 3E1C,15 ;11092          MOV WR[0] TO LS[T14]
U 0D01, 3644,15 ;11093          MOV LS[#4] TO WR[0]             ;Pass by the next two memory locations
U 0D02, 6C12,15 ;11094          ADD WR[0] TO LS[T9]
U 0D03, 0878,AC ;11095          JSR [MOV.DATA]                  ;Load the upper bits of ET2 with 0
U 0D04, 8870,3C ;11096          JSR [LO]                        ;Get address of MOV ET0 TO ET6
U 0D05, 3E1C,15 ;11097          MOV WR[0] TO LS[T14]
U 0D06, 3644,15 ;11098          MOV LS[#4] TO WR[0]             ;Pass by the next two memory locations
U 0D07, 6C12,15 ;11099          ADD WR[0] TO LS[T9]
U 0D08, 0878,AC ;11100          JSR [MOV.DATA]
U 0D09, 2F85,15 ;11101          CLR WR[2]                       ;Clear counter
                ;11102  T11.LOOP:
U 0D0A, 8870,3C ;11103          JSR [LO]                        ;Get address of MOV ET0 TO ETX
U 0D0B, 3E1C,15 ;11104          MOV WR[0] TO LS[T14]
U 0D0C, 8870,3C ;11105          JSR [LO]                        ;Get address of MOV ETX TO ET6
```

```
U 0D0D, BE1E,15 ;11106          MOV WR[0] TO LS[T15]
U 0D0E, 0878,AC ;11107          JSR [MOV.DATA]                  ;Load ETX with zeros
U 0D0F, 2045,15 ;11108          INC WR[2]                       ;Add one to the counter
U 0D10, B646,15 ;11109          MOV LS[#8] TO WR[0]             ;Create #D
U 0D11, 4744,15 ;11110          BIS LS[#4] TO WR[0]
U 0D12, C740,15 ;11111          BIS LS[#1] TO WR[0]
                ;11112          CMP WR[0] WITH WR[2],           ;Have all registers been loaded with 0
U 0D13, 2641,35 ;11113           DT(LONG)&SET.ALU.CC
U 0D14, 08D0,A1 ;11114          JMP.IF[NEQ] TO [T11.LOOP]
U 0D15, 37A2,15 ;11115          MOV LS[T11.POINTER] TO WR[0]    ;Set the pointer back to the begining
U 0D16, BE12,15 ;11116          MOV WR[0] TO LS[T9]
U 0D17, 8870,3C ;11117          JSR [L0]                        ;Get address of SHIFT ET0 LEFT 8
U 0D18, 3E16,15 ;11118          MOV WR[0] TO LS[T11]
U 0D19, B69E,15 ;11119          MOV LS[ONES] TO WR[0]           ;Set DATA to ones
U 0D1A, 3EA2,15 ;11120          MOV WR[0] TO LS[DATA.1]
U 0D1B, 3EA4,15 ;11121          MOV WR[0] TO LS[DATA.2]
U 0D1C, 8879,AC ;11122          JSR [LOAD.DOUBLE]               ;Load ones into the lower bits of the
                ;11123                                          ; exponent
U 0D1D, 0876,2C ;11124          JSR [SHIFT.LEFT.8]              ;Shift the ones into the upper bits
U 0D1E, 8870,3C ;11125          JSR [L0]                        ;Get MOV ET0 TO ET2
U 0D1F, 3E1C,15 ;11126          MOV WR[0] TO LS[T14]
U 0D20, 8870,3C ;11127          JSR [L0]                        ;Get SHIFT RIGHT ET2
U 0D21, 3E10,15 ;11128          MOV WR[0] TO LS[T8]
U 0D22, 8870,3C ;11129          JSR [L0]                        ;Get STORE ET2
U 0D23, BE18,15 ;11130          MOV WR[0] TO LS[T12]
U 0D24, 8877,4C ;11131          JSR [SHIFT.RIGHT.8]             ;Shift the upper bits of ET2 right 8
U 0D25, 0878,6C ;11132  T11.1:  JSR [STORE.DATA.2]              ;Store the data into LS[T13]
U 0D26, B61A,15 ;11133          MOV LS[T13] TO WR[0]            ;Setup received data
U 0D27, B69C,95 ;11134          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0D28, 0869,3C ;11135          JSR [CHECK.RESULT]              ;Check for error
U 0D29, 88D2,54 ;11136          JMP [T11.1]                     ;Loop on error
U 0D2A, 0878,AC ;11137  T11.2:  JSR [MOV.DATA]                  ;Load ones into the upper bits of ET2
U 0D2B, 8877,4C ;11138          JSR [SHIFT.RIGHT.8]             ;Shift the upper bits of ET2 right 8
U 0D2C, 0878,6C ;11139          JSR [STORE.DATA.2]              ;Store the data into LS[T13]
U 0D2D, 0878,AC ;11140          JSR [MOV.DATA]                  ;Reload the upper bits with ones
U 0D2E, B61A,15 ;11141          MOV LS[T13] TO WR[0]            ;Set up received data
U 0D2F, 369E,95 ;11142          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0D30, 0869,3C ;11143          JSR [CHECK.RESULT]              ;Check for error
U 0D31, 88D2,A4 ;11144          JMP [T11.2]                     ;Loop on error
U 0D32, FF82,15 ;11145          INC LS[ERROR.NUMBER]
U 0D33, 8870,3C ;11146          JSR [L0]                        ;Get MOV ET0 TO ET6
U 0D34, 3E1C,15 ;11147          MOV WR[0] TO LS[T14]
U 0D35, 8870,3C ;11148          JSR [L0]                        ;Get SHIFT RIGHT ET6
U 0D36, 3E10,15 ;11149          MOV WR[0] TO LS[T8]
U 0D37, 8870,3C ;11150          JSR [L0]                        ;Get STORE ET6
U 0D38, BE18,15 ;11151          MOV WR[0] TO LS[T12]
U 0D39, 8877,4C ;11152          JSR [SHIFT.RIGHT.8]             ;Shift the upper bits of ET6 right 8
U 0D3A, 0878,6C ;11153  T11.3:  JSR [STORE.DATA.2]              ;Store the data into LS[T13]
U 0D3B, B61A,15 ;11154          MOV LS[T13] TO WR[0]            ;Setup received data
U 0D3C, B69C,95 ;11155          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0D3D, 0869,3C ;11156          JSR [CHECK.RESULT]              ;Check for error
U 0D3E, 08D3,A4 ;11157          JMP [T11.3]                     ;Loop on error
U 0D3F, 0878,AC ;11158  T11.4:  JSR [MOV.DATA]                  ;Load ones into the upper bits of ET6
U 0D40, 8877,4C ;11159          JSR [SHIFT.RIGHT.8]             ;Shift the upper bits of ET6 right 8
U 0D41, 0878,6C ;11160          JSR [STORE.DATA.2]              ;Store the data into LS[T13]
```

```
U 0D42, B61A,15 ;11161          MOV LS[T13] TO WR[0]            ;Set up received data
U 0D43, 369E,95 ;11162          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0D44, 0869,3C ;11163          JSR [CHECK.RESULT]              ;Check for error
U 0D45, 08D3,F4 ;11164          JMP [T11.4]                     ;Loop on error
U 0D46, FF82,15 ;11165          INC LS[ERROR.NUMBER]            ;Add one to the error number
U 0D47, 2F85,15 ;11166          CLR WR[2]                       ;Clear counter
                ;11167  T11.LOOP1:
U 0D48, 8870,3C ;11168          JSR [L0]                        ;Get address of MOV ET0 TO ETX
U 0D49, 3E1C,15 ;11169          MOV WR[0] TO LS[T14]
U 0D4A, 8870,3C ;11170          JSR [L0]                        ;Get address of MOV ETX TO ET6
U 0D4B, BE1E,15 ;11171          MOV WR[0] TO LS[T15]
U 0D4C, 361E,15 ;11172  T11.5:  MOV LS[T15] TO WR[0]            ;Setup to force MOV ETX TO ET6
U 0D4D, B716,95 ;11173          MOV LS[#300] TO WR[1]
U 0D4E, 90F6,15 ;11174          MISC2 [TRAP.ACC] WR[0]          ;Force MOV ETX TO ET6
U 0D4F, 10F6,95 ;11175          MISC2 [TRAP.ACC] WR[1]          ;Force Loop on self
U 0D50, 8877,4C ;11176          JSR [SHIFT.RIGHT.8]             ;Shift data into lower bits
U 0D51, 0878,6C ;11177          JSR [STORE.DATA.2]              ;Store data into LS[T13]
U 0D52, B69C,95 ;11178          MOV LS[#0] TO WR[1]             ;Setup expected data
U 0D53, B61A,15 ;11179          MOV LS[T13] TO WR[0]            ;Setup received data
U 0D54, 0869,3C ;11180          JSR [CHECK.RESULT]              ;Check for error
U 0D55, 88D4,C4 ;11181          JMP [T11.5]                     ;Loop on error
U 0D56, 0878,AC ;11182  T11.6:  JSR [MOV.DATA]                  ;Load ones into ETX
U 0D57, 361E,15 ;11183          MOV LS[T15] TO WR[0]            ;Set up to MOV ETX TO ET6
U 0D58, B716,95 ;11184          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 0D59, 90F6,15 ;11185          MISC2 [TRAP.ACC] WR[0]          ;Force MOV ETX TO ET6
U 0D5A, 10F6,95 ;11186          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
U 0D5B, 8877,4C ;11187          JSR [SHIFT.RIGHT.8]             ;Shift the data into lower bits
U 0D5C, 0878,6C ;11188          JSR [STORE.DATA.2]              ;Store data into LS[T13]
U 0D5D, 369E,95 ;11189          MOV LS[ONES] TO WR[1]           ;Set up expected data
U 0D5E, B61A,15 ;11190          MOV LS[T13] TO WR[0]            ;Setup received data
U 0D5F, 0869,3C ;11191          JSR [CHECK.RESULT]              ;Check for error
U 0D60, 08D5,64 ;11192          JMP [T11.6]                     ;Loop on error
U 0D61, FF82,15 ;11193          INC LS[ERROR.NUMBER]
U 0D62, 2045,15 ;11194          INC WR[2]                       ;Add one to the counter
U 0D63, B646,15 ;11195          MOV LS[#8] TO WR[0]             ;Create D
U 0D64, 4744,15 ;11196          BIS LS[#4] TO WR[0]
U 0D65, C740,15 ;11197          BIS LS[#1] TO WR[0]
                ;11198          CMP WR[0] WITH WR[2],           ;Have all registers been checked
U 0D66, 2641,35 ;11199           DT(LONG)&SET.ALU.CC
U 0D67, 08D4,81 ;11200          JMP.IF[NEQ] TO [T11.LOOP1]      ;If not continue checking
U 0D68, 2F85,15 ;11201          CLR WR[2]                       ;Clear counter
U 0D69, 3716,15 ;11202          MOV LS[#300] TO WR[0]           ;Create 388
U 0D6A, 474E,15 ;11203          BIS LS[#80] TO WR[0]
U 0D6B, C74A,15 ;11204          BIS LS[#20] TO WR[0]
U 0D6C, 4748,15 ;11205          BIS LS[#10] TO WR[0]
U 0D6D, C746,15 ;11206          BIS LS[#8] TO WR[0]
U 0D6E, BE12,15 ;11207          MOV WR[0] TO LS[T9]
U 0D6F, E5A2,15 ;11208          CLR LS[DATA.1]                  ;Load zeros into the data
U 0D70, E5A4,15 ;11209          CLR LS[DATA.2]
U 0D71, 8879,AC ;11210          JSR [LOAD.DOUBLE]               ;Load zero into lower bits of ET0
U 0D72, 0876,2C ;11211          JSR [SHIFT.LEFT.8]              ;Shift zero into the upper bits of ET0
                ;11212  T11.LOOP2:
U 0D73, 0870,DC ;11213          JSR [-L0]                       ;Get address of MOV ETX TO ET6
U 0D74, BE1E,15 ;11214          MOV WR[0] TO LS[T15]
U 0D75, 0870,DC ;11215          JSR [-L0]                       ;Get address of MOV ET0 TO ETX
```

```
U 0D76, 3E1C,15 ;11216                MOV WR[0] TO LS[T14]
U 0D77, 361E,15 ;11217  T11.7:        MOV LS[T15] TO WR[0]          ;Setup to force MOV ETX TO ET6
U 0D78, B716,95 ;11218                MOV LS[#300] TO WR[1]
U 0D79, 90F6,15 ;11219                MISC2 [TRAP.ACC] WR[0]        ;Force MOV ETX TO ET6
U 0D7A, 10F6,95 ;11220                MISC2 [TRAP.ACC] WR[1]        ;Force Loop on self
U 0D7B, 8877,4C ;11221                JSR [SHIFT.RIGHT.8]           ;Shift data into lower bits
U 0D7C, 0878,6C ;11222                JSR [STORE.DATA.2]            ;Store data into LS[T13]
U 0D7D, 369E,95 ;11223                MOV LS[ONES] TO WR[1]         ;Setup expected data
U 0D7E, B61A,15 ;11224                MOV LS[T13] TO WR[0]          ;Setup received data
U 0D7F, 0869,3C ;11225                JSR [CHECK.RESULT]            ;Check for error
U 0D80, 08D7,74 ;11226                JMP [T11.7]                   ;Loop on error
U 0D81, 0878,AC ;11227  T11.8:        JSR [MOV.DATA]                ;Load ones into ETX
U 0D82, 361E,15 ;11228                MOV LS[T15] TO WR[0]          ;Set up to MOV ETX TO ET6
U 0D83, B716,95 ;11229                MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 0D84, 90F6,15 ;11230                MISC2 [TRAP.ACC] WR[0]        ;Force MOV ETX TO ET6
U 0D85, 10F6,95 ;11231                MISC2 [TRAP.ACC] WR[1]        ;Force loop on self
U 0D86, 8877,4C ;11232                JSR [SHIFT.RIGHT.8]           ;Shift the data into lower bits
U 0D87, 0878,6C ;11233                JSR [STORE.DATA.2]            ;Store data into LS[T13]
U 0D88, B69C,95 ;11234                MOV LS[#0] TO WR[1]           ;Set up expected data
U 0D89, B61A,15 ;11235                MOV LS[T13] TO WR[0]          ;Setup received data
U 0D8A, 0869,3C ;11236                JSR [CHECK.RESULT]            ;Check for error
U 0D8B, 08D8,14 ;11237                JMP [T11.8]                   ;Loop on error
U 0D8C, FF82,15 ;11238                INC LS[ERROR.NUMBER]
U 0D8D, 2045,15 ;11239                INC WR[2]                     ;Add one to the counter
U 0D8E, B646,15 ;11240                MOV LS[#8] TO WR[0]           ;Create D
U 0D8F, 4744,15 ;11241                BIS LS[#4] TO WR[0]
U 0D90, C740,15 ;11242                BIS LS[#1] TO WR[0]
                ;11243                CMP WR[0] WITH WR[2],         ;Have all registers been checked
U 0D91, 2641,35 ;11244                 DT(LONG)&SET.ALU.CC
U 0D92, 88D7,31 ;11245                JMP.IF[NEQ] TO [T11.LOOP2]    ;If not continue checking
U 0D93, 3716,15 ;11246                MOV LS[#300] TO WR[0]
U 0D94, 90F6,15 ;11247                MISC2 [TRAP.ACC] WR[0]
U 0D95, FC12,15 ;11248                DEC LS[T9]                    ;Skip over ET6
U 0D96, FC12,15 ;11249                DEC LS[T9]
U 0D97, FC12,15 ;11250                DEC LS[T9]
U 0D98, FC12,15 ;11251                DEC LS[T9]
U 0D99, FC12,15 ;11252                DEC LS[T9]
U 0D9A, FC12,15 ;11253                DEC LS[T9]
U 0D9B, 0870,DC ;11254                JSR [-LO]                     ;Get address of STORE ET2
U 0D9C, BE18,15 ;11255                MOV WR[0] TO LS[T12]
U 0D9D, 0870,DC ;11256                JSR [-LO]                     ;Get address of SHIFT RIGHT ET2
U 0D9E, 3E10,15 ;11257                MOV WR[0] TO LS[T8]
U 0D9F, 0870,DC ;11258                JSR [-LO]                     ;Get address of MOV ET0 TO ET2
U 0DA0, 3E1C,15 ;11259                MOV WR[0] TO LS[T14]
U 0DA1, 8877,4C ;11260                JSR [SHIFT.RIGHT.8]           ;Shift the upper bits of ET2 right 8
U 0DA2, 0878,6C ;11261  T11.9:        JSR [STORE.DATA.2]            ;Store the data into LS[T13]
U 0DA3, B61A,15 ;11262                MOV LS[T13] TO WR[0]          ;Setup received data
U 0DA4, 369E,95 ;11263                MOV LS[ONES] TO WR[1]         ;Setup expected data
U 0DA5, 0869,3C ;11264                JSR [CHECK.RESULT]            ;Check for error
U 0DA6, 88DA,24 ;11265                JMP [T11.9]                   ;Loop on error
U 0DA7, 0878,AC ;11266  T11.A:        JSR [MOV.DATA]                ;Load ones into the upper bits of ET2
U 0DA8, 8877,4C ;11267                JSR [SHIFT.RIGHT.8]           ;Shift the upper bits of ET2 right 8
U 0DA9, 0878,6C ;11268                JSR [STORE.DATA.2]            ;Store the data into LS[T13]
U 0DAA, B61A,15 ;11269                MOV LS[T13] TO WR[0]          ;Set up received data
U 0DAB, B69C,95 ;11270                MOV LS[#0] TO WR[1]           ;Set up expected data
```

```
U ODAC, 0869,3C ;11271          JSR [CHECK.RESULT]              ;Check for error
U ODAD, 88DA,74 ;11272          JMP [T11.A]                     ;Loop on error
                ;11273  T11.END:
```

```
;11274  .PAGE
;11275  .TOC    "TEST 12 HUGE AND GRAND LOAD TEST(M8389 FPA MODULE)"
;11276  ;++
;11277  ;
;11278  ;
;11279  ; Test Description:
;11280  ;
;11281  ;       There are five types of data that can be loaded into the FPA from the
;11282  ;       CPU. They are integer, floating, double, grand, and huge. When an inte-
;11283  ;       ger, floating, or double word is loaded into the FPA the hidden bit is
;11284  ;       in bit 55 of the fraction data path or there is no hidden bit. When a
;11285  ;       grand word is loaded the hidden bit is in bit 52 of the fraction data
;11286  ;       path and when a huge word is loaded the hidden bit is in bit 48 of the
;11287  ;       fraction data path. The other thing that occurs during a 3rd huge load
;11288  ;       is that the extension data path is loaded. The purpose of this test is
;11289  ;       to check the loading of the hidden bit in the four different cases and
;11290  ;       to test the WR in the extension data path when doing a huge load. The
;11291  ;       data path to test the loading of the different types of words will be
;11292  ;       to do a grand load with zeros in all of the bits, then do a read of the
;11293  ;       fraction data path and look for the hidden bit. The same path is fol-
;11294  ;       lowed for a huge load. In testing the WR of the extension data
;11295  ;       path a third huge load will be executed which loads the even register
;11296  ;       of the extension data path and the odd register of fraction data path,
;11297  ;       bits 32 - 55. Since the WR of the extension data path are the only ones
;11298  ;       being tested all others will be zeros. The data patterns that will be
;11299  ;       used to test the WR are zeros, ones, a one shifted through a field of
;11300  ;       zeros, and a zero shifted through a field of ones.
;11301  ;
;11302  ; Assumptions:
;11303  ;
;11304  ;       It is assumed that all the WR in bits 0 - 55 of the fraction data path
;11305  ;       have been tested.
;11306  ;
;11307  ; Test Steps:
;11308  ;
;11309  ;       1) Issue a FORCE instruction to enable FPA bits 10 - 17 onto the
;11310  ;          instruction encode bits and the size bits to set the SIZE bits
;11311  ;          to a grand word.
;11312  ;       2) Issue a MISC instruction to force the address of the load routine to
;11313  ;          load zeros into the fraction data path to test for the hidden bit.
;11314  ;       3) Issue a MOV instruction to read the contents of the high fraction
;11315  ;          data path. If the hidden bit is not there or in the wrong position
;11316  ;          then issue error1, else continue.
;11317  ;       4) Issue a FORCE instruction to enable FPA bits 10 - 17 onto the
;11318  ;          instruction encode bits and the size bits to set the SIZE bits
;11319  ;          to a huge word.
;11320  ;       5) Issue a MISC instruction to load zeros into the fraction data path
;11321  ;          to test for the hidden bit.
;11322  ;       6) Issue a MOV instruction to read the contents of the high fraction
;11323  ;          data path back to the CPU. If the hidden bit is not there or in the
;11324  ;          wrong position, then issue error2, else continue.
;11325  ;       7) Issue a FORCE instruction to enable FPA bits 10 - 17 onto the
;11326  ;          instruction encode bits and the size bits to set the SIZE bits
;11327  ;          in the FPA to floating.
;11328  ;       8) Issue a MISC instruction to load zeros into the high fraction data
```

```
;11329  ;          path of the FPA and shift right.
;11330  ;       9) Issue a MOV instruction to read the contents of the high fraction
;11331  ;          data path back to the CPU. If not all zeros with bit 6 of the FPA
;11332  ;          BUS set then issue error3.
;11333  ;       10) Issue a MISC instruction to force the load routine for a HUGE load.
;11334  ;          This will cause bit 55 of the fraction data path to be set. The data
;11335  ;          that's loaded is zeros and the one occurs in bit 55 because of the
;11336  ;          hidden bit.
;11337  ;       11) Issue a MISC instruction to read the first huge read back. This
;11338  ;          should cause the LSB of the exponent data path to be stored in bit 7
;11339  ;          of the Y BUS. So if bit 7 of the FPA BUS is set then issue error 4.
;11340  ;       12) Issue a MISC instruciton to read the third huge read back. This
;11341  ;          should cause the MSB of the fraction data path to be loaded into bit
;11342  ;          7 of the Y BUS. So if Bit 7 is not set then issue error 5.
;11343  ;       13) Issue a MISC instruction to set the size bits to huge. Issue a MISC
;11344  ;          instruction to load zero's into the extension data path.
;11345  ;       14) Read the contents of the extension data path back to the CPU. If not
;11346  ;          all zeros then issue error6, else contine.
;11347  ;       15) Issue a MISC instruction to load the extension data path with ones.
;11348  ;       16) Read the contents of the extension data path back to the CPU. If not
;11349  ;          all ones then issue error7, else continue.
;11350  ;       17) Issue a MISC instruction to load a data pattern of a one floated
;11351  ;          across zero's or a zero floated across ones.
;11352  ;       18) Read the contents of the extension data path back to the CPU. If not
;11353  ;          the same as the pattern that was put in then issue error8.
;11354  ;
;11355  ; Error:
;11356  ;
;11357  ;       Error1 - Hidden bit failed on a grand word load.
;11358  ;       Errro2 - Hidden bit failed on a huge word load.
;11359  ;       Error3 - Hidden bit failed on a floating word load.
;11360  ;       Error4 - D07 failed unaserted when FRAC55 Y asserted and EXP0 Y
;11361  ;                unasserted.
;11362  ;       Error5 - D07 failed asserted when FRAC55 Y asserted and EXP0 Y
;11363  ;                unasserted.
;11364  ;       Error6 - Extension data path failed with zeros for data.
;11365  ;       Error7 - Extension data path failed with ones for data.
;11366  ;       Error8 - Extension data path failed with shifting patterns for data.
;11367  ;
;11368  ; Debug:
;11369  ;
;11370  ;       Error1: If the Hidden bit failed to show up or showed up in a place
;11371  ;               where it wasn't supposed to then the most likely source of
;11372  ;               error is the DATA IN CTL PAL.If the PAL isn't the cause of the
;11373  ;               error then inputs should be checked. If FPAE LO SEL 1 H is in
;11374  ;               error then the BRANCH 0 PAL should be checked for error. If
;11375  ;               FPAE LO SEL 0 H is in error then the EXT FUNC PAL should be
;11376  ;               checked for error. If the PAL's aren't the cause of the
;11377  ;               error and FPAA SIZE 1 H and FPAA SIZE 0 H aren't both zero or
;11378  ;               zero and one respectively then the IRD MUXES and the IRD ROM
;11379  ;               are likely to be sources of error. If there is no error in the
;11380  ;               LO SEL signals then it's possible that the etch between the
;11381  ;               transceivers and the DATA IN CTL PAL is shorted and thereby
;11382  ;               causing an error.
;11383  ;       Error2: If the Hidden bit failed to show up or showed up in a place
```

```
;11384  ;                where it wasn't supposed to then the most likely source of
;11385  ;                error is the DATA IN CTL PAL.If the PAL isn't the cause of the
;11386  ;                error then inputs should be checked. If FPAE LO SEL 1 H is in
;11387  ;                error then the BRANCH 0 PAL should be checked for error. If
;11388  ;                FPAE LO SEL 0 H is in error then the EXT FUNC PAL should be
;11389  ;                checked for error. If the PAL's aren't the cause of the
;11390  ;                error and FPAA SIZE 1 H and FPAA SIZE 0 H aren't both zero or
;11391  ;                zero and one respectively then the IRD MUXES and the IRD ROM
;11392  ;                should be checked for error. If there is no error in the LO SEL
;11393  ;                signals then it's possible that the etch between the tran-
;11394  ;                sceivers and the DATA IN CTL PAL is shorted and thereby causing
;11395  ;                an error.
;11396  ;        Error3: If the Hidden bit failed to show up or showed up in a place
;11397  ;                where it wasn't supposed to then the most likely source of
;11398  ;                error is the DATA IN CTL PAL or the fraction shift control.
;11399  ;                If the control bits are in error then the DEST PROM or the
;11400  ;                latch preceeding it could be in error or the latch in the lower
;11401  ;                left hand corner of page D could be in error. If the PAL isn't
;11402  ;                the cause of the error then inputs should be checked. If FPAE
;11403  ;                LO SEL 1 H is in error then the BRANCH 0 PAL should be checked
;11404  ;                for error. If FPAE LO SEL 0 H is in error then the EXT FUNC PAL
;11405  ;                should be checked for error. If the PAL's aren't the cause of
;11406  ;                the error and FPAA SIZE 1 H and FPAA SIZE 0 H aren't both zero
;11407  ;                or zero and one respectively then the IRD MUXES and the IRD
;11408  ;                ROM should be checked for error. If there is no error in the
;11409  ;                LO SEL signals then it's possible that the etch between the
;11410  ;                tranceivers and the DATA IN CTL PAL is shorted and thereby
;11411  ;                causing an error.
;11412  ;        Error4: If this error occurs then the IN ENB PAL should be checked for
;11413  ;                error. If the PAL is not in error then LOAD should be checked
;11414  ;                to be unasserted. If LOAD is in error then the EXT FUNC CTL PAL
;11415  ;                should be checked for error. If the PAL is not in error then
;11416  ;                the clock field should be checked to be not 100 and ext clk
;11417  ;                should not be asserted.
;11418  ;        Error5: Same as error 4 except that D07 should be asserted.
;11419  ;        Error6: If the Working Register test didn't fail then the control lines
;11420  ;                I0 - I6, the address lines, and the direct inputs have been
;11421  ;                tested. This leaves the I7 and I8 of the control lines and the
;11422  ;                2901's as the source of the error. If I7 and I8 aren't both
;11423  ;                ones then the PROM on page E should be checked. If the inputs
;11424  ;                FPAE FRAC I8 (1) H and FPAE FRAC I7 (1) H aren't both one then
;11425  ;                the PROM probably isn't the source of the error and the latch
;11426  ;                that they come from should be checked. It's also possible that
;11427  ;                since the extension part of the huge load has never been tested
;11428  ;                before that the addresssing logic associated the extension part
;11429  ;                of a huge load could be in error. If this were the case the out
;11430  ;                enable should be checked. This signal comes from the STORE CTL
;11431  ;                PAL. If the PAL is not in error then the latch the input
;11432  ;                signals came from should be checked for error. If the two
;11433  ;                control signals aren't in error then the 2901 is probably
;11434  ;                in error. The 2901 that's in error can be determined by the
;11435  ;                position of the bit in error.
;11436  ;        Error7: Same as error6.
;11437  ;        Error8: Same as error6.
;11438  ;
```

```
                ;11439  ;
                ;11440  ;--
                ;11441  T.12:
U 0DAE, B65E,15 ;11442          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;11443                                          ; STATUS WORD
U 0DAF, 3E80,15 ;11444          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;11445                                          ; BIT 15 INDICATES START OF TEST TO
                ;11446                                          ; CONSOLE
U 0DB0, 10E0,15 ;11447          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;11448  WAIT.T12.0:
U 0DB1, 08DB,14 ;11449          JMP [WAIT.T12.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;11450                                          ; RESPOND
U 0DB2, 087E,6C ;11451          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;11452                                          ; AND SIZE BITS
U 0DB3, B65E,15 ;11453          MOV LS[BIT15] TO WR[0]          ;Setup error mask
U 0DB4, 3E8A,15 ;11454          MOV WR[0] TO LS[ERROR.MASK]
U 0DB5, 37A4,15 ;11455          MOV LS[T12.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 0DB6, BE12,15 ;11456          MOV WR[0] TO LS[T9]
U 0DB7, 3716,15 ;11457  T12.1:  MOV LS[#300] TO WR[0]           ;Set the size bits to grand
U 0DB8, C762,15 ;11458          BIS LS[BIT17] TO WR[0]
U 0DB9, C764,15 ;11459          BIS LS[BIT18] TO WR[0]
U 0DBA, 90F6,15 ;11460          MISC2 [TRAP.ACC] WR[0]
U 0DBB, E5A2,15 ;11461          CLR LS[DATA.1]                  ;Load 0 into ET0 and FT0
U 0DBC, E5A4,15 ;11462          CLR LS[DATA.2]
U 0DBD, 65A6,15 ;11463          CLR LS[DATA.3]
U 0DBE, 8879,AC ;11464          JSR [LOAD.DOUBLE]
U 0DBF, 087B,5C ;11465          JSR [STORE.0.TRIPLE]            ;Store contents of ET0 and FT0
U 0DC0, B648,95 ;11466          MOV LS[BIT4] TO WR[1]           ;Setup received data
U 0DC1, B6A8,15 ;11467          MOV LS[FIRST.FLT] TO WR[0]      ;Setup expected data
U 0DC2, 0869,3C ;11468          JSR [CHECK.RESULT]              ;Check for error
U 0DC3, 08DB,74 ;11469          JMP [T12.1]                     ;Loop on error
U 0DC4, FF82,15 ;11470          INC LS[ERROR.NUMBER]            ;Go onto error 2
U 0DC5, 3716,15 ;11471  T12.2:  MOV LS[#300] TO WR[0]           ;Set size bits to huge
U 0DC6, 4760,15 ;11472          BIS LS[BIT16] TO WR[0]
U 0DC7, C762,15 ;11473          BIS LS[BIT17] TO WR[0]
U 0DC8, C764,15 ;11474          BIS LS[BIT18] TO WR[0]
U 0DC9, 90F6,15 ;11475          MISC2 [TRAP.ACC] WR[0]
U 0DCA, 8879,AC ;11476          JSR [LOAD.DOUBLE]               ;LOAD ET0 & FT0 with 0
U 0DCB, 087B,5C ;11477          JSR [STORE.0.TRIPLE]            ;Store contents of ET0 and FT0
U 0DCC, 3640,95 ;11478          MOV LS[BIT0] TO WR[1]           ;Setup expected data
U 0DCD, B6A8,15 ;11479          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 0DCE, 0869,3C ;11480          JSR [CHECK.RESULT]              ;Check for error
U 0DCF, 08DC,54 ;11481          JMP [T12.2]                     ;Loop on error
U 0DD0, FF82,15 ;11482          INC LS[ERROR.NUMBER]            ;Go on to error 3
U 0DD1, 3716,15 ;11483          MOV LS[#300] TO WR[0]           ;Set size bits to floating
U 0DD2, C764,15 ;11484          BIS LS[BIT18] TO WR[0]
U 0DD3, 4760,15 ;11485          BIS LS[BIT16] TO WR[0]
U 0DD4, 90F6,15 ;11486          MISC2 [TRAP.ACC] WR[0]
U 0DD5, 8870,3C ;11487          JSR [L0]                        ;Get address of shift right FT0
U 0DD6, BE14,15 ;11488          MOV WR[0] TO LS[T10]
U 0DD7, 8879,AC ;11489  T12.3:  JSR [LOAD.DOUBLE]               ;Load zeros into ET0 and FT0
U 0DD8, 3614,15 ;11490          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 0DD9, B716,95 ;11491          MOV LS[#300] TO WR[1]           ;Setup to loop on error
U 0DDA, 90F6,15 ;11492          MISC2 [TRAP.ACC] WR[0]          ;Force SHIFT RIGHT
U 0DDB, 10F6,95 ;11493          MISC2 [TRAP.ACC] WR[1]          ;Force loop on self
```

```
U 0DDC, 087B,5C ;11494          JSR [STORE.0.TRIPLE]          ;Store ET0 and FT0 to the CPU
U 0DDD, B6A8,15 ;11495          MOV LS[FIRST.FLT] TO WR[0]    ;Setup received data
U 0DDE, 364C,95 ;11496          MOV LS[BIT6] TO WR[1]         ;Setup expected data
U 0DDF, 0869,3C ;11497          JSR [CHECK.RESULT]            ;Check for error
U 0DE0, 08DD,74 ;11498          JMP [T12.3]                   ;Loop on error
U 0DE1, 3644,15 ;11499  T12.4:  MOV LS[#4] TO WR[0]           ;Set error # to 4
U 0DE2, BE82,15 ;11500          MOV WR[0] TO LS[ERROR.NUMBER]
U 0DE3, B65E,15 ;11501          MOV LS[BIT15] TO WR[0]        ;Mask out the sign bit
U 0DE4, 3E8A,15 ;11502          MOV WR[0] TO LS[ERROR.MASK]
U 0DE5, 0878,FC ;11503          JSR [LOAD.TRIPLE]             ;Load zeros into ET0 and FT0
U 0DE6, 087B,5C ;11504          JSR [STORE.0.TRIPLE]          ;Store ET0 and FT0 to the CPU
U 0DE7, B6A8,15 ;11505          MOV LS[FIRST.FLT] TO WR[0]    ;Setup received data
U 0DE8, 2F82,95 ;11506          CLR WR[1]                     ;Setup expected data
U 0DE9, 0869,3C ;11507          JSR [CHECK.RESULT]            ;Check for error
U 0DEA, 08DE,14 ;11508          JMP [T12.4]                   ;Loop on error
U 0DEB, FF82,15 ;11509          INC LS[ERROR.NUMBER]          ;Go on to error 5
U 0DEC, 371A,15 ;11510          MOV LS[#FFFF807F] TO WR[0]    ;Change the error mask
U 0DED, 3E8A,15 ;11511          MOV WR[0] TO LS[ERROR.MASK]
U 0DEE, 36AC,15 ;11512          MOV LS[THIRD.FLT] TO WR[0]    ;Setup received data
U 0DEF, B64E,95 ;11513          MOV LS[BIT7] TO WR[1]         ;Setup expected data
U 0DF0, 0869,3C ;11514          JSR [CHECK.RESULT]            ;Check for error
U 0DF1, 08DE,14 ;11515          JMP [T12.4]                   ;Loop on error
U 0DF2, FF82,15 ;11516          INC LS[ERROR.NUMBER]          ;Go onto error 6
U 0DF3, 3716,15 ;11517          MOV LS[#300] TO WR[0]         ;Set the size bits to huge
U 0DF4, C764,15 ;11518          BIS LS[BIT18] TO WR[0]
U 0DF5, 4760,15 ;11519          BIS LS[BIT16] TO WR[0]
U 0DF6, C762,15 ;11520          BIS LS[BIT17] TO WR[0]
U 0DF7, 90F6,15 ;11521          MISC2 [TRAP.ACC] WR[0]
U 0DF8, 0878,FC ;11522  T12.5:  JSR [LOAD.TRIPLE]             ;Load ET0 and FT0 with zero
U 0DF9, 087B,5C ;11523          JSR [STORE.0.TRIPLE]          ;Store ET0 and FT0 to CPU
U 0DFA, 36AC,15 ;11524          MOV LS[THIRD.FLT] TO WR[0]    ;Setup received data
U 0DFB, B69C,95 ;11525          MOV LS[#0] TO WR[1]           ;Setup expected data
U 0DFC, 0869,3C ;11526          JSR [CHECK.RESULT]            ;Check for error
U 0DFD, 88DF,84 ;11527          JMP [T12.5]                   ;Loop on error
U 0DFE, FF82,15 ;11528          INC LS[ERROR.NUMBER]          ;Set error number to 7
U 0DFF, B69E,15 ;11529          MOV LS[ONES] TO WR[0]         ;Set data to ONES
U 0E00, 3EA2,15 ;11530          MOV WR[0] TO LS[DATA.1]
U 0E01, 3EA4,15 ;11531          MOV WR[0] TO LS[DATA.2]
U 0E02, BEA6,15 ;11532          MOV WR[0] TO LS[DATA.3]
U 0E03, 0878,FC ;11533  T12.6:  JSR [LOAD.TRIPLE]             ;Load ones into ET0 and FT0
U 0E04, 087B,5C ;11534          JSR [STORE.0.TRIPLE]          ;Store ET0 and FT0 to the CPU
U 0E05, 36AC,15 ;11535          MOV LS[THIRD.FLT] TO WR[0]    ;Setup received data
U 0E06, 369E,95 ;11536          MOV LS[ONES] TO WR[1]         ;Setup expected data
U 0E07, 0869,3C ;11537          JSR [CHECK.RESULT]            ;Check for error
U 0E08, 08E0,34 ;11538          JMP [T12.6]                   ;Loop on error
U 0E09, FF82,15 ;11539          INC LS[ERROR.NUMBER]          ;Set error number to 8
U 0E0A, 3650,15 ;11540          MOV LS[BIT8] TO WR[0]         ;Setup for shift pattern through
U 0E0B, BEA6,15 ;11541  T12.8:  MOV WR[0] TO LS[DATA.3]       ; extension bits
U 0E0C, 0878,FC ;11542  T12.7:  JSR [LOAD.TRIPLE]             ;Load data into extension bits
U 0E0D, 087B,5C ;11543          JSR [STORE.0.TRIPLE]          ;Store extension bits to the CPU
U 0E0E, B6A6,95 ;11544          MOV LS[DATA.3] TO WR[1]       ;Setup expected data
U 0E0F, 36AC,15 ;11545          MOV LS[THIRD.FLT] TO WR[0]    ;Setup received data
U 0E10, 0869,3C ;11546          JSR [CHECK.RESULT]            ;Check for error
U 0E11, 08E0,C4 ;11547          JMP [T12.7]                   ;Loop on error
U 0E12, 36A6,15 ;11548          MOV LS[DATA.3] TO WR[0]       ;Shift data left one
```

```
U 0E13, 23D0,15 ;11549          SHL WR[0]
                ;11550          CMP WR[0] WITH LS[BIT16],       ;All bit postions checked
U 0E14, 4F60,35 ;11551           DT(LONG)&SET.ALU.CC
U 0E15, 88E0,B1 ;11552          JMP.IF[NEQ] TO [T12.8]          ;If not then continue checking
                ;11553  T12.END:
```

```
;11554  .PAGE
;11555  .TOC    "TEST 13 AND FUNCTION, A.Q SRC TEST(M8389 FPA MODULE)"
;11556  ;++
;11557  ;
;11558  ;
;11559  ; Test Description:
;11560  ;
;11561  ;       The purpose of this test is to check the AND function of the fraction
;11562  ;       data path.  The A.Q SRC logic is also tested in this test.  The expo-
;11563  ;       nent data path has no 'AND' function available and is not tested.  The
;11564  ;       data path used is to load WR 0 with a pattern, move this pattern to the
;11565  ;       Q register, load WR 0 with the second pattern, and execute an AND
;11566  ;       FWR[0] WITH FQ TO FWR[2].  The result is then checked by reading back
;11567  ;       (storeing) FWR[2] to the CPU.  The 2901 DST is WRITE.B which has been
;11568  ;       tested previously.
;11569  ;         Note that the AND micro-instruction used is a word in the FPA's
;11570  ;       normal u-code (i.e. not a diagnostic u-word).
;11571  ;
;11572  ; Assumptions:
;11573  ;
;11574  ;       It is assumed that all WR in all 2901's in the FPA and the LOAD, MOVE,
;11575  ;       and STORE routines are error free.
;11576  ;
;11577  ; Test Steps:
;11578  ;
;11579  ;       1) Load zeros into WR 0, and MOV the content of WR 0 into the Q reg.
;11580  ;       2) Issue a MISC instruction to force an AND of WR 0 and Q, the
;11581  ;          result of which will be in WR 2 and can be read back to the CPU
;11582  ;          to be checked. If not all zeros then issue error1.
;11583  ;       3) Load ones into WR 0, move WR 0 to Q, and then load zeros into WR 0.
;11584  ;       4) Issue a MISC instruction to force an AND of WR 0 and Q, the
;11585  ;          result of which will be in WR 2 and can be read back to the CPU
;11586  ;          to be checked. If not all zeros then issue error2.
;11587  ;       5) Load zeros into WR 0, move WR 0 to Q, and then load ones into WR 0.
;11588  ;       6) Issue a MISC instruction to force an AND of WR 0 and Q, the
;11589  ;          result of which will be in WR 2 and can be read back to the CPU
;11590  ;          to be checked. If not all zeros then issue error3.
;11591  ;       7) Load ones into WR 0 and mov to Q.
;11592  ;       8) Issue a MISC instruction to force an AND of WR 0 and Q, the
;11593  ;          result of which will be in WR 2 and can be read back to the CPU
;11594  ;          and be checked. If not all ones then issue error4.
;11595  ;       9) Load ones into WR 0 and move to Q.
;11596  ;       10)Load WR 0 with a shifting ones pattern.
;11597  ;       11)Issue a MISC instruction to force an AND of WR 0 and Q, the
;11598  ;          result of which will be in WR 2 and can be read back to the CPU
;11599  ;          and be checked. If the result is not the shifting pattern, issue
;11600  ;          error 5.
;11601  ;       12)Repeat steps 10 and 11 until 32 shifting patterns have been used.
;11602  ;       13)Repeat steps 9 through 12 with a shifting 0's pattern.
;11603  ;
;11604  ; Error:
;11605  ;
;11606  ;       Note:  For the following errors, the data printed under OTHER in the
;11607  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;11608  ;       FLT) which failed.
```

```
                ;11609  ;
                ;11610  ;       Error1 - AND instruction failed with zeros and zeros.
                ;11611  ;       Error2 - AND instruction failed with zeros in FWR 0 and ones in FQ.
                ;11612  ;       Error3 - AND instruction failed with ones in FWR 0 and zeros in FQ.
                ;11613  ;       Error4 - AND instruction failed with ones in both FWR 0 and FQ.
                ;11614  ;       Error5 - AND instruction failed with ones in FQ and shifting 1's in FWR.
                ;11615  ;       Error6 - AND instruction failed with ones in FQ and shifting 0's in FWR.
                ;11616  ;
                ;11617  ; Debug:
                ;11618  ;
                ;11619  ;       Error1: The possible sources of error are the control signals of the
                ;11620  ;               fraction data path 2901's or one of the 2901's itself.  The
                ;11621  ;               fraction control lines should be 011100000 (from I8 through I0
                ;11622  ;               respectively) during the 'AND' u-instruction.  Of the fraction
                ;11623  ;               control lines both the function (AND) and the SRC (A,Q) have
                ;11624  ;               not been previously tested.
                ;11625  ;                 If the control lines I3, I4, or I5 of the function control
                ;11626  ;               are in error, trace them back through the buffers to the
                ;11627  ;               control store ROM.  I3 comes through a MUX.  The select lines
                ;11628  ;               ENB MUL (1) L and ENB DIV (1) L should both be high.
                ;11629  ;                 If one of the other control lines is incorrect, trace it back
                ;11630  ;               in the same manner.
                ;11631  ;                 If the control lines are OK, suspect the 2901 which is out-
                ;11632  ;               putting the bad bit.
                ;11633  ;       Error2: If none of the control lines failed in the first test then the
                ;11634  ;               most likely cause of the error is the 2901 itself.
                ;11635  ;       Error3: Same as Error2.
                ;11636  ;       Error4: Same as Error2.
                ;11637  ;       Error5: Same as Error2.
                ;11638  ;       Error6: Same as Error2.
                ;11639  ;
                ;11640  ;
                ;11641  ;--
                ;11642
                ;11643  ;
                ;11644  ;       TEMPORARY LS LOCATIONS USED:
                ;11645  ;
                ;11646  ; T9     MAIN MEMORY POINTER
                ;11647  ; T10    ADDRESS OF 'AND FWR0 WITH FQ TO FWR2'
                ;11648  ; T11    EXPECTED DATA
                ;11649  ; T14    ADDRESS OF 'MOV FWR0 TO FQ'
                ;11650  ;
                ;11651  T.13:
                ;11652
U 0E16, B65E,15 ;11653          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;11654                                          ; STATUS WORD
U 0E17, 3E80,15 ;11655          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;11656                                          ; BIT 15 INDICATES START OF TEST TO
                ;11657                                          ; CONSOLE
U 0E18, 10E0,15 ;11658          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;11659  WAIT.T13.0:
U 0E19, 88E1,94 ;11660          JMP [WAIT.T13.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;11661                                          ; RESPOND
U 0E1A, 087E,6C ;11662          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;11663                                          ; AND SIZE BITS
```

```
U 0E1B, B670,15 ;11664          MOV LS[OTHER.DATA] TO WR[0]      ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;11665                                           ; OTHER DATA
U 0E1C, 3E80,15 ;11666          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 0E1D, 3738,15 ;11667          MOV LS[T13.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 0E1E, BE12,15 ;11668          MOV WR[0] TO LS[T9]              ;SET UP POINTER IN LS
U 0E1F, 8870,3C ;11669          JSR [LO]                         ;LOAD WR 0 WITH MEMORY DATA
U 0E20, 3E1C,15 ;11670          MOV WR[0] TO LS[T14]             ;PUT FPA ADDRESS IN LS FOR MOV SUBROUTINE
                ;11671                                           ; MOVE IS FROM FPA WR 0 TO FPA Q
U 0E21, 8870,3C ;11672          JSR [LO]                         ;LOAD WR 0 WITH MEMORY DATA
U 0E22, BE14,15 ;11673          MOV WR[0] TO LS[T10]             ;PUT FPA ADDRESS IN LS FOR 'AND' INSTRUCTION
                ;11674                                           ; FPA U-WORD IS FWR[0] AND FQ TO FWR[2]
                ;11675                                           ;              EWR[0] AND EQ TO EWR[2]
U 0E23, 3725,15 ;11676          MOV LS[#FFFF00FF] TO WR[2]       ;BITS 8-15 = 0
U 0E24, A0C5,15 ;11677          COM WR[2]                        ;BITS 8-15 = 1
U 0E25, C74F,15 ;11678          BIS LS[BIT7] TO WR[2]            ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;11679                                           ; BIT AND EXPONENT)
U 0E26, B725,95 ;11680          MOV LS[#FFFF00FF] TO WR[3]       ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;11681
U 0E27, E516,15 ;11682          CLR LS[T11]                      ;EXPECTED DATA STORED HERE (ALL 0'S)
U 0E28, 087C,1C ;11683          JSR [CLR.FTO]                    ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11684                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E29, 0878,AC ;11685          JSR [MOV.DATA]                   ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11686                                           ; EQ
U 0E2A, 88E6,3C ;11687          JSR [LOOP.T13.1.6]               ;TEST 'AND' FUNCTION WITH 0'S AND 0'S
                ;11688
U 0E2B, FF82,15 ;11689          INC LS[ERROR.NUMBER]             ;ERROR 2
U 0E2C, FDA2,15 ;11690          COM LS[DATA.1]                   ;ONES IN FIRST FLOAT
U 0E2D, FDA4,15 ;11691          COM LS[DATA.2]                   ;ONES IN SECOND FLOAT
U 0E2E, 7DA6,15 ;11692          COM LS[DATA.3]                   ;ONES IN THIRD (HUGE) FLOAT
U 0E2F, 0878,FC ;11693          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11694                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E30, 0878,AC ;11695          JSR [MOV.DATA]                   ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11696                                           ; EQ.  NOW Q CONTAINS 1'S.
U 0E31, 087C,1C ;11697          JSR [CLR.FTO]                    ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11698                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E32, 88E6,3C ;11699          JSR [LOOP.T13.1.6]               ;TEST 'AND' FUNCTION 0'S AND 1'S
                ;11700
U 0E33, FF82,15 ;11701          INC LS[ERROR.NUMBER]             ;ERROR 3
U 0E34, 0878,FC ;11702          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11703                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E35, 0878,AC ;11704          JSR [MOV.DATA]                   ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11705                                           ; EQ.  NOW Q CONTAINS 0'S
U 0E36, FDA2,15 ;11706          COM LS[DATA.1]                   ;ONES IN FIRST FLOAT
U 0E37, FDA4,15 ;11707          COM LS[DATA.2]                   ;ONES IN SECOND FLOAT
U 0E38, 7DA6,15 ;11708          COM LS[DATA.3]                   ;ONES IN THIRD (HUGE) FLOAT
U 0E39, 0878,FC ;11709          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11710                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E3A, 88E6,3C ;11711          JSR [LOOP.T13.1.6]               ;TEST 'AND' FUNCTION WITH 1'S AND 0'S
                ;11712
U 0E3B, FF82,15 ;11713          INC LS[ERROR.NUMBER]             ;ERROR 4
U 0E3C, FD16,15 ;11714          COM LS[T11]                      ;EXPECTED DATA = 1'S
U 0E3D, 0878,FC ;11715          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11716                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0E3E, 0878,AC ;11717          JSR [MOV.DATA]                   ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11718                                           ; EQ.  NOW Q CONTAINS 1'S.
```

```
U 0E3F, 88E6,3C ;11719          JSR [LOOP.T13.1.6]                  ;TEST 'AND' FUNCTION WITH 1'S AND 1'S
                ;11720
U 0E40, FF82,15 ;11721          INC LS[ERROR.NUMBER]                ;ERROR 5
U 0E41, 0878,FC ;11722          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11723                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0E42, 0878,AC ;11724          JSR [MOV.DATA]                      ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11725                                              ; EQ.  NOW Q CONTAINS 1'S.
U 0E43, E5F8,15 ;11726          CLR LS[OS]                          ;CLEAR INDEX
                ;11727  REPEAT.T13.5:
U 0E44, 36F6,15 ;11728          MOV LS[SHIFT.OS(4-0)] TO WR[0]      ;GET SHIFTING 1 DATA PATTERN FROM LS
U 0E45, 3E16,15 ;11729          MOV WR[0] TO LS[T11]                ;EXPECTED DATA
U 0E46, 3EA2,15 ;11730          MOV WR[0] TO LS[DATA.1]             ;FIRST FLOAT DATA
U 0E47, 3EA4,15 ;11731          MOV WR[0] TO LS[DATA.2]             ;SECOND FLOAT DATA
U 0E48, BEA6,15 ;11732          MOV WR[0] TO LS[DATA.3]             ;THIRD FLOAT DATA
U 0E49, 0878,FC ;11733          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11734                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0E4A, 88E6,3C ;11735          JSR [LOOP.T13.1.6]                  ;TEST 'AND' FUNCTION WITH SHIFTING 1'S
                ;11736                                              ; AND 1'S
U 0E4B, 7FF8,15 ;11737          INC LS[OS]                          ;POINT TO NEXT PATTERN
U 0E4C, B6F8,15 ;11738          MOV LS[OS] TO WR[0]                 ;GET VALUE OF INDEX
                ;11739          CMP LS[#20] WITH WR[0],             ;SEE IF DONE 32 PATTERNS YET
U 0E4D, 4E4A,35 ;11740           DT(LONG)&SET.ALU.CC                ; AND SET CONDITION CODES
U 0E4E, 08E4,41 ;11741          JMP.IF[NEQ] TO [REPEAT.T13.5]       ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;11742
U 0E4F, FF82,15 ;11743          INC LS[ERROR.NUMBER]                ;ERROR 6
U 0E50, B69E,15 ;11744          MOV LS[ONES] TO WR[0]               ;GET A PATTERN OF ONES
U 0E51, 3EA2,15 ;11745          MOV WR[0] TO LS[DATA.1]             ;LOAD ONES IN FIRST FLOAT
U 0E52, 3EA4,15 ;11746          MOV WR[0] TO LS[DATA.2]             ;LOAD ONES IN SECOND FLOAT
U 0E53, BEA6,15 ;11747          MOV WR[0] TO LS[DATA.3]             ;LOAD ONES IN THIRD FLOAT
U 0E54, 0878,FC ;11748          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11749                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0E55, 0878,AC ;11750          JSR [MOV.DATA]                      ;MOVE DATA FROM FWR[0] TO FQ, EWR[0] TO
                ;11751                                              ; EQ.  NOW Q CONTAINS 1'S.
U 0E56, E5F8,15 ;11752          CLR LS[OS]                          ;CLEAR INDEX
                ;11753  REPEAT.T13.6:
U 0E57, 5FF6,15 ;11754          MCOM LS[SHIFT.OS(4-0)] TO WR[0]     ;GET SHIFTING 0 DATA PATTERN FROM LS
U 0E58, 3E16,15 ;11755          MOV WR[0] TO LS[T11]                ;EXPECTED DATA
U 0E59, 3EA2,15 ;11756          MOV WR[0] TO LS[DATA.1]             ;FIRST FLOAT DATA
U 0E5A, 3EA4,15 ;11757          MOV WR[0] TO LS[DATA.2]             ;SECOND FLOAT DATA
U 0E5B, BEA6,15 ;11758          MOV WR[0] TO LS[DATA.3]             ;THIRD FLOAT DATA
U 0E5C, 0878,FC ;11759          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11760                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0E5D, 88E6,3C ;11761          JSR [LOOP.T13.1.6]                  ;TEST 'AND' FUNCTION WITH SHIFTING 0'S
                ;11762                                              ; AND 1'S
U 0E5E, 7FF8,15 ;11763          INC LS[OS]                          ;POINT TO NEXT PATTERN
U 0E5F, B6F8,15 ;11764          MOV LS[OS] TO WR[0]                 ;GET VALUE OF INDEX
                ;11765          CMP LS[#20] WITH WR[0],             ;SEE IF DONE 32 PATTERNS YET
U 0E60, 4E4A,35 ;11766           DT(LONG)&SET.ALU.CC                ; AND SET CONDITION CODES
U 0E61, 88E5,71 ;11767          JMP.IF[NEQ] TO [REPEAT.T13.6]       ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;11768
U 0E62, 88E7,C4 ;11769          JMP [END.T13]                       ;ELSE DONE WITH 'AND' TEST
                ;11770
                ;11771
                ;11772  LOOP.T13.1.6:
U 0E63, 6588,15 ;11773          CLR LS[ADDRESS.DATA]                ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
```

```
U 0E64, FF88,15 ;11774          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 0E65, 3614,15 ;11775          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'AND' INSTRUCTION
U 0E66, B716,95 ;11776          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 0E67, 90F6,15 ;11777          MISC2 [TRAP.ACC] WR[0]          ;FORCE FWR[0] WITH FQ TO FWR[2],
                ;11778                                          ; EWR[0] WITH EQ TO EWR[2]
U 0E68, 10F6,95 ;11779          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 0E69, 087A,1C ;11780          JSR [STORE.2.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;11781                                          ; AND THIRD.FLT
U 0E6A, BE8B,15 ;11782          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;11783                                          ; AND EXPONENT (BITS 7-14)
U 0E6B, B6A8,15 ;11784          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 0E6C, 3616,95 ;11785          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0E6D, 0869,3C ;11786          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0E6E, 08E6,34 ;11787          JMP [LOOP.T13.1.6]              ;ERROR LOOP IF ENABLED
                ;11788
U 0E6F, FF88,15 ;11789          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 0E70, E58A,15 ;11790          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 0E71, 36AA,15 ;11791          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 0E72, 3616,95 ;11792          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0E73, 0869,3C ;11793          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0E74, 08E6,34 ;11794          JMP [LOOP.T13.1.6]              ;ERROR LOOP IF ENABLED
                ;11795
U 0E75, FF88,15 ;11796          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 0E76, 3E8B,95 ;11797          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 0E77, 36AC,15 ;11798          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 0E78, 3616,95 ;11799          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0E79, 0869,3C ;11800          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0E7A, 08E6,34 ;11801          JMP [LOOP.T13.1.6]              ;ERROR LOOP IF ENABLED
U 0E7B, 5B00,14 ;11802          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;11803  END.T13:
```

```
;11804  .PAGE
;11805  .TOC    "TEST 14 R NOT AND S FUNCTION, A.B SRC TEST(M8389 FPA MODULE)"
;11806  ;++
;11807  ;
;11808  ;
;11809  ; Test Description:
;11810  ;
;11811  ;       This test checks the 'R NOT AND S' function of the fraction data path.
;11812  ;       The SRC A.B is also being checked for the first time. The exponent data
;11813  ;       path does not use this function, so the exponent bits will not be
;11814  ;       checked.  The data path used is to load FWR 0 with a data pattern and
;11815  ;       move the pattern into FRW INT.MASK and FRW 2.  The function used is a
;11816  ;       'AND NOT FWR[INT.MASK] WITH FWR[2]' with the result going to FWR[2].
;11817  ;       The 2901 SRC is A.B, and the DST is WRITE B.  Data patterns used are
;11818  ;       ones, zeros, shifted ones, and shifted zeros.  The DST is WRITE.B.
;11819  ;         Note that the AND NOT micro-instruction used is a word in the FPA's
;11820  ;       normal u-code (i.e. not a diagnostic u-word).
;11821  ;
;11822  ; Assumptions:
;11823  ;
;11824  ;       It is assumed that all previous tests have run successfully.
;11825  ;
;11826  ; Test Steps:
;11827  ;
;11828  ;       1) Load zeros into FWR 0, and MOV the contents of FWR 0 into FWR
;11829  ;          INT.MASK.  Then MOV the contents of FWR 0 to FWR 2.
;11830  ;       2) Issue a MISC instruction to force an AND of NOT FWR INT.MASK and
;11831  ;          FWR 2 the result of which will be in WR 2 and can be read back to
;11832  ;          the CPU to be checked. If not all zeros then issue error1.
;11833  ;       3) Load ones into FWR 0, move FWR 0 to FWR INT.MASK, and then load
;11834  ;          zeros into FWR 0.  MOV FWR 0 to FWR 2.
;11835  ;       4) Issue a MISC instruction to force an AND of NOT FWR INT.MASK and
;11836  ;          FWR 2, the result of which will be in WR 2 and can be read back to
;11837  ;          the CPU to be checked. If not all zeros then issue error2.
;11838  ;       5) Load zeros into FWR 0, move FWR 0 to FWR INT.MASK, and then load
;11839  ;          ones into FWR 0.  MOV FWR 0 to FWR 2.
;11840  ;       6) Issue a MISC instruction to force an AND of NOT FWR INT.MASK and FWR
;11841  ;          2, the result of which will be in WR 2 and can be read back to the
;11842  ;          CPU to be checked. If not all ones then issue error3.
;11843  ;       7) Load ones into FWR 0 and mov to FWR INT.MASK.  Then mov FWR 0 to FWR
;11844  ;          2.
;11845  ;       8) Issue a MISC instruction to force an AND of NOT FWR INT.MASK and FWR
;11846  ;          2, the result of which will be in WR 2 and can be read back to the
;11847  ;          CPU and be checked. If not all zeros then issue error4.
;11848  ;       9) Load zeros into FWR 0 and move to FWR INT.MASK.
;11849  ;       10)Load FWR 0 with a shifting ones pattern.  Mov FWR 0 to FWR 2.
;11850  ;       11)Issue a MISC instruction to force an AND of NOT FWR INT.MASK and
;11851  ;          FWR 2, the result of which will be in WR 2 and can be read back to
;11852  ;          the CPU and be checked. If the result is not the shifting pattern,
;11853  ;          issue error 5.
;11854  ;       12)Repeat steps 10 and 11 until 32 shifting patterns have been used.
;11855  ;       13)Repeat steps 9 through 12 with a shifting 0's pattern.
;11856  ;
;11857  ; Error:
;11858  ;
```

```
                ;11859  ;       Note:  For the following errors, the data printed under OTHER in the
                ;11860  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
                ;11861  ;       FLT) which failed.
                ;11862  ;
                ;11863  ;       Error1 - AND NOT FWR WITH FWR instruction failed with zeros and zeros.
                ;11864  ;       Error2 - AND NOT FWR WITH FWR instruction failed with ones in FWR
                ;11865  ;                INT.MASK and zeros in FWR 2.
                ;11866  ;       Error3 - AND NOT FWR WITH FWR instruction failed with zeros in FWR
                ;11867  ;                INT.MASK and ones in FWR 2.
                ;11868  ;       Error4 - AND NOT FWR WITH FWR instruction failed with ones in both FWR
                ;11869  ;                INT.MASK and FWR 2.
                ;11870  ;       Error5 - AND NOT FWR WITH FWR instruction failed with zeros in FWR
                ;11871  ;                INT.MASK and shifting 1's in FWR 2.
                ;11872  ;       Error6 - AND NOT FWR WITH FWR instruction failed with zeros in FWR
                ;11873  ;                INT.MASK and shifting 0's in FWR 2.
                ;11874  ;
                ;11875  ; Debug:
                ;11876  ;
                ;11877  ;       Error1: The possible sources of error are the control signals of the
                ;11878  ;               fraction data path 2901's or one of the 2901's itself.  The
                ;11879  ;               fraction control lines should be 011101001 (from I8 through I0
                ;11880  ;               respectively) during the 'AND NOT' u-instruction.  Of the frac-
                ;11881  ;               tion control lines both the FUNC (NOT R AND S) and the SRC
                ;11882  ;               (A,B) are being tested for the first time.
                ;11883  ;                 If the control lines I3, I4, or I5 of the function control
                ;11884  ;               are in error, trace them back through the buffers to the
                ;11885  ;               control store ROM.  I3 comes through a MUX.  The select lines
                ;11886  ;               ENB MUL (1) L and ENB DIV (1) L should both be high.
                ;11887  ;                 If one of the other control lines is incorrect, trace it back
                ;11888  ;               in the same manner.
                ;11889  ;                 If the control lines are OK, suspect the 2901 which is out-
                ;11890  ;               putting the bad bit.
                ;11891  ;       Error2: If there was no error in the first test then the most likely
                ;11892  ;               cause of the error is the 2901 itself.
                ;11893  ;       Error3: Same as Error2.
                ;11894  ;       Error4: Same as Error2.
                ;11895  ;       Error5: Same as Error2.
                ;11896  ;       Error6: Same as Error2.
                ;11897  ;
                ;11898  ;--
                ;11899
                ;11900  ;
                ;11901  ;       TEMPORARY LS LOCATIONS USED:
                ;11902  ;
                ;11903  ; T9    MAIN MEMORY POINTER
                ;11904  ; T10   ADDRESS OF 'AND NOT FWR[INT.MASK] WITH FWR2'
                ;11905  ; T11   EXPECTED DATA
                ;11906  ; T12   ADDRESS OF 'MOV FWR0 TO FWR2'
                ;11907  ; T13   ADDRESS OF 'MOV FWR0 TO  FWR[INT.MASK]'
                ;11908  ; T14   MOV SUBROUTINE USES T14 AS ADDRESS OF MOV ABOVE
                ;11909  ;
                ;11910  T.14:
                ;11911
U 0E7C, B65E,15 ;11912          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;11913                                          ; STATUS WORD
```

```
U 0E7D, 3E80,15 ;11914          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;11915                                         ; BIT 15 INDICATES START OF TEST TO
                ;11916                                         ; CONSOLE
U 0E7E, 10E0,15 ;11917          MISC [SET.CP.ATTN]             ;ISSUE CPU ATTN TO CONSOLE
                ;11918  WAIT.T14.0:
U 0E7F, 88E7,F4 ;11919          JMP [WAIT.T14.0]               ;LOOP HERE WAITING FOR CONSOLE TO
                ;11920                                         ; RESPOND
U 0E80, 087E,6C ;11921          JSR [SETUP]                    ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;11922                                         ; AND SIZE BITS
U 0E81, B670,15 ;11923          MOV LS[OTHER.DATA] TO WR[0]    ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;11924                                         ; OTHER DATA
U 0E82, 3E80,15 ;11925          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 0E83, B73A,15 ;11926          MOV LS[T14.POINTER] TO WR[0]   ;GET POINTER TO MAIN MEMORY ADDRESS
U 0E84, BE12,15 ;11927          MOV WR[0] TO LS[T9]            ;SET UP POINTER IN LS
U 0E85, 8870,3C ;11928          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0E86, BE18,15 ;11929          MOV WR[0] TO LS[T12]           ;STORE FPA ADDRESS IN LS
                ;11930                                         ; MOVE FPA WR 0 TO FPA WR 2
U 0E87, 8870,3C ;11931          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0E88, 3E1A,15 ;11932          MOV WR[0] TO LS[T13]           ;STORE FPA ADDRESS IN LS
                ;11933                                         ; MOVE FPA WR 0 TO FPA WR INT.MASK
U 0E89, 8870,3C ;11934          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0E8A, BE14,15 ;11935          MOV WR[0] TO LS[T10]           ;PUT FPA ADDRESS IN LS FOR 'AND' INSTRUCTION
                ;11936                                         ; FPA U-WORD IS 'AND NOT FWR[INT.MASK]
                ;11937                                         ; TO FWR2'
U 0E8B, 3725,15 ;11938          MOV LS[#FFFF00FF] TO WR[2]     ;BITS 8-15 = 0
U 0E8C, A0C5,15 ;11939          COM WR[2]                      ;BITS 8-15 = 1
U 0E8D, C74F,15 ;11940          BIS LS[BIT7] TO WR[2]          ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;11941                                         ; BIT AND EXPONENT)
U 0E8E, B725,95 ;11942          MOV LS[#FFFF00FF] TO WR[3]     ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;11943
U 0E8F, E516,15 ;11944          CLR LS[T11]                    ;EXPECTED DATA STORED HERE (ALL 0'S)
U 0E90, 087C,1C ;11945          JSR [CLR.FTO]                  ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11946                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0E91, B61A,15 ;11947          MOV LS[T13] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0E92, 3E1C,15 ;11948          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0E93, 0878,AC ;11949          JSR [MOV.DATA]                 ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
U 0E94, 3618,15 ;11950          MOV LS[T12] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR2
U 0E95, 3E1C,15 ;11951          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0E96, 08EE,0C ;11952          JSR [LOOP.T14.1.6]             ;TEST 'AND NOT' FUNCTION WITH 0'S AND 0'S
                ;11953
U 0E97, FF82,15 ;11954          INC LS[ERROR.NUMBER]           ;ERROR 2
U 0E98, FDA2,15 ;11955          COM LS[DATA.1]                 ;ONES IN FIRST FLOAT
U 0E99, FDA4,15 ;11956          COM LS[DATA.2]                 ;ONES IN SECOND FLOAT
U 0E9A, 7DA6,15 ;11957          COM LS[DATA.3]                 ;ONES IN THIRD (HUGE) FLOAT
U 0E9B, 0878,FC ;11958          JSR [LOAD.TRIPLE]              ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11959                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0E9C, B61A,15 ;11960          MOV LS[T13] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0E9D, 3E1C,15 ;11961          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0E9E, 0878,AC ;11962          JSR [MOV.DATA]                 ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
U 0E9F, 087C,1C ;11963          JSR [CLR.FTO]                  ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11964                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0EA0, 3618,15 ;11965          MOV LS[T12] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR2
U 0EA1, 3E1C,15 ;11966          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0EA2, 08EE,0C ;11967          JSR [LOOP.T14.1.6]             ;TEST 'AND NOT' FUNCTION WITH 1'S AND 0'S
                ;11968
```

```
U 0EA3, FF82,15 ;11969          INC LS[ERROR.NUMBER]              ;ERROR 3
U 0EA4, FD16,15 ;11970          COM LS[T11]                       ;EXPECTED DATA (1'S)
U 0EA5, 0878,FC ;11971          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11972                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0EA6, B61A,15 ;11973          MOV LS[T13] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0EA7, 3E1C,15 ;11974          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EA8, 0878,AC ;11975          JSR [MOV.DATA]                    ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                ;11976                                            ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0EA9, FDA2,15 ;11977          COM LS[DATA.1]                    ;ONES IN FIRST FLOAT
U 0EAA, FDA4,15 ;11978          COM LS[DATA.2]                    ;ONES IN SECOND FLOAT
U 0EAB, 7DA6,15 ;11979          COM LS[DATA.3]                    ;ONES IN THIRD (HUGE) FLOAT
U 0EAC, 0878,FC ;11980          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11981                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0EAD, 3618,15 ;11982          MOV LS[T12] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR2
U 0EAE, 3E1C,15 ;11983          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EAF, 08EE,0C ;11984          JSR [LOOP.T14.1.6]                ;TEST 'AND NOT' FUNCTION WITH 0'S AND 1'S
                ;11985
U 0EB0, FF82,15 ;11986          INC LS[ERROR.NUMBER]              ;ERROR 4
U 0EB1, FD16,15 ;11987          COM LS[T11]                       ;EXPECTED DATA = 0'S
U 0EB2, 0878,FC ;11988          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;11989                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0EB3, B61A,15 ;11990          MOV LS[T13] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0EB4, 3E1C,15 ;11991          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EB5, 0878,AC ;11992          JSR [MOV.DATA]                    ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                ;11993                                            ; NOW FWR[INT.MASK] CONTAINS 1'S
U 0EB6, 3618,15 ;11994          MOV LS[T12] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR2
U 0EB7, 3E1C,15 ;11995          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EB8, 08EE,0C ;11996          JSR [LOOP.T14.1.6]                ;TEST 'AND NOT' FUNCTION WITH 1'S AND 1'S
                ;11997
U 0EB9, FF82,15 ;11998          INC LS[ERROR.NUMBER]              ;ERROR 5
U 0EBA, 087C,1C ;11999          JSR [CLR.FT0]                     ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12000                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0EBB, B61A,15 ;12001          MOV LS[T13] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0EBC, 3E1C,15 ;12002          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EBD, 0878,AC ;12003          JSR [MOV.DATA]                    ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                ;12004                                            ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0EBE, E5F8,15 ;12005          CLR LS[OS]                        ;CLEAR INDEX
                ;12006  REPEAT.T14.5:
U 0EBF, 36F6,15 ;12007          MOV LS[SHIFT.OS(4-0)] TO WR[0]    ;GET SHIFTING 1 DATA PATTERN FROM LS
U 0EC0, 3E16,15 ;12008          MOV WR[0] TO LS[T11]              ;EXPECTED DATA
U 0EC1, 3EA2,15 ;12009          MOV WR[0] TO LS[DATA.1]           ;FIRST FLOAT DATA
U 0EC2, 3EA4,15 ;12010          MOV WR[0] TO LS[DATA.2]           ;SECOND FLOAT DATA
U 0EC3, BEA6,15 ;12011          MOV WR[0] TO LS[DATA.3]           ;THIRD FLOAT DATA
U 0EC4, 0878,FC ;12012          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12013                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0EC5, 3618,15 ;12014          MOV LS[T12] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR2
U 0EC6, 3E1C,15 ;12015          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EC7, 08EE,0C ;12016          JSR [LOOP.T14.1.6]                ;TEST 'AND NOT' FUNCTION WITH SHIFTING
                ;12017                                            ; 1'S AND ALL 0'S (0'S IN INT.MASK)
U 0EC8, 7FF8,15 ;12018          INC LS[OS]                        ;POINT TO NEXT PATTERN
U 0EC9, B6F8,15 ;12019          MOV LS[OS] TO WR[0]               ;GET VALUE OF INDEX
                ;12020          CMP LS[#20] WITH WR[0],           ;SEE IF DONE 32 PATTERNS YET
U 0ECA, 4E4A,35 ;12021           DT(LONG)&SET.ALU.CC              ; AND SET CONDITION CODES
U 0ECB, 88EB,F1 ;12022          JMP.IF[NEQ] TO [REPEAT.T14.5]     ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;12023
```

```
U 0ECC, FF82,15 ;12024          INC LS[ERROR.NUMBER]              ;ERROR 6
U 0ECD, 087C,1C ;12025          JSR [CLR.FTO]                     ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12026                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0ECE, B61A,15 ;12027          MOV LS[T13] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0ECF, 3E1C,15 ;12028          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0ED0, 0878,AC ;12029          JSR [MOV.DATA]                    ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                ;12030                                            ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0ED1, E5F8,15 ;12031          CLR LS[OS]                        ;CLEAR INDEX
                ;12032  REPEAT.T14.6:
U 0ED2, 5FF6,15 ;12033          MCOM LS[SHIFT.OS(4-0)] TO WR[0]   ;GET SHIFTING 0 DATA PATTERN FROM LS
U 0ED3, 3E16,15 ;12034          MOV WR[0] TO LS[T11]              ;EXPECTED DATA
U 0ED4, 3EA2,15 ;12035          MOV WR[0] TO LS[DATA.1]           ;FIRST FLOAT DATA
U 0ED5, 3EA4,15 ;12036          MOV WR[0] TO LS[DATA.2]           ;SECOND FLOAT DATA
U 0ED6, BEA6,15 ;12037          MOV WR[0] TO LS[DATA.3]           ;THIRD FLOAT DATA
U 0ED7, 0878,FC ;12038          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12039                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0ED8, 3618,15 ;12040          MOV LS[T12] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR2
U 0ED9, 3E1C,15 ;12041          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0EDA, 08EE,0C ;12042          JSR [LOOP.T14.1.6]                ;TEST 'AND NOT' FUNCTION WITH SHIFTING
                ;12043                                            ; 0'S AND ALL 0'S (0'S IN INT.MASK)
U 0EDB, 7FF8,15 ;12044          INC LS[OS]                        ;POINT TO NEXT PATTERN
U 0EDC, B6F8,15 ;12045          MOV LS[OS] TO WR[0]               ;GET VALUE OF INDEX
                ;12046          CMP LS[#20] WITH WR[0],           ;SEE IF DONE 32 PATTERNS YET
U 0EDD, 4E4A,35 ;12047           DT(LONG)&SET.ALU.CC              ; AND SET CONDITION CODES
U 0EDE, 08ED,21 ;12048          JMP.IF[NEQ] TO [REPEAT.T14.6]     ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;12049
U 0EDF, 08EF,A4 ;12050          JMP [END.T14]                     ;ELSE DONE WITH 'AND' TEST
                ;12051
                ;12052
                ;12053  LOOP.T14.1.6:
U 0EE0, 6588,15 ;12054          CLR LS[ADDRESS.DATA]              ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 0EE1, FF88,15 ;12055          INC LS[ADDRESS.DATA]              ;1 INDICATES FIRST FLOAT FAILED
U 0EE2, 0878,AC ;12056          JSR [MOV.DATA]                    ;MOV DATA FROM FWR 0 TO FWR 2
U 0EE3, 3614,15 ;12057          MOV LS[T10] TO WR[0]              ;GET ADDRESS OF FPA 'AND NOT' INSTRUCTION
U 0EE4, B716,95 ;12058          MOV LS[#300] TO WR[1]             ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 0EE5, 90F6,15 ;12059          MISC2 [TRAP.ACC] WR[0]            ;FORCE FWR[0] WITH FQ TO FWR[2],
                ;12060                                            ; EWR[0] WITH EQ TO EWR[2]
U 0EE6, 10F6,95 ;12061          MISC2 [TRAP.ACC] WR[1]            ;FORCE LOOP SELF
U 0EE7, 087A,1C ;12062          JSR [STORE.2.TRIPLE]              ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;12063                                            ; AND THIRD.FLT
U 0EE8, BE8B,15 ;12064          MOV WR[2] TO LS[ERROR.MASK]       ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;12065                                            ; AND EXPONENT (BITS 7-14)
U 0EE9, B6A8,15 ;12066          MOV LS[FIRST.FLT] TO WR[0]        ;PUT FIRST FLT RESULT IN WR 0
U 0EEA, 3616,95 ;12067          MOV LS[T11] TO WR[1]              ;EXPECTED DATA
U 0EEB, 0869,3C ;12068          JSR [CHECK.RESULT]                ;CHECK FOR ERRORS
U 0EEC, 88EE,04 ;12069          JMP [LOOP.T14.1.6]                ;ERROR LOOP IF ENABLED
                ;12070
U 0EED, FF88,15 ;12071          INC LS[ADDRESS.DATA]              ;INC PRINTOUT UNDER OTHER DATA TO 2
U 0EEE, E58A,15 ;12072          CLR LS[ERROR.MASK]                ;CHECK ALL BITS
U 0EEF, 36AA,15 ;12073          MOV LS[SEC.FLT] TO WR[0]          ;PUT SECOND FLT RESULT IN WR 0
U 0EF0, 3616,95 ;12074          MOV LS[T11] TO WR[1]              ;EXPECTED DATA
U 0EF1, 0869,3C ;12075          JSR [CHECK.RESULT]                ;CHECK FOR ERRORS
U 0EF2, 88EE,04 ;12076          JMP [LOOP.T14.1.6]                ;ERROR LOOP IF ENABLED
                ;12077
U 0EF3, FF88,15 ;12078          INC LS[ADDRESS.DATA]              ;INC PRINTOUT UNDER OTHER DATA TO 3
```

```
U 0EF4, 3E8B,95 ;12079          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 0EF5, 36AC,15 ;12080          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 0EF6, 3616,95 ;12081          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0EF7, 0869,3C ;12082          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0EF8, 88EE,04 ;12083          JMP [LOOP.T14.1.6]              ;ERROR LOOP IF ENABLED
U 0EF9, 5B00,14 ;12084          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;12085  END.T14:
```

```
;12086  .PAGE
;12087  .TOC    "TEST 15 R XOR S FUNCTION TEST(M8389 FPA MODULE)"
;12088  ;++
;12089  ;
;12090  ;
;12091  ; Test Description:
;12092  ;
;12093  ;       This test checks the 'XOR' function of the fraction data path.  The
;12094  ;       exponent data path does not use this function, so the exponent bits
;12095  ;       will not be checked.  The data path used is to load FWR 0 with a data
;12096  ;       pattern and move the pattern into FRW INT.MASK and FRW 2.  The function
;12097  ;       used is a 'XOR FWR[INT.MASK] WITH FWR[2]' with the result going to
;12098  ;       FWR[2].  The 2901 SRC is A.B, and the DST is WRITE B.  Data patterns
;12099  ;       used are ones, zeros, shifted ones, and shifted zeros.  The DST is
;12100  ;       WRITE.B.
;12101  ;         Note that the 'XOR' micro-instruction used is a word in the FPA's
;12102  ;       normal u-code (i.e. not a diagnostic u-word).
;12103  ;
;12104  ; Assumptions:
;12105  ;
;12106  ;       It is assumed that all previous tests have run successfully.
;12107  ;
;12108  ; Test Steps:
;12109  ;
;12110  ;       1) Load zeros into FWR 0, and MOV the contents of FWR 0 into FWR
;12111  ;          INT.MASK.  Then MOV the contents of FWR 0 to FWR 2.
;12112  ;       2) Issue a MISC instruction to force an XOR of FWR INT.MASK and
;12113  ;          FWR 2 the result of which will be in WR 2 and can be read back to
;12114  ;          the CPU to be checked. If not all zeros then issue error1.
;12115  ;       3) Load ones into FWR 0, move FWR 0 to FWR INT.MASK, and then load
;12116  ;          zeros into FWR 0.  MOV FWR 0 to FWR 2.
;12117  ;       4) Issue a MISC instruction to force an XOR of FWR INT.MASK and
;12118  ;          FWR 2, the result of which will be in WR 2 and can be read back to
;12119  ;          the CPU to be checked. If not all ones then issue error2.
;12120  ;       5) Load zeros into FWR 0, move FWR 0 to FWR INT.MASK, and then load
;12121  ;          ones into FWR 0.  MOV FWR 0 to FWR 2.
;12122  ;       6) Issue a MISC instruction to force an XOR of FWR INT.MASK and FWR
;12123  ;          2, the result of which will be in WR 2 and can be read back to the
;12124  ;          CPU to be checked. If not all ones then issue error3.
;12125  ;       7) Load ones into FWR 0 and mov to FWR INT.MASK.  Then mov FWR 0 to FWR
;12126  ;          2.
;12127  ;       8) Issue a MISC instruction to force an XOR of FWR INT.MASK and FWR
;12128  ;          2, the result of which will be in WR 2 and can be read back to the
;12129  ;          CPU and be checked. If not all zeros then issue error4.
;12130  ;       9) Load zeros into FWR 0 and move to FWR INT.MASK.
;12131  ;       10)Load FWR 0 with a shifting ones pattern.  Mov FWR 0 to FWR 2.
;12132  ;       11)Issue a MISC instruction to force an XOR of FWR INT.MASK and
;12133  ;          FWR 2, the result of which will be in WR 2 and can be read back to
;12134  ;          the CPU and be checked. If the result is not the shifting pattern,
;12135  ;          issue error 5.
;12136  ;       12)Repeat steps 10 and 11 until 32 shifting patterns have been used.
;12137  ;       13)Repeat steps 9 through 12 with a shifting 0's pattern.
;12138  ;
;12139  ; Error:
;12140  ;
```

```
                ;12141  ;       Note:  For the following errors, the data printed under OTHER in the
                ;12142  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
                ;12143  ;       FLT) which failed.
                ;12144  ;
                ;12145  ;       Error1 - XOR FWR WITH FWR instruction failed with zeros and zeros.
                ;12146  ;       Error2 - XOR FWR WITH FWR instruction failed with ones in FWR
                ;12147  ;                INT.MASK and zeros in FWR 2.
                ;12148  ;       Error3 - XOR FWR WITH FWR instruction failed with zeros in FWR
                ;12149  ;                INT.MASK and ones in FWR 2.
                ;12150  ;       Error4 - XOR FWR WITH FWR instruction failed with ones in both FWR
                ;12151  ;                INT.MASK and FWR 2.
                ;12152  ;       Error5 - XOR FWR WITH FWR instruction failed with zeros in FWR
                ;12153  ;                INT.MASK and shifting 1's in FWR 2.
                ;12154  ;       Error6 - XOR FWR WITH FWR instruction failed with zeros in FWR
                ;12155  ;                INT.MASK and shifting 0's in FWR 2.
                ;12156  ;
                ;12157  ; Debug:
                ;12158  ;
                ;12159  ;       Error1: The possible sources of error are the control signals of the
                ;12160  ;               fraction data path 2901's or one of the 2901's itself.  The
                ;12161  ;               fraction control lines should be 011110001 (from I8 through I0
                ;12162  ;               respectively) during the 'XOR' u-instruction.  Of the frac-
                ;12163  ;               tion control lines only the function (R XOR S) is being tested
                ;12164  ;               for the first time.
                ;12165  ;                 If the control lines I3, I4, or I5 of the function control
                ;12166  ;               are in error, trace them back through the buffers to the
                ;12167  ;               control store ROM.  I3 comes through a MUX.  The select lines
                ;12168  ;               ENB MUL (1) L and ENB DIV (1) L should both be high.
                ;12169  ;                 If one of the other control lines is incorrect, trace it back
                ;12170  ;               in the same manner.
                ;12171  ;                 If the control lines are OK, suspect the 2901 which is out-
                ;12172  ;               putting the bad bit.
                ;12173  ;       Error2: If there was no error in the first test then the most likely
                ;12174  ;               cause of the error is the 2901 itself.
                ;12175  ;       Error3: Same as Error2.
                ;12176  ;       Error4: Same as Error2.
                ;12177  ;       Error5: Same as Error2.
                ;12178  ;       Error6: Same as Error2.
                ;12179  ;
                ;12180  ;--
                ;12181
                ;12182  ;
                ;12183  ;       TEMPORARY LS LOCATIONS USED:
                ;12184  ;
                ;12185  ; T9    MAIN MEMORY POINTER
                ;12186  ; T10   ADDRESS OF 'XOR FWR[INT.MASK] WITH FWR[FT2]'
                ;12187  ; T11   EXPECTED DATA
                ;12188  ; T12   ADDRESS OF 'MOV FWR0 TO FWR2'
                ;12189  ; T13   ADDRESS OF 'MOV FWR0 TO  FWR[INT.MASK]'
                ;12190  ; T14   MOV SUBROUTINE USES T14 AS ADDRESS OF MOV ABOVE
                ;12191  ;
                ;12192  T.15:
                ;12193
U 0EFA, B65E,15 ;12194          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;12195                                          ; STATUS WORD
```

```
U 0EFB, 3E80,15 ;12196          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;12197                                         ; BIT 15 INDICATES START OF TEST TO
                ;12198                                         ; CONSOLE
U 0EFC, 10E0,15 ;12199          MISC [SET.CP.ATTN]             ;ISSUE CPU ATTN TO CONSOLE
                ;12200  WAIT.T15.0:
U 0EFD, 88EF,D4 ;12201          JMP [WAIT.T15.0]               ;LOOP HERE WAITING FOR CONSOLE TO
                ;12202                                         ; RESPOND
U 0EFE, 087E,6C ;12203          JSR [SETUP]                    ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;12204                                         ; AND SIZE BITS
U 0EFF, B670,15 ;12205          MOV LS[OTHER.DATA] TO WR[0]    ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;12206                                         ; OTHER DATA
U 0F00, 3E80,15 ;12207          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 0F01, B7C0,15 ;12208          MOV LS[T15.POINTER] TO WR[0]   ;GET POINTER TO MAIN MEMORY ADDRESS
U 0F02, BE12,15 ;12209          MOV WR[0] TO LS[T9]            ;SET UP POINTER IN LS
U 0F03, 8870,3C ;12210          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0F04, BE18,15 ;12211          MOV WR[0] TO LS[T12]           ;STORE FPA ADDRESS IN LS
                ;12212                                         ; MOVE FPA WR 0 TO FPA WR 2
U 0F05, 8870,3C ;12213          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0F06, 3E1A,15 ;12214          MOV WR[0] TO LS[T13]           ;STORE FPA ADDRESS IN LS
                ;12215                                         ; MOVE FPA WR 0 TO FPA WR INT.MASK
U 0F07, 8870,3C ;12216          JSR [LO]                       ;LOAD WR 0 WITH MEMORY DATA
U 0F08, BE14,15 ;12217          MOV WR[0] TO LS[T10]           ;PUT FPA ADDRESS IN LS FOR 'XOR' INSTRUCTION
                ;12218                                         ; FPA U-WORD IS 'XOR FWR[INT.MASK] WITH
                ;12219                                         ; FWR2'
U 0F09, 3725,15 ;12220          MOV LS[#FFFF00FF] TO WR[2]     ;BITS 8-15 = 0
U 0F0A, A0C5,15 ;12221          COM WR[2]                      ;BITS 8-15 = 1
U 0F0B, C74F,15 ;12222          BIS LS[BIT7] TO WR[2]          ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;12223                                         ; BIT AND EXPONENT)
U 0F0C, B725,95 ;12224          MOV LS[#FFFF00FF] TO WR[3]     ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;12225
U 0F0D, E516,15 ;12226          CLR LS[T11]                    ;EXPECTED DATA STORED HERE (ALL 0'S)
U 0F0E, 087C,1C ;12227          JSR [CLR.FT0]                  ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12228                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0F0F, B61A,15 ;12229          MOV LS[T13] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F10, 3E1C,15 ;12230          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0F11, 0878,AC ;12231          JSR [MOV.DATA]                 ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
U 0F12, 3618,15 ;12232          MOV LS[T12] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR2
U 0F13, 3E1C,15 ;12233          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0F14, 88F5,EC ;12234          JSR [LOOP.T15.1.6]             ;TEST 'XOR' FUNCTION WITH 0'S AND 0'S
                ;12235
U 0F15, FF82,15 ;12236          INC LS[ERROR.NUMBER]           ;ERROR 2
U 0F16, FD16,15 ;12237          COM LS[T11]                    ;EXPECTED DATA (1'S)
U 0F17, FDA2,15 ;12238          COM LS[DATA.1]                 ;ONES IN FIRST FLOAT
U 0F18, FDA4,15 ;12239          COM LS[DATA.2]                 ;ONES IN SECOND FLOAT
U 0F19, 7DA6,15 ;12240          COM LS[DATA.3]                 ;ONES IN THIRD (HUGE) FLOAT
U 0F1A, 0878,FC ;12241          JSR [LOAD.TRIPLE]              ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12242                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0F1B, B61A,15 ;12243          MOV LS[T13] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F1C, 3E1C,15 ;12244          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0F1D, 0878,AC ;12245          JSR [MOV.DATA]                 ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
U 0F1E, 087C,1C ;12246          JSR [CLR.FT0]                  ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12247                                         ; 8 EXT BITS IN WR 0 OF FPA
U 0F1F, 3618,15 ;12248          MOV LS[T12] TO WR[0]           ;ADDRESS OF MOV FWR0 TO FWR2
U 0F20, 3E1C,15 ;12249          MOV WR[0] TO LS[T14]           ;SET UP LS FOR MOV SUBROUTINE
U 0F21, 88F5,EC ;12250          JSR [LOOP.T15.1.6]             ;TEST 'XOR' FUNCTION WITH 1'S AND 0'S
```

```
                    ;12251
U 0F22, FF82,15     ;12252          INC LS[ERROR.NUMBER]            ;ERROR 3
U 0F23, 0878,FC     ;12253          JSR [LOAD.TRIPLE]               ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                    ;12254                                          ; 8 EXT BITS IN WR 0 OF FPA
U 0F24, B61A,15     ;12255          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F25, 3E1C,15     ;12256          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F26, 0878,AC     ;12257          JSR [MOV.DATA]                  ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                    ;12258                                          ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0F27, FDA2,15     ;12259          COM LS[DATA.1]                  ;ONES IN FIRST FLOAT
U 0F28, FDA4,15     ;12260          COM LS[DATA.2]                  ;ONES IN SECOND FLOAT
U 0F29, 7DA6,15     ;12261          COM LS[DATA.3]                  ;ONES IN THIRD (HUGE) FLOAT
U 0F2A, 0878,FC     ;12262          JSR [LOAD.TRIPLE]               ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                    ;12263                                          ; 8 EXT BITS IN WR 0 OF FPA
U 0F2B, 3618,15     ;12264          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR2
U 0F2C, 3E1C,15     ;12265          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F2D, 88F5,EC     ;12266          JSR [LOOP.T15.1.6]              ;TEST 'XOR' FUNCTION WITH 0'S AND 1'S
                    ;12267
U 0F2E, FF82,15     ;12268          INC LS[ERROR.NUMBER]            ;ERROR 4
U 0F2F, FD16,15     ;12269          COM LS[T11]                     ;EXPECTED DATA = 0'S
U 0F30, 0878,FC     ;12270          JSR [LOAD.TRIPLE]               ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                    ;12271                                          ; 8 EXT BITS IN WR 0 OF FPA
U 0F31, B61A,15     ;12272          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F32, 3E1C,15     ;12273          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F33, 0878,AC     ;12274          JSR [MOV.DATA]                  ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                    ;12275                                          ; NOW FWR[INT.MASK] CONTAINS 1'S
U 0F34, 3618,15     ;12276          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR2
U 0F35, 3E1C,15     ;12277          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F36, 88F5,EC     ;12278          JSR [LOOP.T15.1.6]              ;TEST 'XOR' FUNCTION WITH 1'S AND 1'S
                    ;12279
U 0F37, FF82,15     ;12280          INC LS[ERROR.NUMBER]            ;ERROR 5
U 0F38, 087C,1C     ;12281          JSR [CLR.FT0]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                    ;12282                                          ; 8 EXT BITS IN WR 0 OF FPA
U 0F39, B61A,15     ;12283          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F3A, 3E1C,15     ;12284          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F3B, 0878,AC     ;12285          JSR [MOV.DATA]                  ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                    ;12286                                          ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0F3C, E5F8,15     ;12287          CLR LS[OS]                      ;CLEAR INDEX
                    ;12288  REPEAT.T15.5:
U 0F3D, 36F6,15     ;12289          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;GET SHIFTING 1 DATA PATTERN FROM LS
U 0F3E, 3E16,15     ;12290          MOV WR[0] TO LS[T11]            ;EXPECTED DATA
U 0F3F, 3EA2,15     ;12291          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 0F40, 3EA4,15     ;12292          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 0F41, BEA6,15     ;12293          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 0F42, 0878,FC     ;12294          JSR [LOAD.TRIPLE]               ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                    ;12295                                          ; 8 EXT BITS IN WR 0 OF FPA
U 0F43, 3618,15     ;12296          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV FWR0 TO FWR2
U 0F44, 3E1C,15     ;12297          MOV WR[0] TO LS[T14]            ;SET UP LS FOR MOV SUBROUTINE
U 0F45, 88F5,EC     ;12298          JSR [LOOP.T15.1.6]              ;TEST 'XOR' FUNCTION WITH SHIFTING
                    ;12299                                          ; 1'S AND ALL 0'S (0'S IN INT.MASK)
U 0F46, 7FF8,15     ;12300          INC LS[OS]                      ;POINT TO NEXT PATTERN
U 0F47, B6F8,15     ;12301          MOV LS[OS] TO WR[0]             ;GET VALUE OF INDEX
                    ;12302          CMP LS[#20] WITH WR[0],         ;SEE IF DONE 32 PATTERNS YET
U 0F48, 4E4A,35     ;12303           DT(LONG)&SET.ALU.CC            ; AND SET CONDITION CODES
U 0F49, 08F3,D1     ;12304          JMP.IF[NEQ] TO [REPEAT.T15.5]   ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                    ;12305
```

```
U 0F4A, FF82,15 ;12306          INC LS[ERROR.NUMBER]              ;ERROR 6
U 0F4B, 087C,1C ;12307          JSR [CLR.FT0]                     ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12308                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0F4C, B61A,15 ;12309          MOV LS[T13] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR INT.MASK
U 0F4D, 3E1C,15 ;12310          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0F4E, 0878,AC ;12311          JSR [MOV.DATA]                    ;MOVE DATA FROM FWR[0] TO FWR[INT.MASK]
                ;12312                                            ; NOW FWR[INT.MASK] CONTAINS 0'S
U 0F4F, E5F8,15 ;12313          CLR LS[OS]                        ;CLEAR INDEX
                ;12314  REPEAT.T15.6:
U 0F50, 5FF6,15 ;12315          MCOM LS[SHIFT.OS(4-0)] TO WR[0]   ;GET SHIFTING 0 DATA PATTERN FROM LS
U 0F51, 3E16,15 ;12316          MOV WR[0] TO LS[T11]              ;EXPECTED DATA
U 0F52, 3EA2,15 ;12317          MOV WR[0] TO LS[DATA.1]           ;FIRST FLOAT DATA
U 0F53, 3EA4,15 ;12318          MOV WR[0] TO LS[DATA.2]           ;SECOND FLOAT DATA
U 0F54, BEA6,15 ;12319          MOV WR[0] TO LS[DATA.3]           ;THIRD FLOAT DATA
U 0F55, 0878,FC ;12320          JSR [LOAD.TRIPLE]                 ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12321                                            ; 8 EXT BITS IN WR 0 OF FPA
U 0F56, 3618,15 ;12322          MOV LS[T12] TO WR[0]              ;ADDRESS OF MOV FWR0 TO FWR2
U 0F57, 3E1C,15 ;12323          MOV WR[0] TO LS[T14]              ;SET UP LS FOR MOV SUBROUTINE
U 0F58, 88F5,EC ;12324          JSR [LOOP.T15.1.6]                ;TEST 'XOR' FUNCTION WITH SHIFTING
                ;12325                                            ; 0'S AND ALL 0'S (0'S IN INT.MASK)
U 0F59, 7FF8,15 ;12326          INC LS[OS]                        ;POINT TO NEXT PATTERN
U 0F5A, B6F8,15 ;12327          MOV LS[OS] TO WR[0]               ;GET VALUE OF INDEX
                ;12328          CMP LS[#20] WITH WR[0],           ;SEE IF DONE 32 PATTERNS YET
U 0F5B, 4E4A,35 ;12329           DT(LONG)&SET.ALU.CC;             ; AND SET CONDITION CODES
U 0F5C, 88F5,01 ;12330          JMP.IF[NEQ] TO [REPEAT.T15.6]     ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;12331
U 0F5D, 88F7,84 ;12332          JMP [END.T15]                     ;ELSE DONE WITH 'XOR' TEST
                ;12333
                ;12334  LOOP.T15.1.6:
U 0F5E, 6588,15 ;12335          CLR LS[ADDRESS.DATA]              ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 0F5F, FF88,15 ;12336          INC LS[ADDRESS.DATA]              ;1 INDICATES FIRST FLOAT FAILED
U 0F60, 0878,AC ;12337          JSR [MOV.DATA]                    ;MOV DATA FROM FWR 0 TO FWR 2
U 0F61, 3614,15 ;12338          MOV LS[T10] TO WR[0]              ;GET ADDRESS OF FPA 'XOR' INSTRUCTION
U 0F62, B716,95 ;12339          MOV LS[#300] TO WR[1]             ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 0F63, 90F6,15 ;12340          MISC2 [TRAP.ACC] WR[0]            ;FORCE FWR[0] WITH FQ TO FWR[2],
                ;12341                                            ; EWR[0] WITH EQ TO EWR[2]
U 0F64, 10F6,95 ;12342          MISC2 [TRAP.ACC] WR[1]            ;FORCE LOOP SELF
U 0F65, 087A,1C ;12343          JSR [STORE.2.TRIPLE]              ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;12344                                            ; AND THIRD.FLT
U 0F66, BE8B,15 ;12345          MOV WR[2] TO LS[ERROR.MASK]       ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;12346                                            ; AND EXPONENT (BITS 7-14)
U 0F67, B6A8,15 ;12347          MOV LS[FIRST.FLT] TO WR[0]        ;PUT FIRST FLT RESULT IN WR 0
U 0F68, 3616,95 ;12348          MOV LS[T11] TO WR[1]              ;EXPECTED DATA
U 0F69, 0869,3C ;12349          JSR [CHECK.RESULT]                ;CHECK FOR ERRORS
U 0F6A, 08F5,E4 ;12350          JMP [LOOP.T15.1.6]                ;ERROR LOOP IF ENABLED
                ;12351
U 0F6B, FF88,15 ;12352          INC LS[ADDRESS.DATA]              ;INC PRINTOUT UNDER OTHER DATA TO 2
U 0F6C, E58A,15 ;12353          CLR LS[ERROR.MASK]                ;CHECK ALL BITS
U 0F6D, 36AA,15 ;12354          MOV LS[SEC.FLT] TO WR[0]          ;PUT SECOND FLT RESULT IN WR 0
U 0F6E, 3616,95 ;12355          MOV LS[T11] TO WR[1]              ;EXPECTED DATA
U 0F6F, 0869,3C ;12356          JSR [CHECK.RESULT]                ;CHECK FOR ERRORS
U 0F70, 08F5,E4 ;12357          JMP [LOOP.T15.1.6]                ;ERROR LOOP IF ENABLED
                ;12358
U 0F71, FF88,15 ;12359          INC LS[ADDRESS.DATA]              ;INC PRINTOUT UNDER OTHER DATA TO 3
U 0F72, 3E8B,95 ;12360          MOV WR[3] TO LS[ERROR.MASK]       ;CHECK BITS 8-15 ONLY
```

```
U 0F73, 36AC,15 ;12361          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 0F74, 3616,95 ;12362          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0F75, 0869,3C ;12363          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0F76, 08F5,E4 ;12364          JMP [LOOP.T15.1.6]              ;ERROR LOOP IF ENABLED
U 0F77, 5B00,14 ;12365          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;12366  END.T15:
```

```
;12367  .PAGE
;12368  .TOC    "TEST 16 XNOR FUNCTION TEST(M8389 FPA MODULE)"
;12369  ;++
;12370  ;
;12371  ;
;12372  ; Test Description:
;12373  ;
;12374  ;       This test checks the XNOR function of the fraction data path. The FPA
;12375  ;       micro-code contains only 2 XNOR instructions that can be used.  These
;12376  ;       are XNOR FWR[FT2] WITH FWR[FT2] and XNOR ZEROS WITH FWR[FT5] TO
;12377  ;       FWR[FT6].  The FT2 to FT2 instruction will be used to check XNOR of
;12378  ;       zeros with zeros and ones with ones.  The other XNOR instruction will
;12379  ;       be used to check XNOR of zeros with ones and zeros with shifting ones
;12380  ;       and shifting zeros.
;12381  ;
;12382  ; Assumptions:
;12383  ;
;12384  ;       It is assumed that all of the WR and the Q registers of all the 2901's
;12385  ;       have been tested. All of the logical functions with the exception of
;12386  ;       X-NOR have also been tested and work.
;12387  ;
;12388  ; Test Steps:
;12389  ;
;12390  ;       1) Load zeros into FWR 0, and MOV the contents of FWR 0 into FWR 2.
;12391  ;       2) Issue a MISC instruction to force an XNOR of FWR 2 and FWR 2,
;12392  ;          the result of which will be in WR 2 and can be read back to
;12393  ;          the CPU to be checked. If not all ones then issue error1.
;12394  ;       3) Load ones into FWR 0, move FWR 0 to FWR 2.
;12395  ;       4) Issue a MISC instruction to force an XNOR of FWR 2 and FWR 2,
;12396  ;          the result of which will be in WR 2 and can be read back to
;12397  ;          the CPU to be checked. If not all ones then issue error2.
;12398  ;       5) Load ones into FWR 0, then MOV FWR 0 to FWR 5.
;12399  ;       6) Issue a MISC instruction to force an XNOR of zeros and FWR 5, the
;12400  ;          result of which will be in WR 6.  MOV FWR 6 to FWR 2 and read the
;12401  ;          result back to the CPU to be checked. If not all zeros, then issue
;12402  ;          error3.
;12403  ;       7) Load FWR 0 with a shifting ones pattern.  Mov FWR 0 to FWR 5.
;12404  ;       8) Issue a MISC instruction to force an XNOR of zeros and FWR 5,
;12405  ;          the result of which will be in WR 6.  MOV FWR 6 to FWR 2 and read
;12406  ;          the result back to the CPU to be checked. If the result is not the
;12407  ;          complement of the shifting pattern, issue error 4.
;12408  ;       9) Repeat steps 7 and 8 until 32 shifting patterns have been used.
;12409  ;       10)Repeat steps 7 through 9 with a shifting 0's pattern.
;12410  ;
;12411  ; Error:
;12412  ;
;12413  ;       Note:  For the following errors, the data printed under OTHER in the
;12414  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;12415  ;       FLT) which failed.
;12416  ;
;12417  ;       Error1 - XNOR failed when R and S were both zero.
;12418  ;       Error2 - XNOR failed when R and S were both one.
;12419  ;       Error3 - XNOR failed when R was zero and S was one.
;12420  ;       Error4 - XNOR failed when R was zero and S was shifting 1's.
;12421  ;       Error5 - XNOR failed when R was zero and S was shifting 0's.
```

```
                    ;12422  ;
                    ;12423  ; Debug:
                    ;12424  ;
                    ;12425  ;       Error1: The possible sources of error are the control signals of the
                    ;12426  ;               fraction data path 2901's or one of the 2901's itself.  The
                    ;12427  ;               fraction control lines should be 011111001 (from I8 through I0
                    ;12428  ;               respectively) during the 'XOR' u-instruction.  Of the frac-
                    ;12429  ;               tion control lines only the function (R XOR S) is being tested
                    ;12430  ;               for the first time.
                    ;12431  ;                 If the control lines I3, I4, or I5 of the function control
                    ;12432  ;               are in error, trace them back through the buffers to the
                    ;12433  ;               control store ROM.  I3 comes through a MUX.  The select lines
                    ;12434  ;               ENB MUL (1) L and ENB DIV (1) L should both be high.
                    ;12435  ;                 If one of the other control lines is incorrect, trace it back
                    ;12436  ;               in the same manner.
                    ;12437  ;                 If the control lines are OK, suspect the 2901 which is out-
                    ;12438  ;               putting the bad bit.
                    ;12439  ;       Error2: If there was no error in the first test then the most likely
                    ;12440  ;               cause of the error is the 2901 itself.
                    ;12441  ;       Error3: This error could indicate a problem in the SRC control lines,
                    ;12442  ;               as these are different than error 1.  The control lines should
                    ;12443  ;               be 011111100 (from I8 through I0 respectively).  However, the
                    ;12444  ;               most likely error is in a 2901 itself.
                    ;12445  ;       Error4: Same as Error2.
                    ;12446  ;       Error5: Same as Error2.
                    ;12447  ;
                    ;12448  ;--
                    ;12449
                    ;12450  ;
                    ;12451  ;       TEMPORARY LS LOCATIONS USED:
                    ;12452  ;
                    ;12453  ; T9    MAIN MEMORY POINTER
                    ;12454  ; T10   ADDRESS OF 'XNOR FWR[FT2] WITH FWR[FT2]'
                    ;12455  ; T11   EXPECTED DATA
                    ;12456  ; T14   ADDRESS OF 'MOV FWR0 TO FWR2'
                    ;12457  ;
                    ;12458  T.16:
                    ;12459
U 0F78, B65E,15     ;12460          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;12461                                          ; STATUS WORD
U 0F79, 3E80,15     ;12462          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;12463                                          ; BIT 15 INDICATES START OF TEST TO
                    ;12464                                          ; CONSOLE
U 0F7A, 10E0,15     ;12465          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;12466  WAIT.T16.0:
U 0F7B, 88F7,B4     ;12467          JMP [WAIT.T16.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;12468                                          ; RESPOND
U 0F7C, 087E,6C     ;12469          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;12470                                          ; AND SIZE BITS
U 0F7D, B670,15     ;12471          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                    ;12472                                          ; OTHER DATA
U 0F7E, 3E80,15     ;12473          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 0F7F, 37C2,15     ;12474          MOV LS[T16.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 0F80, BE12,15     ;12475          MOV WR[0] TO LS[T9]              ;SET UP POINTER IN LS
U 0F81, 8870,3C     ;12476          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
```

```
U 0F82, 3E1C,15 ;12477          MOV WR[0] TO LS[T14]             ;PUT FPA ADDRESS IN LS FOR MOV SUBROUTINE
                ;12478                                           ; MOVE IS FROM FPA WR 0 TO FPA WR 2
U 0F83, 8870,3C ;12479          JSR [L0]                         ;LOAD WR 0 WITH MEMORY DATA
U 0F84, BE14,15 ;12480          MOV WR[0] TO LS[T10]             ;PUT FPA ADDRESS IN LS FOR 'XNOR' INSTRUCTION
                ;12481                                           ; FPA U-WORD IS XNOR FWR[FT2] WITH FWR[FT2]
U 0F85, 3725,15 ;12482          MOV LS[#FFFF00FF] TO WR[2]       ;BITS 8-15 = 0
U 0F86, A0C5,15 ;12483          COM WR[2]                        ;BITS 8-15 = 1
U 0F87, C74F,15 ;12484          BIS LS[BIT7] TO WR[2]            ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;12485                                           ; BIT AND EXPONENT)
U 0F88, B725,95 ;12486          MOV LS[#FFFF00FF] TO WR[3]       ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;12487
U 0F89, E516,15 ;12488          CLR LS[T11]                      ;ZEROS IN AN LS LOCATION
U 0F8A, FD16,15 ;12489          COM LS[T11]                      ;EXPECTED DATA STORED HERE (ALL 1'S)
U 0F8B, 087C,1C ;12490          JSR [CLR.FT0]                    ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12491                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0F8C, 88FC,7C ;12492          JSR [LOOP.T16.1.2]               ;TEST 'XNOR' FUNCTION WITH 0'S AND 0'S
                ;12493
U 0F8D, FF82,15 ;12494          INC LS[ERROR.NUMBER]             ;ERROR 2
U 0F8E, FDA2,15 ;12495          COM LS[DATA.1]                   ;ONES IN FIRST FLOAT
U 0F8F, FDA4,15 ;12496          COM LS[DATA.2]                   ;ONES IN SECOND FLOAT
U 0F90, 7DA6,15 ;12497          COM LS[DATA.3]                   ;ONES IN THIRD (HUGE) FLOAT
U 0F91, 0878,FC ;12498          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12499                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0F92, 88FC,7C ;12500          JSR [LOOP.T16.1.2]               ;TEST 'XNOR' FUNCTION 1'S AND 1'S
                ;12501
                ;12502  ;
                ;12503  ;       TEMPORARY LS LOCATIONS USED:
                ;12504  ;
                ;12505  ; T9    MAIN MEMORY POINTER
                ;12506  ; T10   ADDRESS OF 'XNOR ZEROS WITH FWR[FT5] TO FWR[FT6]'
                ;12507  ; T11   EXPECTED DATA
                ;12508  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FWR[FT5]'
                ;12509  ; T13   ADDRESS OF 'MOV FWR[FT6] TO FWR[FT2]
                ;12510  ; T14   MOV SUBROUTINE USES LS T14 AS ADDRESS OF MOV ABOVE
                ;12511  ;
U 0F93, FF82,15 ;12512          INC LS[ERROR.NUMBER]             ;ERROR 3
U 0F94, E516,15 ;12513          CLR LS[T11]                      ;EXPECTED DATA = 0'S
U 0F95, 8870,3C ;12514          JSR [L0]                         ;GET FPA ADDRESS FROM MEMORY
U 0F96, BE18,15 ;12515          MOV WR[0] TO LS[T12]             ;LOAD ADDRESS OF MOV FWR0 TO FWR5 IN LS
U 0F97, 8870,3C ;12516          JSR [L0]                         ;GET NEXT FPA ADDRESS FROM MEMORY
U 0F98, 3E1A,15 ;12517          MOV WR[0] TO LS[T13]             ;LOAD ADDRESS OF MOV FWR6 TO FWR2 IN LS
U 0F99, 8870,3C ;12518          JSR [L0]                         ;GET NEXT FPA ADDRESS FROM MEMORY
U 0F9A, BE14,15 ;12519          MOV WR[0] TO LS[T10]             ;LOAD ADDRESS OF XNOR ZEROS WITH FWR5
                ;12520                                           ; TO FWR6
U 0F9B, 0878,FC ;12521          JSR [LOAD.TRIPLE]                ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12522                                           ; 8 EXT BITS IN WR 0 OF FPA
U 0F9C, 3618,15 ;12523          MOV LS[T12] TO WR[0]             ;ADDRESS OF MOV FWR0 TO FWR5
U 0F9D, 3E1C,15 ;12524          MOV WR[0] TO LS[T14]             ;LOAD LS T14 FOR MOV SUBROUTINE
U 0F9E, 0878,AC ;12525          JSR [MOV.DATA]                   ;MOVE DATA FROM FWR[0] TO FWR5
                ;12526                                           ; NOW FWR5 CONTAINS 1'S
U 0F9F, B61A,15 ;12527          MOV LS[T13] TO WR[0]             ;ADDRESS OF MOV FWR6 TO FWR2
U 0FA0, 3E1C,15 ;12528          MOV WR[0] TO LS[T14]             ;LOAD LS T14 FOR MOV SUBROUTINE
U 0FA1, 08FE,1C ;12529          JSR [LOOP.T16.3.5]               ;TEST 'XNOR' FUNCTION WITH 0'S AND 1'S
                ;12530
U 0FA2, FF82,15 ;12531          INC LS[ERROR.NUMBER]             ;ERROR 4
```

```
U 0FA3, E5F8,15 ;12532          CLR LS[OS]                          ;CLEAR INDEX
                ;12533  REPEAT.T16.4:
U 0FA4, 36F6,15 ;12534          MOV LS[SHIFT.OS(4-0)] TO WR[0]      ;GET SHIFTING 1 DATA PATTERN FROM LS
U 0FA5, F816,15 ;12535          MCOM WR[0] TO LS[T11]               ;EXPECTED DATA
U 0FA6, 3EA2,15 ;12536          MOV WR[0] TO LS[DATA.1]             ;FIRST FLOAT DATA
U 0FA7, 3EA4,15 ;12537          MOV WR[0] TO LS[DATA.2]             ;SECOND FLOAT DATA
U 0FA8, BEA6,15 ;12538          MOV WR[0] TO LS[DATA.3]             ;THIRD FLOAT DATA
U 0FA9, 0878,FC ;12539          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12540                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0FAA, 3618,15 ;12541          MOV LS[T12] TO WR[0]                ;ADDRESS OF MOV FWR0 TO FWR5
U 0FAB, 3E1C,15 ;12542          MOV WR[0] TO LS[T14]                ;LOAD LS T14 FOR MOV SUBROUTINE
U 0FAC, 0878,AC ;12543          JSR [MOV.DATA]                      ;MOVE DATA FROM FWR[0] TO FWR5
                ;12544                                              ; NOW FWR5 CONTAINS SHIFTING 1'S
U 0FAD, B61A,15 ;12545          MOV LS[T13] TO WR[0]                ;ADDRESS OF MOV FWR6 TO FWR2
U 0FAE, 3E1C,15 ;12546          MOV WR[0] TO LS[T14]                ;LOAD LS T14 FOR MOV SUBROUTINE
U 0FAF, 08FE,1C ;12547          JSR [LOOP.T16.3.5]                  ;TEST 'XNOR' FUNCTION WITH SHIFTING 1'S
                ;12548                                              ; AND 0'S
U 0FB0, 7FF8,15 ;12549          INC LS[OS]                          ;POINT TO NEXT PATTERN
U 0FB1, B6F8,15 ;12550          MOV LS[OS] TO WR[0]                 ;GET VALUE OF INDEX
                ;12551          CMP LS[#20] WITH WR[0],             ;SEE IF DONE 32 PATTERNS YET
U 0FB2, 4E4A,35 ;12552           DT(LONG)&SET.ALU.CC                ; AND SET CONDITION CODES
U 0FB3, 08FA,41 ;12553          JMP.IF[NEQ] TO [REPEAT.T16.4]       ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;12554
U 0FB4, FF82,15 ;12555          INC LS[ERROR.NUMBER]                ;ERROR 5
U 0FB5, E5F8,15 ;12556          CLR LS[OS]                          ;CLEAR INDEX
                ;12557  REPEAT.T16.5:
U 0FB6, 5FF6,15 ;12558          MCOM LS[SHIFT.OS(4-0)] TO WR[0]     ;GET SHIFTING 0 DATA PATTERN FROM LS
U 0FB7, F816,15 ;12559          MCOM WR[0] TO LS[T11]               ;EXPECTED DATA
U 0FB8, 3EA2,15 ;12560          MOV WR[0] TO LS[DATA.1]             ;FIRST FLOAT DATA
U 0FB9, 3EA4,15 ;12561          MOV WR[0] TO LS[DATA.2]             ;SECOND FLOAT DATA
U 0FBA, BEA6,15 ;12562          MOV WR[0] TO LS[DATA.3]             ;THIRD FLOAT DATA
U 0FBB, 0878,FC ;12563          JSR [LOAD.TRIPLE]                   ;LOAD 56 FRAC BITS, 8 EXP BITS, AND
                ;12564                                              ; 8 EXT BITS IN WR 0 OF FPA
U 0FBC, 3618,15 ;12565          MOV LS[T12] TO WR[0]                ;ADDRESS OF MOV FWR0 TO FWR5
U 0FBD, 3E1C,15 ;12566          MOV WR[0] TO LS[T14]                ;LOAD LS T14 FOR MOV SUBROUTINE
U 0FBE, 0878,AC ;12567          JSR [MOV.DATA]                      ;MOVE DATA FROM FWR[0] TO FWR5
                ;12568                                              ; NOW FWR5 CONTAINS SHIFTING 0'S
U 0FBF, B61A,15 ;12569          MOV LS[T13] TO WR[0]                ;ADDRESS OF MOV FWR6 TO FWR2
U 0FC0, 3E1C,15 ;12570          MOV WR[0] TO LS[T14]                ;LOAD LS T14 FOR MOV SUBROUTINE
U 0FC1, 08FE,1C ;12571          JSR [LOOP.T16.3.5]                  ;TEST 'XNOR' FUNCTION WITH SHIFTING 0'S
                ;12572                                              ; AND 0'S
U 0FC2, 7FF8,15 ;12573          INC LS[OS]                          ;POINT TO NEXT PATTERN
U 0FC3, B6F8,15 ;12574          MOV LS[OS] TO WR[0]                 ;GET VALUE OF INDEX
                ;12575          CMP LS[#20] WITH WR[0],             ;SEE IF DONE 32 PATTERNS YET
U 0FC4, 4E4A,35 ;12576           DT(LONG)&SET.ALU.CC                ; AND SET CONDITION CODES
U 0FC5, 08FB,61 ;12577          JMP.IF[NEQ] TO [REPEAT.T16.5]       ;CONTINUE WITH NEXT PATTERN UNTIL DONE
                ;12578
U 0FC6, 08FF,B4 ;12579          JMP [END.T16]                       ;ELSE DONE WITH 'AND' TEST
                ;12580
                ;12581
                ;12582  LOOP.T16.1.2:
U 0FC7, 6588,15 ;12583          CLR LS[ADDRESS.DATA]                ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 0FC8, FF88,15 ;12584          INC LS[ADDRESS.DATA]                ;1 INDICATES FIRST FLOAT FAILED
U 0FC9, 0878,AC ;12585          JSR [MOV.DATA]                      ;MOVE DATA FROM FWR[0] TO FWR[2]
U 0FCA, 3614,15 ;12586          MOV LS[T10] TO WR[0]                ;GET ADDRESS OF FPA 'XNOR' INSTRUCTION
```

```
U 0FCB, B716,95 ;12587          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 0FCC, 90F6,15 ;12588          MISC2 [TRAP.ACC] WR[0]          ;FORCE XNOR ZEROS WITH FWR[5] TO FWR[6]
U 0FCD, 10F6,95 ;12589          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 0FCE, 087A,1C ;12590          JSR [STORE.2.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;12591                                          ; AND THIRD.FLT
U 0FCF, BE8B,15 ;12592          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;12593                                          ; AND EXPONENT (BITS 7-14)
U 0FD0, B6A8,15 ;12594          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 0FD1, 3616,95 ;12595          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0FD2, 0869,3C ;12596          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0FD3, 08FC,74 ;12597          JMP [LOOP.T16.1.2]              ;ERROR LOOP IF ENABLED
                ;12598
U 0FD4, FF88,15 ;12599          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 0FD5, E58A,15 ;12600          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 0FD6, 36AA,15 ;12601          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 0FD7, 3616,95 ;12602          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0FD8, 0869,3C ;12603          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0FD9, 08FC,74 ;12604          JMP [LOOP.T16.1.2]              ;ERROR LOOP IF ENABLED
                ;12605
U 0FDA, FF88,15 ;12606          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 0FDB, 3E8B,95 ;12607          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 0FDC, 36AC,15 ;12608          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 0FDD, 3616,95 ;12609          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0FDE, 0869,3C ;12610          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0FDF, 08FC,74 ;12611          JMP [LOOP.T16.1.2]              ;ERROR LOOP IF ENABLED
U 0FE0, 5B00,14 ;12612          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;12613
                ;12614  LOOP.T16.3.5:
U 0FE1, 6588,15 ;12615          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 0FE2, FF88,15 ;12616          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 0FE3, 3614,15 ;12617          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'XNOR' INSTRUCTION
U 0FE4, B716,95 ;12618          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 0FE5, 90F6,15 ;12619          MISC2 [TRAP.ACC] WR[0]          ;FORCE XNOR ZEROS WITH FWR[5] TO FWR[6]
U 0FE6, 10F6,95 ;12620          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 0FE7, 0878,AC ;12621          JSR [MOV.DATA]                  ;MOVE RESULT FROM FWR[FT6] TO FWR[FT2]
U 0FE8, 087A,1C ;12622          JSR [STORE.2.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;12623                                          ; AND THIRD.FLT
U 0FE9, BE8B,15 ;12624          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;12625                                          ; AND EXPONENT (BITS 7-14)
U 0FEA, B6A8,15 ;12626          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 0FEB, 3616,95 ;12627          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0FEC, 0869,3C ;12628          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0FED, 88FE,14 ;12629          JMP [LOOP.T16.3.5]              ;ERROR LOOP IF ENABLED
                ;12630
U 0FEE, FF88,15 ;12631          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 0FEF, E58A,15 ;12632          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 0FF0, 36AA,15 ;12633          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 0FF1, 3616,95 ;12634          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 0FF2, 0869,3C ;12635          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 0FF3, 88FE,14 ;12636          JMP [LOOP.T16.3.5]              ;ERROR LOOP IF ENABLED
                ;12637
U 0FF4, FF88,15 ;12638          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 0FF5, 3E8B,95 ;12639          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 0FF6, 36AC,15 ;12640          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 0FF7, 3616,95 ;12641          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
```

```
U OFF8, 0869,3C ;12642          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U OFF9, 88FE,14 ;12643          JMP [LOOP.T16.3.5]              ;ERROR LOOP IF ENABLED
U OFFA, 5B00,14 ;12644          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;12645  END.T16:
```

```
;12646  .PAGE
;12647  .TOC    "TEST 17 R PLUS S FUNCTION TEST(M8389 FPA MODULE)"
;12648  ;++
;12649  ;
;12650  ;
;12651  ; Test Description:
;12652  ;
;12653  ;       The purpose of this test is to check the R+S function of the fraction
;12654  ;       and exponent data path. The instruction that will be used is ADD FWR[]
;12655  ;       TO FWR[] + FCOUT and ADD EWR[] TO EWR[] + FCOUT. The source is A and B,
;12656  ;       the function is R+S, and the destination is F->B.  CIN to the least
;12657  ;       significant bit of the fraction or exponent data path will be set when
;12658  ;       an ADD is performed on the fraction data path without clearing one of
;12659  ;       the operands first.  This occurs because the hidden bit is set on a
;12660  ;       load, and this is the most significant bit when an ADD is performed.
;12661  ;       To avoid a COUT, the FWR used in the add must first be cleared.  This
;12662  ;       test will check the case where CIN is either set or clear in both the
;12663  ;       fraction and exponent data paths.  The exponent data path will be
;12664  ;       checked seperately due to the need to shift data into the upper 8 bits
;12665  ;       instead of loading them directly.  The data patterns that will be used
;12666  ;       are 0's and 0's, all 1's and 0's, 0's and all 1's, all 1's and all 1's,
;12667  ;       and shifting 1's and shifting 1's. These data patterns will be suffi-
;12668  ;       ceint to test all GEN and PROP logic.
;12669  ;
;12670  ; Assumptions:
;12671  ;
;12672  ;       It is assumed that all of the logic functions have been tested.
;12673  ;
;12674  ; Test Steps:
;12675  ;
;12676  ;       1) Load FWR[0] first float to set the hidden bit (FRAC 55).  Mov FWR[0]
;12677  ;          to FWR[3] so that the MSB in this odd WR is set.
;12678  ;       2) Load FWR[0] with all 0's in all bits and mov to FWR[2].  Load FWR[0]
;12679  ;          with all 0's again.  Next clear FWR[1] so COUT will not be set.
;12680  ;       3) Perform ADD FWR[3] PLUS FWR[1] TO FQ.  Since the MSB of FWR[1] is
;12681  ;          clear, FCOUT will not set.  Perform ADD FWR[2] TO FWR[0] + FCOUT.
;12682  ;       4) Store result from FWR[0] and check for 0's (0 + 0 with no carry in).
;12683  ;       5) Load FWR[0] with ones.  Then clear FWR[1] again so COUT will not
;12684  ;          set.  Repeat step 3.
;12685  ;       6) Store result from FWR[0] and check for 1's (0 + 1's with no carry in).
;12686  ;       7) Load FWR[0] with 1's and move to FWR[2].  Clear FRW[0] and FWR[1]
;12687  ;          and repeat step 3.
;12688  ;       8) Store result from FWR[0] and check for 1's (1's + 0 with no carry in).
;12689  ;       9) Load FWR[0] with ones.  Clear FWR[1] and repeat step 3.
;12690  ;       10)Store result from FWR[0] and check for FFFFFFFE (1's + 1's with no
;12691  ;          carry in).
;12692  ;       11)Load FWR[0] with a shifting 1 pattern and mov to FWR[2].  Reload
;12693  ;          FWR[0] with the same shifting data pattern, clear FWR[1] and repeat
;12694  ;          step 3.
;12695  ;       12)Store result from FWR[0] and check for shifting pattern shifted left
;12696  ;          one place (shifting 1's + shifting 1's with no carry in).
;12697  ;       13)Repeat steps 2 through 10 except do not clear FWR[1] before the add.
;12698  ;          Instead move FWR[0] to FWR[1] before each add.  This will produce a
;12699  ;          COUT.  Check result for correct data.
;12700  ;       14)Repeat steps 2 through 13, this time checking the exponent data path
```

```
;12701  ;          by shifting data into the upper 8 bits.
;12702  ;
;12703  ; Error:
;12704  ;
;12705  ;       Note:  For the following errors, the data printed under OTHER in the
;12706  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;12707  ;       FLT) which failed.
;12708  ;
;12709  ;       Error1 - ADD failed with data of 0's in FWR[2] and 0's in FWR[0], no
;12710  ;                carry in.
;12711  ;       Error2 - ADD failed with data of 0's in FWR[2] and 1's in FWR[0], no
;12712  ;                carry in.
;12713  ;       Error3 - ADD failed with data of 1's in FWR[2] and 0's in FWR[0], no
;12714  ;                carry in.
;12715  ;       Error4 - ADD failed with data of 1's in FWR[2] and 1's in FWR[0], no
;12716  ;                carry in.
;12717  ;       Error5 - ADD failed with data of shifting 1's in FWR[2] and shifting
;12718  ;                1's in FWR[0], no carry in.
;12719  ;       Error6 - ADD failed with data of 0's in FWR[2] and 0's in FWR[0], with
;12720  ;                carry in set.
;12721  ;       Error7 - ADD failed with data of 0's in FWR[2] and 1's in FWR[0], with
;12722  ;                carry in set.
;12723  ;       Error8 - ADD failed with data of 1's in FWR[2] and 0's in FWR[0], with
;12724  ;                carry in set.
;12725  ;       Error9 - ADD failed with data of 1's in FWR[2] and 1's in FWR[0], with
;12726  ;                carry in set.
;12727  ;
;12728  ;       Note:  For the following errors, the data printed under OTHER in the
;12729  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;12730  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;12731  ;
;12732  ;       ErrorA - ADD failed with data of 0's in EWR[ZERO] and 0's in EWR[0], no
;12733  ;                carry in.
;12734  ;       ErrorB - ADD failed with data of 0's in EWR[ZERO] and 1's in EWR[0], no
;12735  ;                carry in.
;12736  ;       ErrorC - ADD failed with data of 1's in EWR[ZERO] and 0's in EWR[0], no
;12737  ;                carry in.
;12738  ;       ErrorD - ADD failed with data of 1's in EWR[ZERO] and 1's in EWR[0], no
;12739  ;                carry in.
;12740  ;       ErrorE - ADD failed with data of shifting 1's in EWR[ZERO] and shifting
;12741  ;                1's in EWR[0], no carry in.
;12742  ;       ErrorF - ADD failed with data of 0's in EWR[ZERO] and 0's in EWR[0],
;12743  ;                with carry in set.
;12744  ;       Error10 - ADD failed with data of 0's in EWR[ZERO] and 1's in EWR[0],
;12745  ;                with carry in set.
;12746  ;       Error11 - ADD failed with data of 1's in EWR[ZERO] and 0's in EWR[0],
;12747  ;                with carry in set.
;12748  ;       Error12 - ADD failed with data of 1's in EWR[ZERO] and 1's in EWR[0],
;12749  ;                with carry in set.
;12750  ;
;12751  ;
;12752  ; Debug:
;12753  ;
;12754  ;       Error1: This error indicates a basic problem with either the control
;12755  ;               logic, the EXP DECODE PAL, the carry skipper logic, or one or
```

```
                ;12756  ;               more of the 2901's.  The fraction control lines should be
                ;12757  ;               011000001 during the ADD FWR[2] to FWR[0] + FCOUT insruction.
                ;12758  ;               Of the control lines only the FUNC (I5 through I3) has not been
                ;12759  ;               previously tested.  If the control lines are in error, trace
                ;12760  ;               them back to their source at the CS PROMS.
                ;12761  ;                 If the low order bit is in error, check that CIN EXT00 H is
                ;12762  ;               low after the ADD FWR[FT2] TO FWR[FT0] instruction.  If not,
                ;12763  ;               check the inputs to the EXP DECODE PAL for a low on ENB CLK5 L,
                ;12764  ;               a high on EXTEND CLOCK (1) H, and a low on FRAC COUT SAVE H.
                ;12765  ;               If these inputs are OK, then the PAL is probably bad.
                ;12766  ;                 If FRAC COUT SAVE H is high, trace it back through the latch
                ;12767  ;               to the carry out output of the 2901 that produces FRAC COUT
                ;12768  ;                 If a random bit or two is setting, the problem is probably
                ;12769  ;               in a 2901 or a CARRY SKIPPER.  All CIN's to the 2901 should be
                ;12770  ;               low.
                ;12771  ;       Error2: Same as Error 1 except the most likely suspect is a 2901.
                ;12772  ;       Error3: Same as error2.
                ;12773  ;       Error4: Same as Error 1 except the most likely suspect is a 2901 or a
                ;12774  ;               CARRY SKIPPER.  All CIN's should be asserted high on the inputs
                ;12775  ;               to the 2901's from the CARRY SKIPPERS.
                ;12776  ;       Error5: The most likely suspect is the 2901 with the failing bit.
                ;12777  ;       Error6: The most likely suspect is the EXP DECODE PAL.  This is the
                ;12778  ;               check on the FCOUT logic being asserted.  Check FRAC COUT SAVE
                ;12779  ;               H going into the EXP DECODE PAL during the ADD FWR[FT2] TO FWR
                ;12780  ;               [FT0] instruction.  The inputs should be the same as in Error
                ;12781  ;               1 except FRAC COUT SAVE should be high and EXT CIN H should be
                ;12782  ;               high.  If the inputs are OK but EXT CIN H is not asserted, the
                ;12783  ;               PAL is probably bad.
                ;12784  ;                 If FRAC COUT SAVE H is low, trace it back through the latch
                ;12785  ;               to the 2901 that produces FRAC COUT H.  This should be asserted
                ;12786  ;               during the ADD FWR[FT3] PLUS FWR[FT1] TO FQ instruction.
                ;12787  ;       Errors 7-9: Suspect the 2901 outputting the failing bit.
                ;12788  ;       Errors A-12: Same as Error 1-9 except the exponent path is being test-
                ;12789  ;               ed.
                ;12790  ;
                ;12791  ;--
                ;12792
                ;12793  ;
                ;12794  ;       TEMPORARY LS LOCATIONS USED:
                ;12795  ;
                ;12796  ; T8    ADDRESS OF 'CLR FWR[FT1]'
                ;12797  ; T9    MAIN MEMORY POINTER
                ;12798  ; T10   ADDRESS OF 'ADD FWR[FT3] PLUS FWR[FT1] TO FQ' (FIRST WORD)
                ;12799  ;               'ADD FWR[FT2] TO FWR[FT0] + FCOUT' (SECOND WORD)
                ;12800  ; T12   EXPECTED DATA FROM FIRST AND SECOND FLOAT
                ;12801  ; T13   EXPECTED DATA FROM THIRD FLOAT
                ;12802  ; T14   ADDRESS OF 'MOV FWR[FT0] TO FWR[FT2]'
                ;12803  ;
                ;12804  T.17:
                ;12805
U OFFB, B65E,15 ;12806          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;12807                                          ; STATUS WORD
U OFFC, 3E80,15 ;12808          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;12809                                          ; BIT 15 INDICATES START OF TEST TO
                ;12810                                          ; CONSOLE
```

```
U OFFD, 10E0,15 ;12811          MISC [SET.CP.ATTN]                ;ISSUE CPU ATTN TO CONSOLE
                ;12812  WAIT.T17.0:
U OFFE, 08FF,E4 ;12813          JMP [WAIT.T17.0]                  ;LOOP HERE WAITING FOR CONSOLE TO
                ;12814                                            ; RESPOND
U OFFF, 087E,6C ;12815          JSR [SETUP]                       ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;12816                                            ; AND SIZE BITS
U 1000, B670,15 ;12817          MOV LS[OTHER.DATA] TO WR[0]       ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;12818                                            ; OTHER DATA
U 1001, 3E80,15 ;12819          MOV WR[0] TO LS[CONTROL.STATUS]   ;SET BIT IN CONTROL WORD
U 1002, 37C4,15 ;12820          MOV LS[T17.POINTER] TO WR[0]      ;GET POINTER TO MAIN MEMORY ADDRESS
U 1003, BE12,15 ;12821          MOV WR[0] TO LS[T9]               ;SET UP POINTER IN LS
U 1004, 0870,8C ;12822          JSR [L1]                          ;LOAD WR 1 WITH ADDRESS OF MOV FWR[0]
                ;12823                                            ; TO FWR[3]
U 1005, B72E,15 ;12824          MOV LS[#2A7] TO WR[0]             ;ADDRESS OF DOUBLE LOAD ROUTINE
U 1006, 90F6,15 ;12825          MISC2 [TRAP.ACC] WR[0]            ;FORCE ADDRESS OF DOUBLE LOAD
U 1007, DB00,15 ;12826          NOP                               ;WAIT FOR FPA TO GET READY TO ACCEPT DATA
U 1008, 10F4,15 ;12827          MISC2 [ASSERT.DA.AND.PASS.WR]     ;LOAD FIRST FLT INTO FWR 0 (SETS HIDDEN
                ;12828                                            ; BIT)
U 1009, 10F6,95 ;12829          MISC2 [TRAP.ACC] WR[1]            ;MOVE FROM FWR 0 TO FWR 3 (SETS MSB OF
                ;12830                                            ; FWR 3)
U 100A, 8870,3C ;12831          JSR [L0]                          ;LOAD WR 0 WITH ADDRESS OF MOV FWR[0]
                ;12832                                            ; TO FWR[2]
U 100B, 3E1C,15 ;12833          MOV WR[0] TO LS[T14]               ;PUT FPA ADDRESS IN LS FOR MOV SUBROUTINE
                ;12834                                            ; MOVE IS FROM FPA WR 0 TO FPA WR 2
U 100C, 8870,3C ;12835          JSR [L0]                          ;LOAD WR 0 WITH MEMORY DATA
U 100D, BE14,15 ;12836          MOV WR[0] TO LS[T10]              ;PUT FPA ADDRESS IN LS FOR 'ADD' INSTRUCTION
U 100E, 8870,3C ;12837          JSR [L0]                          ;LOAD WR 0 WITH MEMORY DATA
U 100F, 3E10,15 ;12838          MOV WR[0] TO LS[T8]               ;STORE 'CLR FWR[FT1]' ADDRESS IN LS
U 1010, 3725,15 ;12839          MOV LS[#FFFF00FF] TO WR[2]        ;BITS 8-15 = 0
U 1011, A0C5,15 ;12840          COM WR[2]                         ;BITS 8-15 = 1
U 1012, C74F,15 ;12841          BIS LS[BIT7] TO WR[2]             ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;12842                                            ; BIT AND EXPONENT)
U 1013, B725,95 ;12843          MOV LS[#FFFF00FF] TO WR[3]        ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;12844
U 1014, 6518,15 ;12845          CLR LS[T12]                       ;EXPECTED DATA = 0'S IN FIRST AND
                ;12846                                            ; SECOND FLOAT
U 1015, E51A,15 ;12847          CLR LS[T13]                       ;EXPECTED DATA = 0'S IN THIRD FLOAT
U 1016, 087C,1C ;12848          JSR [CLR.FT0]                     ;LOAD ZEROS INTO FT0
U 1017, 0878,AC ;12849          JSR [MOV.DATA]                    ;MOV FWR0 TO FWR2
U 1018, 890B,DC ;12850          JSR [LOOP.T17.1.5]                ;DO ADD OF 0 PLUS 0 WITH NO CARRY IN
                ;12851
U 1019, FF82,15 ;12852          INC LS[ERROR.NUMBER]              ;ERROR 2
U 101A, FDA2,15 ;12853          COM LS[DATA.1]                    ;FIRST FLOAT DATA OF ONES
U 101B, FDA4,15 ;12854          COM LS[DATA.2]                    ;SECOND FLOAT DATA OF ONES
U 101C, 7DA6,15 ;12855          COM LS[DATA.3]                    ;THIRD FLOAT DATA OF ONES
U 101D, 7D18,15 ;12856          COM LS[T12]                       ;EXPECTED DATA = ALL 1'S IN FIRST AND
                ;12857                                            ; SECOND FLOAT
U 101E, FD1A,15 ;12858          COM LS[T13]                       ;EXPECTED DATA = ALL 1'S IN THIRD FLOAT
U 101F, 890B,DC ;12859          JSR [LOOP.T17.1.5]                ;DO ADD OF 0 PLUS ALL 1'S WITH NO CARRY IN
                ;12860
U 1020, FF82,15 ;12861          INC LS[ERROR.NUMBER]              ;ERROR 3
U 1021, 0878,FC ;12862          JSR [LOAD.TRIPLE]                 ;LOAD FWR 0 WITH ALL ONES
U 1022, 0878,AC ;12863          JSR [MOV.DATA]                    ;MOV FWR[0] TO FWR[2]
U 1023, E5A2,15 ;12864          CLR LS[DATA.1]                    ;DATA OF 0'S IN FIRST FLOAT
U 1024, E5A4,15 ;12865          CLR LS[DATA.2]                    ;DATA OF 0'S IN SECOND FLOAT
```

```
U 1025, 65A6,15 ;12866          CLR LS[DATA.3]                  ;DATA OF 0'S IN THIRD FLOAT
U 1026, 890B,DC ;12867          JSR [LOOP.T17.1.5]              ;DO ADD OF ALL 1'S PLUS 0 WITH NO CARRY IN
                ;12868
U 1027, FF82,15 ;12869          INC LS[ERROR.NUMBER]            ;ERROR 4
U 1028, 3650,15 ;12870          MOV LS[BIT8] TO WR[0]           ;SET BIT 8 IN WR
U 1029, EF1A,15 ;12871          XOR WR[0] TO LS[T13]            ;EXPECTED DATA IN THIRD FLOAT (EXTEN-
                ;12872                                          ; SION PATH) = FE (LSB CLEAR)
U 102A, FDA2,15 ;12873          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 102B, FDA4,15 ;12874          COM LS[DATA.2]                  ;DATA OF 1'S IN SECOND FLOAT
U 102C, 7DA6,15 ;12875          COM LS[DATA.3]                  ;DATA OF 1'S IN THIRD FLOAT
U 102D, 890B,DC ;12876          JSR [LOOP.T17.1.5]              ;DO ADD OF 1'S PLUS 1'S WITH NO CARRY IN
                ;12877
U 102E, FF82,15 ;12878          INC LS[ERROR.NUMBER]            ;ERROR 5
U 102F, E5F8,15 ;12879          CLR LS[OS]                      ;CLEAR INDEX
                ;12880  REPEAT.T17.5:
U 1030, 36F6,15 ;12881          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 1031, 3EA2,15 ;12882          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 1032, 3EA4,15 ;12883          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 1033, BEA6,15 ;12884          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 1034, A3C0,15 ;12885          ROL WR[0]                       ;EXPECTED DATA WILL BE NEXT BIT
U 1035, BE18,15 ;12886          MOV WR[0] TO LS[T12]            ;EXPECTED DATA FOR FIRST AND SECOND
                ;12887                                          ; FLOAT
U 1036, C550,15 ;12888          BIC LS[BIT8] TO WR[0]           ;CLEAR BIT 8 (THIS IS LSB OF EXTENSION
                ;12889                                          ; PATH WHICH IS NEVER SET WITHOUT A
                ;12890                                          ; CARRY IN WITH THIS PATTERN)
U 1037, 3E1A,15 ;12891          MOV WR[0] TO LS[T13]            ;EXPECTED DATA FOR THIRD FLOAT
U 1038, 0878,FC ;12892          JSR [LOAD.TRIPLE]               ;LOAD SHIFT PATTERN INTO FWR[0]
U 1039, 0878,AC ;12893          JSR [MOV.DATA]                  ;MOV PATTERN TO FWR[2]
U 103A, 890B,DC ;12894          JSR [LOOP.T17.1.5]              ;GO ADD SHIFTING 1'S TO SHIFTING 1'S
U 103B, 7FF8,15 ;12895          INC LS[OS]                      ;INCREMENT INDEX
U 103C, B6F8,15 ;12896          MOV LS[OS] TO WR[0]             ;GET INDEX
                ;12897          CMP WR[0] WITH LS[#20],         ;DONE ALL PATTERNS?
U 103D, CF4A,35 ;12898           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 103E, 0903,01 ;12899          JMP.IF[NEQ] TO [REPEAT.T17.5]   ;REPEAT IF NOT
                ;12900
U 103F, FF82,15 ;12901          INC LS[ERROR.NUMBER]            ;ERROR 6
U 1040, 6518,15 ;12902          CLR LS[T12]                     ;EXPECTED DATA = 0'S IN FIRST AND
                ;12903                                          ; SECOND FLOAT
U 1041, 3650,15 ;12904          MOV LS[BIT8] TO WR[0]           ;SET BIT 8 IN WR
U 1042, 3E1A,15 ;12905          MOV WR[0] TO LS[T13]            ;EXPECTED DATA = 1 IN LSB OF EXTENSION
U 1043, 087C,1C ;12906          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 1044, 0878,AC ;12907          JSR [MOV.DATA]                  ;MOV FWR[0] TO FWR[2]
U 1045, 890D,BC ;12908          JSR [LOOP.T17.6.9]              ;DO ADD OF 0 PLUS 0 WITH CARRY IN SET
                ;12909
U 1046, FF82,15 ;12910          INC LS[ERROR.NUMBER]            ;ERROR 7
U 1047, FDA2,15 ;12911          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1048, FDA4,15 ;12912          COM LS[DATA.2]                  ;SECOND FLOAT DATA OF ONES
U 1049, 7DA6,15 ;12913          COM LS[DATA.3]                  ;THIRD FLOAT DATA OF ONES  EXCEPT BIT 7
U 104A, E51A,15 ;12914          CLR LS[T13]                     ;EXPECTED DATA = 0 IN THIRD FLOAT
U 104B, 890D,BC ;12915          JSR [LOOP.T17.6.9]              ;DO ADD OF 0 PLUS ALL 1'S WITH CARRY IN
                ;12916                                          ; SET
                ;12917
U 104C, FF82,15 ;12918          INC LS[ERROR.NUMBER]            ;ERROR 8
U 104D, 0878,FC ;12919          JSR [LOAD.TRIPLE]               ;LOAD FWR 0 WITH ALL ONES
U 104E, 0878,AC ;12920          JSR [MOV.DATA]                  ;MOV FWR[0] TO FWR[2]
```

```
U 104F, E5A2,15 ;12921          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1050, E5A4,15 ;12922          CLR LS[DATA.2]                  ;DATA OF 0'S IN SECOND FLOAT
U 1051, 65A6,15 ;12923          CLR LS[DATA.3]                  ;DATA OF 0'S IN THIRD FLOAT
U 1052, 890D,BC ;12924          JSR [LOOP.T17.6.9]              ;DO ADD OF ALL 1'S PLUS 0 WITH CARRY IN
                ;12925
U 1053, FF82,15 ;12926          INC LS[ERROR.NUMBER]            ;ERROR 9
U 1054, 7D18,15 ;12927          COM LS[T12]                     ;EXPECTED DATA = ALL 1'S IN FIRST AND
                ;12928                                          ; SECOND FLOAT
U 1055, FD1A,15 ;12929          COM LS[T13]                     ;EXPECTED DATA = ALL 1'S IN THIRD FLOAT
U 1056, FDA2,15 ;12930          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 1057, FDA4,15 ;12931          COM LS[DATA.2]                  ;DATA OF 1'S IN SECOND FLOAT
U 1058, 7DA6,15 ;12932          COM LS[DATA.3]                  ;DATA OF 1'S IN THIRD FLOAT
U 1059, 890D,BC ;12933          JSR [LOOP.T17.6.9]              ;DO ADD OF 1'S PLUS 1'S WITH CARRY IN
                ;12934
                ;12935  ;
                ;12936  ;       TEMPORARY LS LOCATIONS USED:
                ;12937  ;
                ;12938  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;12939  ; T9    MAIN MEMORY POINTER
                ;12940  ; T10   ADDRESS OF 'ADD FQ TO FWR[FT0] (FIRST WORD)
                ;12941  ;                  'ADD EWR[ZERO] TO EWR[ET0] + FCOUT' (SECOND WORD)
                ;12942  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;12943  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FQ'
                ;12944  ; T13   ADDRESS OF 'CLR FQ'
                ;12945  ; T14   ADDRESS OF 'MOV EWR[ET0] TO EWR[ZERO]'
                ;12946  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;12947  ;
                ;12948
U 105A, FF82,15 ;12949          INC LS[ERROR.NUMBER]            ;ERROR A
U 105B, B724,15 ;12950          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 105C, C75E,15 ;12951          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 105D, C54E,15 ;12952          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 105E, 3E8A,15 ;12953          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 105F, 37EA,15 ;12954          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;12955                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 1060, 3E10,15 ;12956          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 1061, 8870,3C ;12957          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'ADD' ROUTINE
U 1062, BE14,15 ;12958          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 1063, 37EC,15 ;12959          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;12960                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 1064, 3E16,15 ;12961          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 1065, 8870,3C ;12962          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'MOV FWR[FT0]
                ;12963                                          ; TO FQ' INSTRUCTION
U 1066, BE18,15 ;12964          MOV WR[0] TO LS[T12]            ;STORE IN LS T12
U 1067, 8870,3C ;12965          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'CLR FQ'
                ;12966                                          ; INSTRUCTION
U 1068, 3E1A,15 ;12967          MOV WR[0] TO LS[T13]            ;STORE IN LS T13
U 1069, 8870,3C ;12968          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'MOV EWR[ET0]
                ;12969                                          ; TO EWR[ZERO]' INSTRUCTION
U 106A, 3E1C,15 ;12970          MOV WR[0] TO LS[T14]            ;STORE IN LS T14
                ;12971
U 106B, B7EE,15 ;12972          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;12973                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 106C, BE1E,15 ;12974          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
                ;12975
```

```
U 106D. E5A2,15 ;12976          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 106E. E5A4,15 ;12977          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 106F. 65A6,15 ;12978          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 1070. 087D,CC ;12979          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1071. 0878,AC ;12980          JSR [MOV.DATA]                  ;MOV EWR0 TO EWR[ZERO]
U 1072. 890F,6C ;12981          JSR [LOOP.T17.A.E]              ;DO ADD OF 0 PLUS 0 WITH NO CARRY IN
                ;12982
U 1073. FF82,15 ;12983          INC LS[ERROR.NUMBER]            ;ERROR B
U 1074. FDA2,15 ;12984          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1075. FDA4,15 ;12985          COM LS[DATA.2]                  ;ONES IN EXPECTED LOWER 8 BITS OF EXP
U 1076. 7DA6,15 ;12986          COM LS[DATA.3]                  ;ONES IN EXPECTED UPPER 8 BITS OF EXP
U 1077. 890F,6C ;12987          JSR [LOOP.T17.A.E]              ;DO ADD OF 0 PLUS ALL 1'S WITH NO CARRY IN
                ;12988
U 1078. FF82,15 ;12989          INC LS[ERROR.NUMBER]            ;ERROR C
U 1079. 087D,CC ;12990          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 107A. 0878,AC ;12991          JSR [MOV.DATA]                  ;MOV EWR0 TO EWR[ZERO] (ALL 1'S)
U 107B. E5A2,15 ;12992          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 107C. 890F,6C ;12993          JSR [LOOP.T17.A.E]              ;DO ADD OF ALL 1'S PLUS 0 WITH NO CARRY IN
                ;12994
U 107D. FF82,15 ;12995          INC LS[ERROR.NUMBER]            ;ERROR D
U 107E. FDA2,15 ;12996          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 107F. 364E,15 ;12997          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 IN WR
U 1080. EFA4,15 ;12998          XOR WR[0] TO LS[DATA.2]         ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 1081. 890F,6C ;12999          JSR [LOOP.T17.A.E]              ;DO ADD OF 1'S PLUS 1'S WITH NO CARRY IN
                ;13000
U 1082. FF82,15 ;13001          INC LS[ERROR.NUMBER]            ;ERROR E
U 1083. B64C,15 ;13002          MOV LS[#40] TO WR[0]            ;PUT AN 40 IN WR
U 1084. 2100,15 ;13003          DEC WR[0]                       ;3F(H) NOW IN WR
U 1085. 3EF8,15 ;13004          MOV WR[0] TO LS[OS]             ;START INDEX AT 3F (POINTS TO BIT 31
                ;13005                                          ; SET).  BIT 31 WILL END UP IN LSB OF
                ;13006                                          ; EXPONENT.
U 1086. 364E,15 ;13007          MOV LS[BIT7] TO WR[0]           ;FIRST EXPECTED DATA BEFORE IT IS ROTATED
U 1087. BE12,15 ;13008          MOV WR[0] TO LS[T9]             ;STORE IN LS T9
                ;13009  REPEAT.T17.E:
U 1088. B6F6,95 ;13010          MOV LS[SHIFT.OS(4-0)] TO WR[1]  ;LOAD DATA PATTERN IN WR
U 1089. BEA2,95 ;13011          MOV WR[1] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 108A. 3612,15 ;13012          MOV LS[T9] TO WR[0]             ;GET LAST EXPECTED DATA
U 108B. A3C0,15 ;13013          ROL WR[0]                       ;EXPECTED DATA WILL BE NEXT BIT SET
                ;13014          CMP LS[BIT6] WITH WR[1],        ;SEE IF DATA PATTERN WILL SET LSB IN
                ;13015                                          ; UPPER 8 BITS OF EXPONENT
U 108C. CE4C,B5 ;13016           DT(LONG)&SET.ALU.CC            ; AND SET CONDITION CODES
U 108D. 8909,69 ;13017          JMP.IF[EQL] TO [T17.SET.BIT.7]  ;SET LSB OF EXPECTED DATA IF SO
                ;13018          CMP LS[BIT31] WITH WR[1],       ;FIRST PATTERN IS WITH BIT 31 SET. SKIP
                ;13019                                          ; GEQU CHECK ON FIRST PATTERN
U 108E. 4E7E,B5 ;13020           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 108F. 0909,29 ;13021          JMP.IF[EQL] TO [FIRST.TIME.T17] ;SKIP NEXT CHECK IF FIRST PATTERN
                ;13022          CMP WR[1] WITH LS[BIT7],        ;SEE IF DATA PATTERN WILL SET ANY BITS
                ;13023                                          ; IN UPPER 8 BITS OF EXP
U 1090. CF4E,B5 ;13024           DT(LONG)&SET.ALU.CC            ; AND SET CONDITION CODES
U 1091. 0909,73 ;13025          JMP.IF[GEQU] TO [T17.UPPER.BITS];JUMP TO SET UP DATA PATTERN FOR UPPER
                ;13026                                          ; BITS OF EXPONENT IF PATTERN IS 40(H)
                ;13027                                          ; OR GREATER
                ;13028  FIRST.TIME.T17:
U 1092. 3EA4,15 ;13029          MOV WR[0] TO LS[DATA.2]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 1093. BE12,15 ;13030          MOV WR[0] TO LS[T9]             ;SAVE CURRENT PATTERN IN LS T9
```

```
U 1094, 65A6,15 ;13031          CLR LS[DATA.3]                  ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1095, 0909,A4 ;13032          JMP [TEST.T17.E]                ;GO TEST PATTERN
                ;13033  T17.SET.BIT.7:
U 1096, 364E,15 ;13034          MOV LS[BIT7] TO WR[0]           ;SET LOW BIT OF EXPECTED DATA AGAIN
                ;13035  T17.UPPER.BITS:
U 1097, E5A4,15 ;13036          CLR LS[DATA.2]                  ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 1098, BEA6,15 ;13037          MOV WR[0] TO LS[DATA.3]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1099, BE12,15 ;13038          MOV WR[0] TO LS[T9]             ;SAVE CURRENT PATTERN IN LS T9
                ;13039  TEST.T17.E:
U 109A, 087D,CC ;13040          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 109B, 0878,AC ;13041          JSR [MOV.DATA]                  ;MOV PATTERN TO EWR[ZERO]
U 109C, 890F,6C ;13042          JSR [LOOP.T17.A.E]              ;GO ADD SHIFTING 1'S TO SHIFTING 1'S
U 109D, 7FF8,15 ;13043          INC LS[OS]                      ;INCREMENT INDEX
U 109E, B64C,15 ;13044          MOV LS[#40] TO WR[0]            ;GET DATA OF 40
U 109F, 4748,15 ;13045          BIS LS[#10] TO WR[0]            ;NOW HAVE 50 IN WR
                ;13046          CMP WR[0] WITH LS[OS],          ;DONE ALL PATTERNS?
U 10A0, CFF8,35 ;13047           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 10A1, 0908,81 ;13048          JMP.IF[NEQ] TO [REPEAT.T17.E]   ;REPEAT IF NOT
                ;13049
U 10A2, FF82,15 ;13050          INC LS[ERROR.NUMBER]            ;ERROR F
U 10A3, B67C,15 ;13051          MOV LS[BIT30] TO WR[0]          ;SET BIT 30 IN WR
U 10A4, 3EA2,15 ;13052          MOV WR[0] TO LS[DATA.1]         ;DATA OF ZEROS IN FIRST FLOAT EXCEPT
                ;13053                                          ; BIT 30.  BIT 30 ENDS UP IN MSB OF
                ;13054                                          ; FRACTION PATH TO PROVIDE CARRY OUT
U 10A5, 364E,15 ;13055          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 IN WR
U 10A6, 3EA4,15 ;13056          MOV WR[0] TO LS[DATA.2]         ;EXPECTED DATA = 1 IN LSB OF LOWER 8
                ;13057                                          ; BITS OF EXPONENT
U 10A7, 65A6,15 ;13058          CLR LS[DATA.3]                  ;EXPECTED DATA OF 0'S IN UPPER 8 BITS
                ;13059                                          ; OF EXPONENT
                ;13060
U 10A8, 087D,CC ;13061          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 10A9, 0878,AC ;13062          JSR [MOV.DATA]                  ;MOV PATTERN TO EWR[ZERO]
U 10AA, 8910,EC ;13063          JSR [LOOP.T17.F.12]             ;DO ADD OF 0 PLUS 0 WITH CARRY IN SET
                ;13064
U 10AB, FF82,15 ;13065          INC LS[ERROR.NUMBER]            ;ERROR 10
U 10AC, B69E,15 ;13066          MOV LS[ONES] TO WR[0]           ;PUT ONES IN WR 0
U 10AD, 3EA2,15 ;13067          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA OF ONES
U 10AE, E5A4,15 ;13068          CLR LS[DATA.2]                  ;EXPECTED DATA OF 0'S IN EXP
U 10AF, 8910,EC ;13069          JSR [LOOP.T17.F.12]             ;DO ADD OF 0 PLUS ALL 1'S WITH CARRY IN
                ;13070                                          ; SET
                ;13071
U 10B0, FF82,15 ;13072          INC LS[ERROR.NUMBER]            ;ERROR 11
U 10B1, 087D,CC ;13073          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 10B2, 0878,AC ;13074          JSR [MOV.DATA]                  ;MOV PATTERN TO EWR[ZERO]
U 10B3, B67C,15 ;13075          MOV LS[BIT30] TO WR[0]          ;DATA OF 0'S EXCEPT BIT 30
U 10B4, 3EA2,15 ;13076          MOV WR[0] TO LS[DATA.1]         ;DATA PATTERN OF 0'S EXCEPT BIT 30 WHICH
                ;13077                                          ; GOES IN MSB OF FRAC PATH
U 10B5, 8910,EC ;13078          JSR [LOOP.T17.F.12]             ;DO ADD OF ALL 1'S PLUS 0 WITH CARRY IN
                ;13079
U 10B6, FF82,15 ;13080          INC LS[ERROR.NUMBER]            ;ERROR 12
U 10B7, B69E,15 ;13081          MOV LS[ONES] TO WR[0]           ;PUT ONES IN WR 0
U 10B8, 3EA2,15 ;13082          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA OF ONES
U 10B9, FDA4,15 ;13083          COM LS[DATA.2]                  ;EXPECTED DATA OF 1'S IN EXPONENT
U 10BA, 7DA6,15 ;13084          COM LS[DATA.3]                  ;DITTO
U 10BB, 8910,EC ;13085          JSR [LOOP.T17.F.12]             ;DO ADD OF 1'S PLUS 1'S WITH CARRY IN
```

```
                    ;13086
U 10BC, 0912,64     ;13087          JMP [END.T17]                   ;DONE WITH ADD TEST
                    ;13088
                    ;13089  LOOP.T17.1.5:
U 10BD, 6588,15     ;13090          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 10BE, FF88,15     ;13091          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 10BF, 0878,FC     ;13092          JSR [LOAD.TRIPLE]               ;LOAD SECOND OPERAND  OF ADD IN FWR[FTO]
U 10C0, B610,15     ;13093          MOV LS[T8] TO WR[0]             ;GET ADDRESS OF FPA 'CLR FWR1] INSTRUCTION
U 10C1, B716,95     ;13094          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 10C2, 90F6,15     ;13095          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'CLR FWR[FT1].  THIS WILL CLEAR
                    ;13096                                          ; CARRY IN BIT
U 10C3, 10F6,95     ;13097          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 10C4, 3614,15     ;13098          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'ADD' INSTRUCTION
U 10C5, 90F6,15     ;13099          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD FWR[3] PLUS FWR[1] TO FQ'
                    ;13100                                          ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 10C6, DB00,15     ;13101          NOP                             ;FPA WILL DO 'ADD FWR[2] TO FWR[0] +
                    ;13102                                          ; FCOUT DURING THIS NOP
U 10C7, 10F6,95     ;13103          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 10C8, 087B,5C     ;13104          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                    ;13105                                          ; AND THIRD.FLT
U 10C9, BE8B,15     ;13106          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                    ;13107                                          ; AND EXPONENT (BITS 7-14)
U 10CA, B6A8,15     ;13108          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 10CB, B618,95     ;13109          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 10CC, 0869,3C     ;13110          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10CD, 090B,D4     ;13111          JMP [LOOP.T17.1.5]              ;ERROR LOOP IF ENABLED
                    ;13112
U 10CE, FF88,15     ;13113          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 10CF, E58A,15     ;13114          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 10D0, 36AA,15     ;13115          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 10D1, B618,95     ;13116          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 10D2, 0869,3C     ;13117          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10D3, 090B,D4     ;13118          JMP [LOOP.T17.1.5]              ;ERROR LOOP IF ENABLED
                    ;13119
U 10D4, FF88,15     ;13120          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 10D5, 3E8B,95     ;13121          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 10D6, 36AC,15     ;13122          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 10D7, 361A,95     ;13123          MOV LS[T13] TO WR[1]            ;EXPECTED DATA
U 10D8, 0869,3C     ;13124          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10D9, 090B,D4     ;13125          JMP [LOOP.T17.1.5]              ;ERROR LOOP IF ENABLED
U 10DA, 5B00,14     ;13126          RETURN                          ;DONE WITH THIS DATA PATTERN
                    ;13127
                    ;13128  LOOP.T17.6.9:
U 10DB, 6588,15     ;13129          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 10DC, FF88,15     ;13130          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 10DD, 0878,FC     ;13131          JSR [LOAD.TRIPLE]               ;LOAD SECOND OPERAND  OF ADD IN FWR[FTO]
U 10DE, 3614,15     ;13132          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'ADD' INSTRUCTION
U 10DF, B716,95     ;13133          MOV LS[#300] TO WR[1]           ;GET ADDRESS OF FPA LOOP SELF INSTRUCTION
U 10E0, 90F6,15     ;13134          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD FWR[3] PLUS FWR[1] TO FQ'
                    ;13135                                          ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 10E1, DB00,15     ;13136          NOP                             ;FPA WILL DO 'ADD FWR[2] TO FWR[0] +
                    ;13137                                          ; FCOUT DURING THIS NOP
U 10E2, 10F6,95     ;13138          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 10E3, 087B,5C     ;13139          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                    ;13140                                          ; AND THIRD.FLT
```

```
U 10E4, BE8B,15 ;13141          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;13142                                          ; AND EXPONENT (BITS 7-14)
U 10E5, B6A8,15 ;13143          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 10E6, B618,95 ;13144          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 10E7, 0869,3C ;13145          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10E8, 090D,B4 ;13146          JMP [LOOP.T17.6.9]              ;ERROR LOOP IF ENABLED
                ;13147
U 10E9, FF88,15 ;13148          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 10EA, E58A,15 ;13149          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 10EB, 36AA,15 ;13150          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 10EC, B618,95 ;13151          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 10ED, 0869,3C ;13152          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10EE, 090D,B4 ;13153          JMP [LOOP.T17.6.9]              ;ERROR LOOP IF ENABLED
                ;13154
U 10EF, FF88,15 ;13155          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 10F0, 3E8B,95 ;13156          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 10F1, 36AC,15 ;13157          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 10F2, 361A,95 ;13158          MOV LS[T13] TO WR[1]            ;EXPECTED DATA
U 10F3, 0869,3C ;13159          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 10F4, 090D,B4 ;13160          JMP [LOOP.T17.6.9]              ;ERROR LOOP IF ENABLED
U 10F5, 5B00,14 ;13161          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;13162
                ;13163  LOOP.T17.A.E:
U 10F6, 6588,15 ;13164          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 10F7, FF88,15 ;13165          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 10F8, 087D,CC ;13166          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 10F9, B61A,15 ;13167          MOV LS[T13] TO WR[0]            ;GET ADDRESS OF FPA 'CLR FQ' INSTRUCTION
U 10FA, B716,95 ;13168          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 10FB, 90F6,15 ;13169          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'CLR FQ' THIS WILL CLEAR CARRY
                ;13170                                          ; IN BIT
U 10FC, 10F6,95 ;13171          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 10FD, 3614,15 ;13172          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'ADD' INSTRUCTION
U 10FE, 90F6,15 ;13173          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD FQ TO FWR[FT0]
                ;13174                                          ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 10FF, DB00,15 ;13175          NOP                             ;FPA WILL DO 'ADD EWR[ZERO] TO EWR[0] +
                ;13176                                          ; FCOUT' DURING THIS NOP
U 1100, 10F6,95 ;13177          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1101, 087B,5C ;13178          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;13179                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 1102, B6A8,15 ;13180          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1103, 36A4,95 ;13181          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 1104, 0869,3C ;13182          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1105, 090F,64 ;13183          JMP [LOOP.T17.A.E]              ;ERROR LOOP IF ENABLED
                ;13184
U 1106, FF88,15 ;13185          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1107, 8877,4C ;13186          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;13187                                          ; LOWER 8 BITS
U 1108, 087B,5C ;13188          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1109, B6A8,15 ;13189          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 110A, B6A6,95 ;13190          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 110B, 0869,3C ;13191          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 110C, 090F,64 ;13192          JMP [LOOP.T17.A.E]              ;ERROR LOOP IF ENABLED
U 110D, 5B00,14 ;13193          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;13194
                ;13195  LOOP.T17.F.12:
```

```
U 110E, 6588,15 ;13196          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 110F, FF88,15 ;13197          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 EXP BITS FAILED
U 1110, 087D,CC ;13198          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 1111, 3618,15 ;13199          MOV LS[T12] TO WR[0]            ;GET ADDRESS OF FPA 'MOV FWR[FT0] TO
                ;13200                                          ; FQ' INSTRUCTION
U 1112, B716,95 ;13201          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 1113, 90F6,15 ;13202          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'MOV FWR[FT0] TO FQ' THIS WILL
                ;13203                                          ; SET CARRY IN BIT AFTER ADD
U 1114, 10F6,95 ;13204          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1115, 3614,15 ;13205          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'ADD' INSTRUCTION
U 1116, 90F6,15 ;13206          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD FQ TO FWR[FT0]
                ;13207                                          ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 1117, DB00,15 ;13208          NOP                             ;FPA WILL DO 'ADD EWR[ZERO] TO EWR[0] +
                ;13209                                          ; FCOUT' DURING THIS NOP
U 1118, 10F6,95 ;13210          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1119, 087B,5C ;13211          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;13212                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 111A, B6A8,15 ;13213          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 111B, 36A4,95 ;13214          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 111C, 0869,3C ;13215          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 111D, 0910,E4 ;13216          JMP [LOOP.T17.F.12]             ;ERROR LOOP IF ENABLED
                ;13217
U 111E, FF88,15 ;13218          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 111F, 8877,4C ;13219          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;13220                                          ; LOWER 8 BITS
U 1120, 087B,5C ;13221          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1121, B6A8,15 ;13222          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1122, B6A6,95 ;13223          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1123, 0869,3C ;13224          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1124, 0910,E4 ;13225          JMP [LOOP.T17.F.12]             ;ERROR LOOP IF ENABLED
U 1125, 5B00,14 ;13226          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;13227  END.T17:
```

```
;13228  .PAGE
;13229  .TOC    "TEST 18 S MINUS R FUNCTION TEST(M8389 FPA MODULE)"
;13230  ;++
;13231  ;
;13232  ;
;13233  ; Test Description:
;13234  ;
;13235  ;       The purpose of this test is to check the S-R function of the fraction
;13236  ;       and exponent data path. The instruction that will be used is SUB FWR[]
;13237  ;       FROM FWR[] + FCOUT and SUB EWR[] FROM EWR[].  The source is A and B,
;13238  ;       the function is S-R, and the destination is F->B.  CIN to the least
;13239  ;       significant bit of the exponent data path will always be set (it is
;13240  ;       forced by hardware) when a subtract is performed.  CIN to the fraction
;13241  ;       data path is controlled by a subtract of FWR[3] from FWR[1] just prior
;13242  ;       to executing the SUB FWR[2] FROM FWR[4] + FCOUT instruction.  This
;13243  ;       test will check the case where CIN is either set or clear for the
;13244  ;       fraction data path and where CIN is set for the exponent data paths.
;13245  ;       The exponent data path will be checked seperately due to the need to
;13246  ;       shift data into the upper 8 bits instead of loading them directly.  The
;13247  ;       data patterns that will be used are 0's and 0's, all 1's and 0's, 0's
;13248  ;       and all 1's, all 1's and all 1's, and shifting 1's and shifting 1's.
;13249  ;       These data patterns will be sufficeint to test all GEN and PROP logic.
;13250  ;
;13251  ; Assumptions:
;13252  ;
;13253  ;       It is assumed that all of the logic functions have been tested.
;13254  ;
;13255  ; Test Steps:
;13256  ;
;13257  ;       1) Clear FWR[FT3].  This will cause COUT to be asserted prior to the
;13258  ;          subtract of FWR[FT2] from FWR[FT4] + FCOUT.
;13259  ;       2) Load FWR[0] with all 0's in all bits and MOV to FWR[2].  Load FWR[0]
;13260  ;          with all 0's again.
;13261  ;       3) MOV FWR[0] to FWR[4], then clear FWR[1].
;13262  ;       4) Perform SUB FWR[3] FROM FWR[1] TO FQ.  Since FWR[3] is clear, FCOUT
;13263  ;          will be set.  Then execute SUB FWR[2] FROM FWR[4] + FCOUT.  MOV the
;13264  ;          result from FWR[4] to FWR[0].
;13265  ;       5) Store result from FWR[0] and check for 0's (0 - 0 with carry in set).
;13266  ;       6) Load FWR[0] with ones.  Repeat steps 3 and 4 with this data.
;13267  ;       7) Store result from FWR[0] and check for 1's (1's - 0 with carry in
;13268  ;          set).
;13269  ;       8) Load FWR[0] with 1's and move to FWR[2].  Repeat steps 3 and 4.
;13270  ;       9) Store result from FWR[0] and check for a 1 in the LSB of the exten-
;13271  ;          sion (0 - 1's with carry in set).
;13272  ;       10)Load FWR[0] with ones.  Repeat steps 3 and 4.
;13273  ;       11)Store result from FWR[0] and check for  0 (1's - 1's with carry in
;13274  ;          set).
;13275  ;       12)Load FWR[0] with a shifting 1 pattern and mov to FWR[2].  Reload
;13276  ;          FWR[0] with the same shifting data pattern, and repeat steps 3 and 4.
;13277  ;       13)Store result from FWR[0] and check result for 0.
;13278  ;       14)Repeat steps 2 through 11 except set the hidden bit in FWR[3] before
;13279  ;          executing the tests.  This will prevent FCOUT from asserting prior
;13280  ;          to the subtract.  Check result for correct data.
;13281  ;       15)Repeat steps 2 through 11, this time checking the exponent data path
;13282  ;          by shifting data into the upper 8 bits.  FCOUT is always set by the
```

```
;13283  ;          hardware.  EWR[5] and EWR[0] are used.
;13284  ;
;13285  ; Error:
;13286  ;
;13287  ;       Note:  For the following errors, the data printed under OTHER in the
;13288  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;13289  ;       FLT) which failed.
;13290  ;
;13291  ;       Error1 - SUB failed with data of 0's in FWR[2] and 0's in FWR[4],
;13292  ;                carry in set.
;13293  ;       Error2 - SUB failed with data of 0's in FWR[2] and 1's in FWR[4],
;13294  ;                carry in set.
;13295  ;       Error3 - SUB failed with data of 1's in FWR[2] and 0's in FWR[4],
;13296  ;                carry in set.
;13297  ;       Error4 - SUB failed with data of 1's in FWR[2] and 1's in FWR[4],
;13298  ;                carry in set.
;13299  ;       Error5 - SUB failed with data of shifting 1's in FWR[2] and shifting
;13300  ;                1's in FWR[4], carry in set.
;13301  ;       Error6 - SUB failed with data of 0's in FWR[2] and 0's in FWR[4], with
;13302  ;                no carry in.
;13303  ;       Error7 - SUB failed with data of 0's in FWR[2] and 1's in FWR[4], with
;13304  ;                no carry in.
;13305  ;       Error8 - SUB failed with data of 1's in FWR[2] and 0's in FWR[4], with
;13306  ;                no carry in.
;13307  ;       Error9 - SUB failed with data of 1's in FWR[2] and 1's in FWR[4], with
;13308  ;                no carry in.
;13309  ;
;13310  ;       Note:  For the following errors, the data printed under OTHER in the
;13311  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;13312  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;13313  ;
;13314  ;       ErrorA - SUB failed with data of 0's in EWR[5] and 0's in EWR[0],
;13315  ;                carry in set.
;13316  ;       ErrorB - SUB failed with data of 0's in EWR[5] and 1's in EWR[0],
;13317  ;                carry in set.
;13318  ;       ErrorC - SUB failed with data of 1's in EWR[5] and 0's in EWR[0],
;13319  ;                carry in set.
;13320  ;       ErrorD - SUB failed with data of 1's in EWR[5] and 1's in EWR[0],
;13321  ;                carry in set.
;13322  ;       ErrorE - SUB failed with data of shifting 1's in EWR[5] and shifting
;13323  ;                1's in EWR[0], carry in set.
;13324  ;
;13325  ;
;13326  ; Debug:
;13327  ;
;13328  ;       Error1: This error indicates a basic problem with either the control
;13329  ;               logic, the EXP DECODE PAL, or one or more of the 2901's.  The
;13330  ;               fraction control lines should be 011001001 during the SUB
;13331  ;               FWR[2] FROM FWR[4] + FCOUT insruction.  Of the control lines
;13332  ;               only the FUNC (I5 through I3) has not been previously tested.
;13333  ;               If the control lines are in error, trace them back to their
;13334  ;               source at the CS PROMS.
;13335  ;                 If the received data is all 1's, check that CIN EXT00 H is
;13336  ;               high during the SUB FWR[FT2] FROM FWR[FT4] + FCOUT instruction.
;13337  ;               If not, check the inputs to the EXP DECODE PAL for a low on
```

```
                ;13338  ;               ENB CLK5 L, a high on EXTEND CLOCK (1) H, and a high on FRAC
                ;13339  ;               COUT SAVE H.  If these inputs are OK, then the PAL is probably
                ;13340  ;               bad.
                ;13341  ;                 If FRAC COUT SAVE H is low, trace it back through the latch
                ;13342  ;               to the carry out output of the 2901 that produces FRAC COUT.
                ;13343  ;                 If a random bit or two is setting, the problem is probably
                ;13344  ;               in a 2901.  All CIN's to the 2901's should be high.
                ;13345  ;       Error2: Same as Error 1 except the most likely suspect is a 2901.  All
                ;13346  ;               CIN's to the 2901's should be low except CIN EXTOO H.
                ;13347  ;       Error3: Same as error2.
                ;13348  ;       Error4: Same as Error 1 except the most likely suspect is a 2901.  All
                ;13349  ;               CIN's should be asserted high on the inputs to all the 2901's.
                ;13350  ;       Error5: The most likely suspect is the 2901 with the failing bit.
                ;13351  ;       Error6: The most likely suspect is the EXP DECODE PAL.  This is the
                ;13352  ;               check on the FCOUT logic being deasserted.  Check FRAC COUT
                ;13353  ;               SAVE H going into the EXP DECODE PAL during the SUB FWR[FT2]
                ;13354  ;               FROM FWR[FT4] + FCOUT instruction.  The inputs should be the
                ;13355  ;               same as in Error 1 except FRAC COUT SAVE should be low and EXT
                ;13356  ;               CIN H should be low.  If the inputs are OK but EXT CIN H is not
                ;13357  ;               low, the PAL is probably bad.
                ;13358  ;                 If FRAC COUT SAVE H is high, trace it back through the latch
                ;13359  ;               to the 2901 that produces FRAC COUT H.  This should be low
                ;13360  ;               during the SUB FWR[FT3] FROM FWR[FT1] TO FQ instruction.
                ;13361  ;       Errors 7-9: Suspect the 2901 outputting the failing bit.
                ;13362  ;       Errors A-E: Same as Error 1-5 except the exponent path is being test-
                ;13363  ;               ed.
                ;13364  ;
                ;13365  ;--
                ;13366
                ;13367  ;
                ;13368  ;       TEMPORARY LS LOCATIONS USED:
                ;13369  ;
                ;13370  ; T8    ADDRESS OF 'MOV FWR[FT0] TO FWR[FT4]'
                ;13371  ; T9    MAIN MEMORY POINTER
                ;13372  ; T10   ADDRESS OF 'SUB FWR[FT3] FROM FWR[FT1] TO FQ' (FIRST WORD)
                ;13373  ;                  'SUB FWR[FT2] FROM FWR[FT4] + FCOUT' (SECOND WORD)
                ;13374  ; T11   ADDRESS OF 'MOV FWR[FT4] TO FWR[FT0]'
                ;13375  ; T12   EXPECTED DATA FOR FIRST AND SECOND FLOAT
                ;13376  ; T13   EXPECTED DATA FOR THIRD FLOAT
                ;13377  ; T14   ADDRESS OF MOV ABOVE IS PLACED HERE FOR MOV SUBROUTINE
                ;13378  ; T15   ADDRESS OF 'MOV FWR[FT0] TO FWR[FT2]'
                ;13379  ; T57   ADDRESS OF 'CLR FWR[FT1]'
                ;13380  ;
                ;13381  T.18:
                ;13382
U 1126, B65E,15 ;13383          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;13384                                          ; STATUS WORD
U 1127, 3E80,15 ;13385          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;13386                                          ; BIT 15 INDICATES START OF TEST TO
                ;13387                                          ; CONSOLE
U 1128, 10E0,15 ;13388          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;13389  WAIT.T18.0:
U 1129, 0912,94 ;13390          JMP [WAIT.T18.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;13391                                          ; RESPOND
U 112A, 087E,6C ;13392          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
```

```
                ;13393                                  ; AND SIZE BITS
U 112B, B670,15 ;13394          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;13395                                  ; OTHER DATA
U 112C, 3E80,15 ;13396          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 112D, B7C6,15 ;13397          MOV LS[T18.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 112E, BE12,15 ;13398          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 112F, 8870,3C ;13399          JSR [L0]                        ;GET MEMORY DATA INTO WR 0
U 1130, B716,95 ;13400          MOV LS[#300] TO WR[1]           ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 1131, 90F6,15 ;13401          MISC2 [TRAP.ACC] WR[0]          ;EXECUTE 'CLR FWR[FT3]'.  THIS WILL
                ;13402                                  ; CREATE A CIN IN SUBSEQUENT
                ;13403                                  ; SUBTRACTS
U 1132, 10F6,95 ;13404          MISC2 [TRAP.ACC] WR[1]          ;EXECUTE LOOP SELF
U 1133, 8870,3C ;13405          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 1134, 3E10,15 ;13406          MOV WR[0] TO LS[T8]             ;STORE 'MOV FWR[0] TO FWR[4]' ADDRESS
                ;13407                                  ; IN LS
U 1135, 8870,3C ;13408          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 1136, BE14,15 ;13409          MOV WR[0] TO LS[T10]            ;PUT FPA ADDRESS IN LS FOR 'SUB' INSTRUCTION
U 1137, 8870,3C ;13410          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 1138, 3E16,15 ;13411          MOV WR[0] TO LS[T11]            ;STORE 'MOV FWR[4] TO FWR[0]' ADDRESS
                ;13412                                  ; IN LS
U 1139, 8870,3C ;13413          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 113A, BE1E,15 ;13414          MOV WR[0] TO LS[T15]            ;STORE 'MOV FWR[0] TO FWR[2]' ADDRESS
                ;13415                                  ; IN LS
U 113B, 8870,3C ;13416          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 113C, 3EAE,15 ;13417          MOV WR[0] TO LS[T57]            ;STORE 'CLR FWR[FT1]' ADDRESS IN LS
U 113D, 3725,15 ;13418          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 113E, A0C5,15 ;13419          COM WR[2]                       ;BITS 8-15 = 1
U 113F, C74F,15 ;13420          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;13421                                  ; BIT AND EXPONENT)
U 1140, B725,95 ;13422          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;13423
U 1141, 6518,15 ;13424          CLR LS[T12]                     ;EXPECTED DATA = 0'S IN FIRST AND
                ;13425                                  ; SECOND FLOAT
U 1142, E51A,15 ;13426          CLR LS[T13]                     ;EXPECTED DATA = 0'S IN THIRD FLOAT
U 1143, 087C,1C ;13427          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 1144, 361E,15 ;13428          MOV LS[T15] TO WR[0]            ;GET ADDRESS OF MOV FWR0 TO FWR2
U 1145, 3E1C,15 ;13429          MOV WR[0] TO LS[T14]            ;PUT IN LS FOR MOV SUBROUTINE
U 1146, 0878,AC ;13430          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
U 1147, 891C,EC ;13431          JSR [LOOP.T18.1.9]              ;DO SUB OF 0 FROM 0 WITH CARRY IN SET
                ;13432
U 1148, FF82,15 ;13433          INC LS[ERROR.NUMBER]            ;ERROR 2
U 1149, 7D18,15 ;13434          COM LS[T12]                     ;EXPECTED DATA OF 1'S IN FIRST AND
                ;13435                                  ; SECOND FLOAT
U 114A, FD1A,15 ;13436          COM LS[T13]                     ;EXPECTED DATA OF 1'S IN THIRD FLOAT
U 114B, FDA2,15 ;13437          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 114C, FDA4,15 ;13438          COM LS[DATA.2]                  ;SECOND FLOAT DATA OF ONES
U 114D, 7DA6,15 ;13439          COM LS[DATA.3]                  ;THIRD FLOAT DATA OF ONES
U 114E, 891C,EC ;13440          JSR [LOOP.T18.1.9]              ;DO SUB OF 0 FROM ALL 1'S WITH CARRY IN
                ;13441                                  ; SET
                ;13442
U 114F, FF82,15 ;13443          INC LS[ERROR.NUMBER]            ;ERROR 3
U 1150, 0878,FC ;13444          JSR [LOAD.TRIPLE]               ;LOAD FWR 0 WITH ALL ONES
U 1151, 361E,15 ;13445          MOV LS[T15] TO WR[0]            ;GET ADDRESS OF MOV FWR0 TO FWR2
U 1152, 3E1C,15 ;13446          MOV WR[0] TO LS[T14]            ;PUT IN LS FOR MOV SUBROUTINE
U 1153, 0878,AC ;13447          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
```

```
U 1154, 6518,15 ;13448          CLR LS[T12]                     ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13449                                          ; SECOND FLOAT
U 1155, 3650,15 ;13450          MOV LS[BIT8] TO WR[0]           ;SET BIT 8 IN WR
U 1156, 3E1A,15 ;13451          MOV WR[0] TO LS[T13]            ;EXPECTED DATA = LSB SET IN THIRD FLOAT
                ;13452                                          ; (EXTENSION)
U 1157, E5A2,15 ;13453          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1158, E5A4,15 ;13454          CLR LS[DATA.2]                  ;DATA OF 0'S IN SECOND FLOAT
U 1159, 65A6,15 ;13455          CLR LS[DATA.3]                  ;DATA OF 0'S IN THIRD FLOAT
U 115A, 891C,EC ;13456          JSR [LOOP.T18.1.9]              ;DO SUB OF ALL 1'S FROM 0 WITH CARRY IN
                ;13457                                          ; SET
                ;13458
U 115B, FF82,15 ;13459          INC LS[ERROR.NUMBER]            ;ERROR 4
U 115C, 6518,15 ;13460          CLR LS[T12]                     ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13461                                          ; SECOND FLOAT
U 115D, E51A,15 ;13462          CLR LS[T13]                     ;EXPECTED DATA OF 0'S IN THIRD FLOAT
U 115E, FDA2,15 ;13463          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 115F, FDA4,15 ;13464          COM LS[DATA.2]                  ;DATA OF 1'S IN SECOND FLOAT
U 1160, 7DA6,15 ;13465          COM LS[DATA.3]                  ;DATA OF 1'S IN THIRD FLOAT
U 1161, 891C,EC ;13466          JSR [LOOP.T18.1.9]              ;DO SUB OF 1'S FROM 1'S WITH CARRY IN
                ;13467                                          ; SET
                ;13468
U 1162, FF82,15 ;13469          INC LS[ERROR.NUMBER]            ;ERROR 5
U 1163, E5F8,15 ;13470          CLR LS[OS]                      ;CLEAR INDEX
U 1164, 6518,15 ;13471          CLR LS[T12]                     ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13472                                          ; SECOND FLOAT
U 1165, E51A,15 ;13473          CLR LS[T13]                     ;EXPECTED DATA OF 0'S IN THIRD FLOAT
                ;13474  REPEAT.T18.5:
U 1166, 36F6,15 ;13475          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 1167, 3EA2,15 ;13476          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 1168, 3EA4,15 ;13477          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 1169, BEA6,15 ;13478          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 116A, 0878,FC ;13479          JSR [LOAD.TRIPLE]               ;LOAD SHIFT PATTERN INTO FWR[0]
U 116B, 361E,15 ;13480          MOV LS[T15] TO WR[0]            ;GET ADDRESS OF MOV FWR0 TO FWR2
U 116C, 3E1C,15 ;13481          MOV WR[0] TO LS[T14]            ;PUT IN LS FOR MOV SUBROUTINE
U 116D, 0878,AC ;13482          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
U 116E, 891C,EC ;13483          JSR [LOOP.T18.1.9]              ;GO SUB SHIFTING 1'S FROM SHIFTING 1'S
                ;13484                                          ; WITH CARRY IN SET
U 116F, 7FF8,15 ;13485          INC LS[OS]                      ;INCREMENT INDEX
U 1170, B6F8,15 ;13486          MOV LS[OS] TO WR[0]             ;GET INDEX
                ;13487          CMP WR[0] WITH LS[#20],         ;DONE ALL PATTERNS?
U 1171, CF4A,35 ;13488           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 1172, 8916,61 ;13489          JMP.IF[NEQ] TO [REPEAT.T18.5]   ;REPEAT IF NOT
                ;13490
U 1173, FF82,15 ;13491          INC LS[ERROR.NUMBER]            ;ERROR 6
U 1174, B69E,15 ;13492          MOV LS[ONES] TO WR[0]           ;PUT ALL 1'S IN WR 0
U 1175, BE18,15 ;13493          MOV WR[0] TO LS[T12]            ;EXPECTED DATA = 1'S IN FIRST AND
                ;13494                                          ; SECOND FLOAT
U 1176, 3E1A,15 ;13495          MOV WR[0] TO LS[T13]            ;EXPECTED DATA = 1'S IN THIRD FLOAT
U 1177, 087C,1C ;13496          JSR [CLR.FT0]                   ;LOAD FWR[FT0] WITH 0'S (HIDDEN BIT IS
                ;13497                                          ; SET IN BOTH FT1 AND FT0)
U 1178, 8870,3C ;13498          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF MOV FWR[0]
                ;13499                                          ; TO FWR[3]
U 1179, 90F6,15 ;13500          MISC2 [TRAP.ACC] WR[0]          ;FORCE ADDRESS OF MOV FWR[0] TO FWR[3]
                ;13501                                          ; THIS WILL PREVENT A CARRY IN
U 117A, 361E,15 ;13502          MOV LS[T15] TO WR[0]            ;GET ADDRESS OF MOV FWR0 TO FWR2
```

```
U 117B, 3E1C,15 ;13503          MOV WR[0] TO LS[T14]             ;PUT IN LS FOR MOV SUBROUTINE
U 117C, 0878,AC ;13504          JSR [MOV.DATA]                   ;MOV FWR0 TO FWR2
U 117D, 891C,EC ;13505          JSR [LOOP.T18.1.9]               ;DO SUB OF 0 FROM 0 WITH NO CARRY IN
                ;13506
U 117E, FF82,15 ;13507          INC LS[ERROR.NUMBER]             ;ERROR 7
U 117F, 5F50,15 ;13508          MCOM LS[BIT8] TO WR[0]           ;GET ALL 1'S EXCEPT FOR BIT 8 IN WR
U 1180, 3E1A,15 ;13509          MOV WR[0] TO LS[T13]             ;EXPECTED DATA OF 1'S EXCEPT FOR LSB
                ;13510                                           ; OF THIRD FLOAT (EXTENSION)
U 1181, FDA2,15 ;13511          COM LS[DATA.1]                   ;FIRST FLOAT DATA OF ONES
U 1182, FDA4,15 ;13512          COM LS[DATA.2]                   ;SECOND FLOAT DATA OF ONES
U 1183, 7DA6,15 ;13513          COM LS[DATA.3]                   ;THIRD FLOAT DATA OF ONES
U 1184, 891C,EC ;13514          JSR [LOOP.T18.1.9]               ;DO SUB OF 0 FROM ALL 1'S WITH NO CARRY
                ;13515                                           ; IN
                ;13516
U 1185, FF82,15 ;13517          INC LS[ERROR.NUMBER]             ;ERROR 8
U 1186, 0878,FC ;13518          JSR [LOAD.TRIPLE]                ;LOAD FWR 0 WITH ALL ONES
U 1187, 361E,15 ;13519          MOV LS[T15] TO WR[0]             ;GET ADDRESS OF MOV FWR0 TO FWR2
U 1188, 3E1C,15 ;13520          MOV WR[0] TO LS[T14]             ;PUT IN LS FOR MOV SUBROUTINE
U 1189, 0878,AC ;13521          JSR [MOV.DATA]                   ;MOV FWR0 TO FWR2
U 118A, 6518,15 ;13522          CLR LS[T12]                      ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13523                                           ; SECOND FLOAT
U 118B, E51A,15 ;13524          CLR LS[T13]                      ;EXPECTED DATA OF 0'S IN THIRD FLOAT
U 118C, E5A2,15 ;13525          CLR LS[DATA.1]                   ;DATA OF 0'S IN FIRST FLOAT
U 118D, E5A4,15 ;13526          CLR LS[DATA.2]                   ;DATA OF 0'S IN SECOND FLOAT
U 118E, 65A6,15 ;13527          CLR LS[DATA.3]                   ;DATA OF 0'S IN THIRD FLOAT
U 118F, 891C,EC ;13528          JSR [LOOP.T18.1.9]               ;DO SUB OF ALL 1'S FROM 0 WITH NO CARRY
                ;13529                                           ; IN
                ;13530
U 1190, FF82,15 ;13531          INC LS[ERROR.NUMBER]             ;ERROR 9
U 1191, 7D18,15 ;13532          COM LS[T12]                      ;EXPECTED DATA = ALL 1'S IN FIRST AND
                ;13533                                           ; SECOND FLOAT
U 1192, FD1A,15 ;13534          COM LS[T13]                      ;EXPECTED DATA = ALL 1'S IN THIRD FLOAT
U 1193, FDA2,15 ;13535          COM LS[DATA.1]                   ;DATA OF 1'S IN FIRST FLOAT
U 1194, FDA4,15 ;13536          COM LS[DATA.2]                   ;DATA OF 1'S IN SECOND FLOAT
U 1195, 7DA6,15 ;13537          COM LS[DATA.3]                   ;DATA OF 1'S IN THIRD FLOAT
U 1196, 891C,EC ;13538          JSR [LOOP.T18.1.9]               ;DO SUB OF 1'S FROM 1'S WITH NO CARRY IN
                ;13539
                ;13540  ;
                ;13541  ;       TEMPORARY LS LOCATIONS USED:
                ;13542  ;
                ;13543  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;13544  ; T9    MAIN MEMORY POINTER
                ;13545  ; T10   ADDRESS OF 'SUB EWR[ET5] FROM EWR[ET0]'
                ;13546  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;13547  ; T14   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET5]'
                ;13548  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;13549  ;
                ;13550
U 1197, FF82,15 ;13551          INC LS[ERROR.NUMBER]             ;ERROR A
U 1198, B724,15 ;13552          MOV LS[#FFFF00FF] TO WR[0]       ;BITS 8-15 = 0
U 1199, C75E,15 ;13553          BIS LS[BIT15] TO WR[0]           ;SET BIT 15
U 119A, C54E,15 ;13554          BIC LS[BIT7] TO WR[0]            ;CLEAR BIT 7
U 119B, 3E8A,15 ;13555          MOV WR[0] TO LS[ERROR.MASK]      ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 119C, 37EA,15 ;13556          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;13557                                           ; EWR[ET0] AND EQ' INSTRUCTION
```

```
U 119D, 3E10,15 ;13558          MOV WR[0] TO LS[T8]                ;STORE IN LS T8
U 119E, 8870,3C ;13559          JSR [L0]                           ;LOAD WR 0 WITH ADDRESS OF 'SUB' ROUTINE
U 119F, BE14,15 ;13560          MOV WR[0] TO LS[T10]               ;STORE IN LS T10
U 11A0, 37EC,15 ;13561          MOV LS[SHIFT.LEFT.EWR] TO WR[0]    ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;13562                                             ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 11A1, 3E16,15 ;13563          MOV WR[0] TO LS[T11]               ;STORE IN LS T11
U 11A2, 8870,3C ;13564          JSR [L0]                           ;LOAD WR 0 WITH ADDRESS OF 'MOV EWR[ET0]
                ;13565                                             ; TO EWR[ET5]' INSTRUCTION
U 11A3, 3E1C,15 ;13566          MOV WR[0] TO LS[T14]               ;STORE IN LS T14
                ;13567
U 11A4, B7EE,15 ;13568          MOV LS[SHIFT.LEFT.FWR] TO WR[0]    ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;13569                                             ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 11A5, BE1E,15 ;13570          MOV WR[0] TO LS[T15]               ;STORE IN LS T15
                ;13571
U 11A6, E5A2,15 ;13572          CLR LS[DATA.1]                     ;ZEROS IN FIRST FLOAT DATA
U 11A7, E5A4,15 ;13573          CLR LS[DATA.2]                     ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 11A8, 65A6,15 ;13574          CLR LS[DATA.3]                     ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 11A9, 087D,CC ;13575          JSR [LOAD.SHIFT.EWR0]              ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 11AA, 0878,AC ;13576          JSR [MOV.DATA]                     ;MOV EWR0 TO EWR[ET5]
U 11AB, 891F,2C ;13577          JSR [LOOP.T18.A.E]                 ;DO SUB OF 0 FROM 0 WITH CARRY IN SET
                ;13578
U 11AC, FF82,15 ;13579          INC LS[ERROR.NUMBER]               ;ERROR B
U 11AD, FDA2,15 ;13580          COM LS[DATA.1]                     ;FIRST FLOAT DATA OF ONES
U 11AE, FDA4,15 ;13581          COM LS[DATA.2]                     ;ONES IN EXPECTED LOWER 8 BITS OF EXP
U 11AF, 7DA6,15 ;13582          COM LS[DATA.3]                     ;ONES IN EXPECTED UPPER 8 BITS OF EXP
U 11B0, 891F,2C ;13583          JSR [LOOP.T18.A.E]                 ;DO SUB OF 0 FROM ALL 1'S WITH CARRY IN
                ;13584                                             ; SET
                ;13585
U 11B1, FF82,15 ;13586          INC LS[ERROR.NUMBER]               ;ERROR C
U 11B2, 087D,CC ;13587          JSR [LOAD.SHIFT.EWR0]              ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 11B3, 0878,AC ;13588          JSR [MOV.DATA]                     ;MOV EWR0 TO EWR[ET5] (ALL 1'S)
U 11B4, E5A2,15 ;13589          CLR LS[DATA.1]                     ;DATA OF 0'S IN FIRST FLOAT
U 11B5, 364E,15 ;13590          MOV LS[BIT7] TO WR[0]              ;SET BIT 7 IN WR
U 11B6, 3EA4,15 ;13591          MOV WR[0] TO LS[DATA.2]            ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 11B7, 65A6,15 ;13592          CLR LS[DATA.3]                     ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 11B8, 891F,2C ;13593          JSR [LOOP.T18.A.E]                 ;DO SUB OF ALL 1'S FROM 0 WITH CARRY IN
                ;13594                                             ; SET
                ;13595
U 11B9, FF82,15 ;13596          INC LS[ERROR.NUMBER]               ;ERROR D
U 11BA, FDA2,15 ;13597          COM LS[DATA.1]                     ;DATA OF 1'S IN FIRST FLOAT
U 11BB, E5A4,15 ;13598          CLR LS[DATA.2]                     ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 11BC, 891F,2C ;13599          JSR [LOOP.T18.A.E]                 ;DO SUB OF 1'S FROM 1'S WITH CARRY IN
                ;13600                                             ; SET
                ;13601
U 11BD, FF82,15 ;13602          INC LS[ERROR.NUMBER]               ;ERROR E
U 11BE, B64A,15 ;13603          MOV LS[#20] TO WR[0]               ;PUT AN 20 IN WR
U 11BF, 2100,15 ;13604          DEC WR[0]                          ;1F(H) NOW IN WR
U 11C0, 3EF8,15 ;13605          MOV WR[0] TO LS[OS]                ;START INDEX AT 1F (POINTS TO BIT 31
                ;13606                                             ; SET).  BIT 31 WILL END UP IN LSB OF
                ;13607                                             ; EXPONENT.
U 11C1, E5A4,15 ;13608          CLR LS[DATA.2]                     ;EXPECTED DATA WILL ALWAYS BE 0 IN LOWER
                ;13609                                             ; 8 BITS OF EXP
U 11C2, 65A6,15 ;13610          CLR LS[DATA.3]                     ;EXPECTED DATA WILL ALWAYS BE 0 IN UPPER
                ;13611                                             ; 8 BITS OF EXP
                ;13612  REPEAT.T18.E:
```

```
U 11C3, B6F6,95 ;13613          MOV LS[SHIFT.OS(4-0)] TO WR[1]  ;LOAD DATA PATTERN IN WR
U 11C4, BEA2,95 ;13614          MOV WR[1] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 11C5, 087D,CC ;13615          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 11C6, 0878,AC ;13616          JSR [MOV.DATA]                  ;MOV PATTERN TO EWR[ET5]
U 11C7, 891F,2C ;13617          JSR [LOOP.T18.A.E]              ;GO SUB SHIFTING 1'S FROM SHIFTING 1'S
U 11C8, 7FF8,15 ;13618          INC LS[OS]                      ;INCREMENT INDEX
U 11C9, B64A,15 ;13619          MOV LS[#20] TO WR[0]            ;GET DATA OF 20
U 11CA, 4748,15 ;13620          BIS LS[#10] TO WR[0]            ;NOW HAVE 30 IN WR
                ;13621          CMP WR[0] WITH LS[OS],          ;DONE ALL PATTERNS?
U 11CB, CFF8,35 ;13622           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 11CC, 891C,31 ;13623          JMP.IF[NEQ] TO [REPEAT.T18.E]   ;REPEAT IF NOT
                ;13624
U 11CD, 8920,64 ;13625          JMP [END.T18]                   ;DONE WITH SUB TEST
                ;13626
                ;13627  LOOP.T18.1.9:
U 11CE, 6588,15 ;13628          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 11CF, FF88,15 ;13629          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 11D0, 0878,FC ;13630          JSR [LOAD.TRIPLE]               ;LOAD SECOND OPERAND  OF SUB IN FWR[FT0]
U 11D1, B610,15 ;13631          MOV LS[T8] TO WR[0]             ;GET ADDRESS OF FPA 'MOV FWR[0] TO FWR
                ;13632                                          ; [4]' INSTRUCTION
U 11D2, 3E1C,15 ;13633          MOV WR[0] TO LS[T14]            ;STORE IN LS FOR MOV ROUTINE
U 11D3, 0878,AC ;13634          JSR [MOV.DATA]                  ;MOV FWR[0] TO FWR[4]
U 11D4, B6AE,15 ;13635          MOV LS[T57] TO WR[0]            ;ADDRESS OF 'CLR FWR[FT1]'
U 11D5, B716,95 ;13636          MOV LS[#300] TO WR[1]           ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 11D6, 90F6,15 ;13637          MISC2 [TRAP.ACC] WR[0]          ;EXECUTE 'CLR FWR[FT1]'.  THIS WILL
                ;13638                                          ; ALLOW CARRY IN TO BE SET CORRECTLY
U 11D7, 10F6,95 ;13639          MISC2 [TRAP.ACC] WR[1]          ;EXECUTE LOOP SELF
U 11D8, 3614,15 ;13640          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 11D9, 90F6,15 ;13641          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'SUB FWR[3] FROM FWR[1] TO FQ'
                ;13642                                          ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 11DA, DB00,15 ;13643          NOP                             ;FPA WILL DO 'SUB FWR[2] FROM FWR[4] +
                ;13644                                          ; FCOUT DURING THIS NOP
U 11DB, 10F6,95 ;13645          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 11DC, B616,15 ;13646          MOV LS[T11] TO WR[0]            ;GET ADDRESS OF MOV FWR[4] TO FWR[0]
U 11DD, 3E1C,15 ;13647          MOV WR[0] TO LS[T14]            ;STORE IN LS FOR MOV ROUTINE
U 11DE, 0878,AC ;13648          JSR [MOV.DATA]                  ;MOV RESULT TO FWR[0]
U 11DF, 087B,5C ;13649          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;13650                                          ; AND THIRD.FLT
U 11E0, BE8B,15 ;13651          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;13652                                          ; AND EXPONENT (BITS 7-14)
U 11E1, B6A8,15 ;13653          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 11E2, B618,95 ;13654          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 11E3, 0869,3C ;13655          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 11E4, 091C,E4 ;13656          JMP [LOOP.T18.1.9]              ;ERROR LOOP IF ENABLED
                ;13657
U 11E5, FF88,15 ;13658          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 11E6, E58A,15 ;13659          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 11E7, 36AA,15 ;13660          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 11E8, B618,95 ;13661          MOV LS[T12] TO WR[1]            ;EXPECTED DATA
U 11E9, 0869,3C ;13662          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 11EA, 091C,E4 ;13663          JMP [LOOP.T18.1.9]              ;ERROR LOOP IF ENABLED
                ;13664
U 11EB, FF88,15 ;13665          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 11EC, 3E8B,95 ;13666          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 11ED, 36AC,15 ;13667          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
```

```
U 11EE, 361A,95 ;13668          MOV LS[T13] TO WR[1]            ;EXPECTED DATA
U 11EF, 0869,3C ;13669          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 11F0, 091C,E4 ;13670          JMP [LOOP.T18.1.9]              ;ERROR LOOP IF ENABLED
U 11F1, 5B00,14 ;13671          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;13672
                ;13673  LOOP.T18.A.E:
U 11F2, 6588,15 ;13674          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 11F3, FF88,15 ;13675          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 11F4, 087D,CC ;13676          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 11F5, 3614,15 ;13677          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 11F6, B716,95 ;13678          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 11F7, 90F6,15 ;13679          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'SUB EWR[5] FROM EWR[0]'
U 11F8, 10F6,95 ;13680          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 11F9, 087B,5C ;13681          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;13682                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 11FA, B6A8,15 ;13683          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 11FB, 36A4,95 ;13684          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 11FC, 0869,3C ;13685          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 11FD, 091F,24 ;13686          JMP [LOOP.T18.A.E]              ;ERROR LOOP IF ENABLED
                ;13687
U 11FE, FF88,15 ;13688          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 11FF, 8877,4C ;13689          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;13690                                          ; LOWER 8 BITS
U 1200, 087B,5C ;13691          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1201, B6A8,15 ;13692          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1202, B6A6,95 ;13693          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1203, 0869,3C ;13694          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1204, 091F,24 ;13695          JMP [LOOP.T18.A.E]              ;ERROR LOOP IF ENABLED
U 1205, 5B00,14 ;13696          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;13697  END.T18:
```

```
;13698  .PAGE
;13699  .TOC    "TEST 19 R MINUS S FUNCTION TEST(M8389 FPA MODULE)"
;13700  ;++
;13701  ;
;13702  ;
;13703  ; Test Description:
;13704  ;
;13705  ;       The purpose of this test is to check the R-S function of the fraction
;13706  ;       and exponent data path. The instruction that will be used is ADD FWR[]
;13707  ;       TO -FWR[] + FCOUT and ADD EWR[] TO -EWR[].  The source is A and B,
;13708  ;       the function is R-S and the destination is F->B.  CIN to the least
;13709  ;       significant bit of the exponent data path will always be set (it is
;13710  ;       forced by hardware) when a subtract is performed.  CIN to the fraction
;13711  ;       data path is controlled by a subtract of FWR[1] from FWR[3] just prior
;13712  ;       to executing the ADD FWR[2] TO -FWR[0] + FCOUT instruction.  This
;13713  ;       test will check the case where CIN is either set or clear for the
;13714  ;       fraction data path and where CIN is set for the exponent data paths.
;13715  ;       The exponent data path will be checked seperately due to the need to
;13716  ;       shift data into the upper 8 bits instead of loading them directly.  The
;13717  ;       data patterns that will be used are 0's and 0's, all 1's and 0's, 0's
;13718  ;       and all 1's, all 1's and all 1's, and shifting 1's and shifting 1's.
;13719  ;       These data patterns will be sufficeint to test all GEN and PROP logic.
;13720  ;
;13721  ; Assumptions:
;13722  ;
;13723  ;       It is assumed that all previous tests have run successfully.
;13724  ;
;13725  ; Test Steps:
;13726  ;
;13727  ;       1) Load FWR[FT3] with all 1's.  This will cause COUT to be asserted
;13728  ;          prior to the subtract of FWR[FT0] from FWR[FT2] + FCOUT.
;13729  ;       2) Load FWR[0] with all 0's in all bits and MOV to FWR[2].  Load FWR[0]
;13730  ;          with all 0's again.
;13731  ;       3) Perform SUB FWR[1] FROM FWR[3] TO FQ.  Since FWR[3] is all 1's, FCOUT
;13732  ;          will be set.  Then execute ADD FWR[2] TO -FWR[0] + FCOUT.
;13733  ;       4) Store result from FWR[0] and check for 0's (0 - 0 with carry in set).
;13734  ;       5) Load FWR[0] with ones.  Repeat step 3 with this data.
;13735  ;       6) Store result from FWR[0] and check for a 1 in the LSB of the exten-
;13736  ;          sion (0 - 1's with carry in set).
;13737  ;       7) Load FWR[0] with 1's and move to FWR[2].  Repeat step 3.
;13738  ;       8) Store result from FWR[0] and check for 1's (1's - 0 with carry in
;13739  ;          set).
;13740  ;       9) Load FWR[0] with ones.  Repeat step 3.
;13741  ;       10)Store result from FWR[0] and check for  0 (1's - 1's with carry in
;13742  ;          set).
;13743  ;       11)Load FWR[0] with a shifting 1 pattern and mov to FWR[2].  Reload
;13744  ;          FWR[0] with the same shifting data pattern, and repeat step 3.
;13745  ;       12)Store result from FWR[0] and check result for 0.
;13746  ;       13)Repeat steps 2 through 10 except clear FWR[3] before executing the
;13747  ;          tests.  This will prevent FCOUT from asserting prior to the subtract.
;13748  ;          Check result for correct data.
;13749  ;       14)Repeat steps 2 through 10, this time checking the exponent data path
;13750  ;          by shifting data into the upper 8 bits.  FCOUT is always set by the
;13751  ;          hardware.  EWR[2] and EWR[0] are used.
;13752  ;
```

```
;13753  ; Error:
;13754  ;
;13755  ;       Note:  For the following errors, the data printed under OTHER in the
;13756  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;13757  ;       FLT) which failed.
;13758  ;
;13759  ;       Error1 - SUB failed with data of 0's in FWR[2] and 0's in FWR[0],
;13760  ;                carry in set.
;13761  ;       Error2 - SUB failed with data of 0's in FWR[2] and 1's in FWR[0],
;13762  ;                carry in set.
;13763  ;       Error3 - SUB failed with data of 1's in FWR[2] and 0's in FWR[0],
;13764  ;                carry in set.
;13765  ;       Error4 - SUB failed with data of 1's in FWR[2] and 1's in FWR[0],
;13766  ;                carry in set.
;13767  ;       Error5 - SUB failed with data of shifting 1's in FWR[2] and shifting
;13768  ;                1's in FWR[0], carry in set.
;13769  ;       Error6 - SUB failed with data of 0's in FWR[2] and 0's in FWR[0], with
;13770  ;                no carry in.
;13771  ;       Error7 - SUB failed with data of 0's in FWR[2] and 1's in FWR[0], with
;13772  ;                no carry in.
;13773  ;       Error8 - SUB failed with data of 1's in FWR[2] and 0's in FWR[0], with
;13774  ;                no carry in.
;13775  ;       Error9 - SUB failed with data of 1's in FWR[2] and 1's in FWR[0], with
;13776  ;                no carry in.
;13777  ;
;13778  ;       Note:  For the following errors, the data printed under OTHER in the
;13779  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;13780  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;13781  ;
;13782  ;       ErrorA - SUB failed with data of 0's in EWR[2] and 0's in EWR[0],
;13783  ;                carry in set.
;13784  ;       ErrorB - SUB failed with data of 0's in EWR[2] and 1's in EWR[0],
;13785  ;                carry in set.
;13786  ;       ErrorC - SUB failed with data of 1's in EWR[2] and 0's in EWR[0],
;13787  ;                carry in set.
;13788  ;       ErrorD - SUB failed with data of 1's in EWR[2] and 1's in EWR[0],
;13789  ;                carry in set.
;13790  ;       ErrorE - SUB failed with data of shifting 1's in EWR[2] and shifting
;13791  ;                1's in EWR[0], carry in set.
;13792  ;
;13793  ;
;13794  ; Debug:
;13795  ;
;13796  ;       Error1: This error indicates a basic problem with either the control
;13797  ;               logic or one or more of the 2901's.  The fraction control lines
;13798  ;               should be 011010001 during the ADD FWR[2] TO -FWR[4] + FCOUT
;13799  ;               insruction.  Of the control lines only the FUNC (I5 through I3)
;13800  ;               has not been previously tested.  If the control lines are in
;13801  ;               error, trace them back to their source at the CS PROMS.
;13802  ;                 If a random bit or two is setting, the problem is probably
;13803  ;               in a 2901.  All CIN's to the 2901's should be high.
;13804  ;       Error2: Same as Error 1 except the most likely suspect is a 2901.  All
;13805  ;               CIN's to the 2901's should be low except CIN EXT00 H.
;13806  ;       Error3: Same as Error 1 except the most likely suspect is a 2901.  All
;13807  ;               CIN's should be asserted high on the inputs to all the 2901's.
```

```
                ;13808  ;       Error4: Same as error3.
                ;13809  ;       Error5-9: The most likely suspect is the 2901 with the failing bit.
                ;13810  ;       Errors A-E: Same as Error 1-5 except the exponent path is being test-
                ;13811  ;               ed.  The EXP DECODE PAL is responsible for controling I0 through
                ;13812  ;               I6 to the 2901's.
                ;13813  ;
                ;13814  ;--
                ;13815
                ;13816  ;
                ;13817  ;       TEMPORARY LS LOCATIONS USED:
                ;13818  ;
                ;13819  ; T9    MAIN MEMORY POINTER
                ;13820  ; T10   ADDRESS OF 'SUB FWR[FT1] FROM FWR[FT3] TO FQ' (FIRST WORD)
                ;13821  ;               'ADD FWR[FT2] TO -FWR[FTC] + FCOUT' (SECOND WORD)
                ;13822  ; T12   EXPECTED DATA FOR FIRST AND SECOND FLOAT
                ;13823  ; T13   EXPECTED DATA FOR THIRD FLOAT
                ;13824  ; T14   ADDRESS OF 'MOV FWR[FT0] TO FWR[FT2]'
                ;13825  ;
                ;13826  T.19:
                ;13827
U 1206, B65E,15 ;13828          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;13829                                          ; STATUS WORD
U 1207, 3E80,15 ;13830          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;13831                                          ; BIT 15 INDICATES START OF TEST TO
                ;13832                                          ; CONSOLE
U 1208, 10E0,15 ;13833          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;13834  WAIT.T19.0:
U 1209, 8920,94 ;13835          JMP [WAIT.T19.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;13836                                          ; RESPOND
U 120A, 087E,6C ;13837          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;13838                                          ; AND SIZE BITS
U 120B, B670,15 ;13839          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;13840                                          ; OTHER DATA
U 120C, 3E80,15 ;13841          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 120D, 37C8,15 ;13842          MOV LS[T19.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 120E, BE12,15 ;13843          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 120F, B69E,15 ;13844          MOV LS[ONES] TO WR[0]           ;LOAD ONES IN WR
U 1210, 3EA2,15 ;13845          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA OF ONES
U 1211, 3EA4,15 ;13846          MOV WR[0] TO LS[DATA.2]         ;DITTO FOR SECOND FLOAT
U 1212, BEA6,15 ;13847          MOV WR[0] TO LS[DATA.3]         ;DITTO FOR THIRD FLOAT
U 1213, 0878,FC ;13848          JSR [LOAD.TRIPLE]               ;PUT ALL ONES IN FWR[0] INCLUDING HIDDEN
                ;13849                                          ; BIT
U 1214, 8870,3C ;13850          JSR [L0]                        ;GET MEMORY DATA INTO WR 0
U 1215, 90F6,15 ;13851          MISC2 [TRAP.ACC] WR[0]          ;EXECUTE 'MOV FWR[0] TO FWR[3].  THIS
                ;13852                                          ; WILL CREATE A CIN IN SUBSEQUENT
                ;13853                                          ; SUBTRACTS
U 1216, 8870,3C ;13854          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 1217, 3E1C,15 ;13855          MOV WR[0] TO LS[T14]             ;STORE 'MOV FWR[0] TO FWR[2]' ADDRESS
                ;13856                                          ; IN LS
U 1218, 8870,3C ;13857          JSR [L0]                        ;LOAD WR 0 WITH MEMORY DATA
U 1219, BE14,15 ;13858          MOV WR[0] TO LS[T10]             ;PUT FPA ADDRESS IN LS FOR 'SUB' INSTRUCTION
U 121A, 3725,15 ;13859          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 121B, A0C5,15 ;13860          COM WR[2]                       ;BITS 8-15 = 1
U 121C, C74F,15 ;13861          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;13862                                          ; BIT AND EXPONENT)
```

```
U 121D, B725,95 ;13863          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;13864
U 121E, 6518,15 ;13865          CLR LS[T12]                     ;EXPECTED DATA = 0'S IN FIRST AND
                ;13866                                          ; SECOND FLOAT
U 121F, E51A,15 ;13867          CLR LS[T13]                     ;EXPECTED DATA = 0'S IN THIRD FLOAT
U 1220, 087C,1C ;13868          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 1221, 0878,AC ;13869          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
U 1222, 092A,3C ;13870          JSR [LOOP.T19.1.9]              ;DO SUB OF 0 FROM 0 WITH CARRY IN SET
                ;13871
U 1223, FF82,15 ;13872          INC LS[ERROR.NUMBER]            ;ERROR 2
U 1224, 3650,15 ;13873          MOV LS[BIT8] TO WR[0]           ;SET BIT 8 IN WR
U 1225, 3E1A,15 ;13874          MOV WR[0] TO LS[T13]            ;EXPECTED DATA = LSB SET IN THIRD FLOAT
                ;13875                                          ; (EXTENSION)
U 1226, FDA2,15 ;13876          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1227, FDA4,15 ;13877          COM LS[DATA.2]                  ;SECOND FLOAT DATA OF ONES
U 1228, 7DA6,15 ;13878          COM LS[DATA.3]                  ;THIRD FLOAT DATA OF ONES
U 1229, 092A,3C ;13879          JSR [LOOP.T19.1.9]              ;DO SUB OF 1'S FROM 0 WITH CARRY IN
                ;13880                                          ; SET
                ;13881
U 122A, FF82,15 ;13882          INC LS[ERROR.NUMBER]            ;ERROR 3
U 122B, 0878,FC ;13883          JSR [LOAD.TRIPLE]               ;LOAD FWR 0 WITH ALL ONES
U 122C, 0878,AC ;13884          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
U 122D, 7D18,15 ;13885          COM LS[T12]                     ;EXPECTED DATA OF 1'S IN FIRST AND
                ;13886                                          ; SECOND FLOAT
U 122E, E51A,15 ;13887          CLR LS[T13]                     ;PUT 0'S IN LS
U 122F, FD1A,15 ;13888          COM LS[T13]                     ;EXPECTED DATA OF 1'S IN THIRD FLOAT
U 1230, E5A2,15 ;13889          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1231, E5A4,15 ;13890          CLR LS[DATA.2]                  ;DATA OF 0'S IN SECOND FLOAT
U 1232, 65A6,15 ;13891          CLR LS[DATA.3]                  ;DATA OF 0'S IN THIRD FLOAT
U 1233, 092A,3C ;13892          JSR [LOOP.T19.1.9]              ;DO SUB OF 0 FROM ALL 1'S WITH CARRY IN
                ;13893                                          ; SET
                ;13894
U 1234, FF82,15 ;13895          INC LS[ERROR.NUMBER]            ;ERROR 4
U 1235, 6518,15 ;13896          CLR LS[T12]                     ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13897                                          ; SECOND FLOAT
U 1236, E51A,15 ;13898          CLR LS[T13]                     ;EXPECTED DATA OF 0'S IN THIRD FLOAT
U 1237, FDA2,15 ;13899          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 1238, FDA4,15 ;13900          COM LS[DATA.2]                  ;DATA OF 1'S IN SECOND FLOAT
U 1239, 7DA6,15 ;13901          COM LS[DATA.3]                  ;DATA OF 1'S IN THIRD FLOAT
U 123A, 092A,3C ;13902          JSR [LOOP.T19.1.9]              ;DO SUB OF 1'S FROM 1'S WITH CARRY IN
                ;13903                                          ; SET
                ;13904
U 123B, FF82,15 ;13905          INC LS[ERROR.NUMBER]            ;ERROR 5
U 123C, E5F8,15 ;13906          CLR LS[OS]                      ;CLEAR INDEX
U 123D, 6518,15 ;13907          CLR LS[T12]                     ;EXPECTED DATA OF 0'S IN FIRST AND
                ;13908                                          ; SECOND FLOAT
U 123E, E51A,15 ;13909          CLR LS[T13]                     ;EXPECTED DATA OF 0'S IN THIRD FLOAT
                ;13910  REPEAT.T19.5:
U 123F, 36F6,15 ;13911          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 1240, 3EA2,15 ;13912          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 1241, 3EA4,15 ;13913          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 1242, BEA6,15 ;13914          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 1243, 0878,FC ;13915          JSR [LOAD.TRIPLE]               ;LOAD SHIFT PATTERN INTO FWR[0]
U 1244, 0878,AC ;13916          JSR [MOV.DATA]                  ;MOV FWR0 TO FWR2
U 1245, 092A,3C ;13917          JSR [LOOP.T19.1.9]              ;GO SUB SHIFTING 1'S FROM SHIFTING 1'S
```

```
                  ;13918                                        ; WITH CARRY IN SET
U 1246, 7FF8,15   ;13919          INC LS[OS]                     ;INCREMENT INDEX
U 1247, B6F8,15   ;13920          MOV LS[OS] TO WR[0]            ;GET INDEX
                  ;13921          CMP WR[0] WITH LS[#20],        ;DONE ALL PATTERNS?
U 1248, CF4A,35   ;13922           DT(LONG)&SET.ALU.CC           ; SET CONDITION CODES
U 1249, 8923,F1   ;13923          JMP.IF[NEQ] TO [REPEAT.T19.5]  ;REPEAT IF NOT
                  ;13924
U 124A, FF82,15   ;13925          INC LS[ERROR.NUMBER]           ;ERROR 6
U 124B, B69E,15   ;13926          MOV LS[ONES] TO WR[0]          ;PUT ALL 1'S IN WR 0
U 124C, BE18,15   ;13927          MOV WR[0] TO LS[T12]           ;EXPECTED DATA = 1'S IN FIRST AND
                  ;13928                                        ; SECOND FLOAT
U 124D, 3E1A,15   ;13929          MOV WR[0] TO LS[T13]           ;EXPECTED DATA = 1'S IN THIRD FLOAT
U 124E, 087C,1C   ;13930          JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 124F, 8870,3C   ;13931          JSR [L0]                       ;LOAD WR 0 WITH ADDRESS OF CLR FWR[3]
U 1250, B716,95   ;13932          MOV LS[#300] TO WR[1]          ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 1251, 90F6,15   ;13933          MISC2 [TRAP.ACC] WR[0]         ;FORCE ADDRESS OF CLR FWR[3].  THIS
                  ;13934                                        ; WILL PREVENT A CARRY IN
U 1252, 10F6,95   ;13935          MISC2 [TRAP.ACC] WR[1]         ;FORCE LOOP SELF
U 1253, 0878,AC   ;13936          JSR [MOV.DATA]                 ;MOV FWR0 TO FWR2
U 1254, 092A,3C   ;13937          JSR [LOOP.T19.1.9]             ;DO SUB OF 0 FROM 0 WITH NO CARRY IN
                  ;13938
U 1255, FF82,15   ;13939          INC LS[ERROR.NUMBER]           ;ERROR 7
U 1256, 6518,15   ;13940          CLR LS[T12]                    ;EXPECTED DATA OF 0'S IN FIRST AND
                  ;13941                                        ; SECOND FLOAT
U 1257, E51A,15   ;13942          CLR LS[T13]                    ;EXPECTED DATA OF 0'S IN THIRD FLOAT
U 1258, FDA2,15   ;13943          COM LS[DATA.1]                 ;FIRST FLOAT DATA OF ONES
U 1259, FDA4,15   ;13944          COM LS[DATA.2]                 ;SECOND FLOAT DATA OF ONES
U 125A, 7DA6,15   ;13945          COM LS[DATA.3]                 ;THIRD FLOAT DATA OF ONES
U 125B, 092A,3C   ;13946          JSR [LOOP.T19.1.9]             ;DO SUB OF ALL 1'S FROM 0 WITH NO CARRY
                  ;13947                                        ; IN
                  ;13948
U 125C, FF82,15   ;13949          INC LS[ERROR.NUMBER]           ;ERROR 8
U 125D, 0878,FC   ;13950          JSR [LOAD.TRIPLE]              ;LOAD FWR 0 WITH ALL ONES
U 125E, 0878,AC   ;13951          JSR [MOV.DATA]                 ;MOV FWR0 TO FWR2
U 125F, 7D18,15   ;13952          COM LS[T12]                    ;EXPECTED DATA OF 1'S IN FIRST FLOAT
U 1260, 5F50,15   ;13953          MCOM LS[BIT8] TO WR[0]         ;GET ALL 1'S EXCEPT FOR BIT 8 IN WR
U 1261, 3E1A,15   ;13954          MOV WR[0] TO LS[T13]           ;EXPECTED DATA OF 1'S EXCEPT FOR LSB
                  ;13955                                        ; OF THIRD FLOAT (EXTENSION)
U 1262, E5A2,15   ;13956          CLR LS[DATA.1]                 ;DATA OF 0'S IN FIRST FLOAT
U 1263, E5A4,15   ;13957          CLR LS[DATA.2]                 ;DATA OF 0'S IN SECOND FLOAT
U 1264, 65A6,15   ;13958          CLR LS[DATA.3]                 ;DATA OF 0'S IN THIRD FLOAT
U 1265, 092A,3C   ;13959          JSR [LOOP.T19.1.9]             ;DO SUB OF 0 FROM ALL 1'S WITH NO CARRY
                  ;13960                                        ; IN
                  ;13961
U 1266, FF82,15   ;13962          INC LS[ERROR.NUMBER]           ;ERROR 9
U 1267, E51A,15   ;13963          CLR LS[T13],                   ;PUT 0'S IN LS
U 1268, FD1A,15   ;13964          COM LS[T13]                    ;EXPECTED DATA = ALL 1'S IN THIRD FLOAT
U 1269, FDA2,15   ;13965          COM LS[DATA.1]                 ;DATA OF 1'S IN FIRST FLOAT
U 126A, FDA4,15   ;13966          COM LS[DATA.2]                 ;DATA OF 1'S IN SECOND FLOAT
U 126B, 7DA6,15   ;13967          COM LS[DATA.3]                 ;DATA OF 1'S IN THIRD FLOAT
U 126C, 092A,3C   ;13968          JSR [LOOP.T19.1.9]             ;DO SUB OF 1'S FROM 1'S WITH NO CARRY IN
                  ;13969
                  ;13970  ;
                  ;13971  ;       TEMPORARY LS LOCATIONS USED:
                  ;13972  ;
```

```
                ;13973  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;13974  ; T9    MAIN MEMORY POINTER
                ;13975  ; T10   ADDRESS OF 'ADD EWR[ET2] TO -EWR[ET0]'
                ;13976  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;13977  ; T14   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET2]'
                ;13978  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;13979  ;
                ;13980
U 126D, FF82,15 ;13981          INC LS[ERROR.NUMBER]            ;ERROR A
U 126E, B724,15 ;13982          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 126F, C75E,15 ;13983          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 1270, C54E,15 ;13984          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 1271, 3E8A,15 ;13985          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 1272, 37EA,15 ;13986          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;13987                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 1273, 3E10,15 ;13988          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 1274, 8870,3C ;13989          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'SUB' ROUTINE
U 1275, BE14,15 ;13990          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 1276, 37EC,15 ;13991          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;13992                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 1277, 3E16,15 ;13993          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 1278, B7EE,15 ;13994          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;13995                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 1279, BE1E,15 ;13996          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
                ;13997
U 127A, E5A2,15 ;13998          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 127B, E5A4,15 ;13999          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 127C, 65A6,15 ;14000          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 127D, 087D,CC ;14001          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 127E, 0878,AC ;14002          JSR [MOV.DATA]                  ;MOV EWR0 TO EWR[ET2]
U 127F, 092B,EC ;14003          JSR [LOOP.T19.A.E]              ;DO SUB OF 0 FROM 0 WITH CARRY IN SET
                ;14004
U 1280, FF82,15 ;14005          INC LS[ERROR.NUMBER]            ;ERROR B
U 1281, FDA2,15 ;14006          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1282, 364E,15 ;14007          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 IN WR
U 1283, 3EA4,15 ;14008          MOV WR[0] TO LS[DATA.2]         ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 1284, 092B,EC ;14009          JSR [LOOP.T19.A.E]              ;DO SUB OF ALL 1'S FROM 0 WITH CARRY IN
                ;14010                                          ; SET
                ;14011
U 1285, FF82,15 ;14012          INC LS[ERROR.NUMBER]            ;ERROR C
U 1286, 087D,CC ;14013          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1287, 0878,AC ;14014          JSR [MOV.DATA]                  ;MOV EWR0 TO EWR[ET2] (ALL 1'S)
U 1288, E5A2,15 ;14015          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1289, E5A4,15 ;14016          CLR LS[DATA.2]                  ;DATA OF 0 IN LS
U 128A, FDA4,15 ;14017          COM LS[DATA.2]                  ;EXPECTED DATA OF 1'S IN LOWER 8 BITS
U 128B, 7DA6,15 ;14018          COM LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 128C, 092B,EC ;14019          JSR [LOOP.T19.A.E]              ;DO SUB OF 0 FROM ALL 1'S WITH CARRY IN
                ;14020                                          ; SET
                ;14021
U 128D, FF82,15 ;14022          INC LS[ERROR.NUMBER]            ;ERROR D
U 128E, FDA2,15 ;14023          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 128F, E5A4,15 ;14024          CLR LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 1290, 65A6,15 ;14025          CLR LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 1291, 092B,EC ;14026          JSR [LOOP.T19.A.E]              ;DO SUB OF 1'S FROM 1'S WITH CARRY IN
                ;14027                                          ; SET
```

```
                ;14028
U 1292, FF82,15 ;14029          INC LS[ERROR.NUMBER]                ;ERROR E
U 1293, B64A,15 ;14030          MOV LS[#20] TO WR[0]                ;PUT AN 20 IN WR
U 1294, 2100,15 ;14031          DEC WR[0]                           ;1F(H) NOW IN WR
U 1295, 3EF8,15 ;14032          MOV WR[0] TO LS[OS]                 ;START INDEX AT 1F (POINTS TO BIT 31
                ;14033                                              ; SET).  BIT 31 WILL END UP IN LSB OF
                ;14034                                              ; EXPONENT.
U 1296, E5A4,15 ;14035          CLR LS[DATA.2]                      ;EXPECTED DATA WILL ALWAYS BE 0 IN LOWER
                ;14036                                              ; 8 BITS OF EXP
U 1297, 65A6,15 ;14037          CLR LS[DATA.3]                      ;EXPECTED DATA WILL ALWAYS BE 0 IN UPPER
                ;14038                                              ; 8 BITS OF EXP
                ;14039  REPEAT.T19.E:
U 1298, B6F6,95 ;14040          MOV LS[SHIFT.OS(4-0)] TO WR[1]      ;LOAD DATA PATTERN IN WR
U 1299, BEA2,95 ;14041          MOV WR[1] TO LS[DATA.1]             ;FIRST FLOAT DATA
U 129A, 087D,CC ;14042          JSR [LOAD.SHIFT.EWR0]               ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 129B, 0878,AC ;14043          JSR [MOV.DATA]                      ;MOV PATTERN TO EWR[ET2]
U 129C, 092B,EC ;14044          JSR [LOOP.T19.A.E]                  ;GO SUB SHIFTING 1'S FROM SHIFTING 1'S
U 129D, 7FF8,15 ;14045          INC LS[OS]                          ;INCREMENT INDEX
U 129E, B64A,15 ;14046          MOV LS[#20] TO WR[0]                ;GET DATA OF 20
U 129F, 4748,15 ;14047          BIS LS[#10] TO WR[0]                ;NOW HAVE 30 IN WR
                ;14048          CMP WR[0] WITH LS[OS],              ;DONE ALL PATTERNS?
U 12A0, CFF8,35 ;14049           DT(LONG)&SET.ALU.CC                ; SET CONDITION CODES
U 12A1, 0929,81 ;14050          JMP.IF[NEQ] TO [REPEAT.T19.E]       ;REPEAT IF NOT
                ;14051
U 12A2, 892D,24 ;14052          JMP [END.T19]                       ;DONE WITH SUB TEST
                ;14053
                ;14054  LOOP.T19.1.9:
U 12A3, 6588,15 ;14055          CLR LS[ADDRESS.DATA]                ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 12A4, FF88,15 ;14056          INC LS[ADDRESS.DATA]                ;1 INDICATES FIRST FLOAT FAILED
U 12A5, 0878,FC ;14057          JSR [LOAD.TRIPLE]                   ;LOAD SECOND OPERAND  OF SUB IN FWR[FT0]
U 12A6, 3614,15 ;14058          MOV LS[T10] TO WR[0]                ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 12A7, B716,95 ;14059          MOV LS[#300] TO WR[1]               ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 12A8, 90F6,15 ;14060          MISC2 [TRAP.ACC] WR[0]              ;FORCE 'SUB FWR[1] FROM FWR[3] TO FQ'
                ;14061                                              ; THIS WILL SET UP CARRY TO CORRECT VALUE
U 12A9, DB00,15 ;14062          NOP                                 ;FPA WILL DO 'ADD FWR[2] TO -FWR[0] +
                ;14063                                              ; FCOUT DURING THIS NOP
U 12AA, 10F6,95 ;14064          MISC2 [TRAP.ACC] WR[1]              ;FORCE LOOP SELF
U 12AB, 087B,5C ;14065          JSR [STORE.0.TRIPLE]                ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14066                                              ; AND THIRD.FLT
U 12AC, BE8B,15 ;14067          MOV WR[2] TO LS[ERROR.MASK]         ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14068                                              ; AND EXPONENT (BITS 7-14)
U 12AD, B6A8,15 ;14069          MOV LS[FIRST.FLT] TO WR[0]          ;PUT FIRST FLT RESULT IN WR 0
U 12AE, B618,95 ;14070          MOV LS[T12] TO WR[1]                ;EXPECTED DATA
U 12AF, 0869,3C ;14071          JSR [CHECK.RESULT]                  ;CHECK FOR ERRORS
U 12B0, 892A,34 ;14072          JMP [LOOP.T19.1.9]                  ;ERROR LOOP IF ENABLED
                ;14073
U 12B1, FF88,15 ;14074          INC LS[ADDRESS.DATA]                ;INC PRINTOUT UNDER OTHER DATA TO 2
U 12B2, E58A,15 ;14075          CLR LS[ERROR.MASK]                  ;CHECK ALL BITS
U 12B3, 36AA,15 ;14076          MOV LS[SEC.FLT] TO WR[0]            ;PUT SECOND FLT RESULT IN WR 0
U 12B4, B618,95 ;14077          MOV LS[T12] TO WR[1]                ;EXPECTED DATA
U 12B5, 0869,3C ;14078          JSR [CHECK.RESULT]                  ;CHECK FOR ERRORS
U 12B6, 892A,34 ;14079          JMP [LOOP.T19.1.9]                  ;ERROR LOOP IF ENABLED
                ;14080
U 12B7, FF88,15 ;14081          INC LS[ADDRESS.DATA]                ;INC PRINTOUT UNDER OTHER DATA TO 3
U 12B8, 3E8B,95 ;14082          MOV WR[3] TO LS[ERROR.MASK]         ;CHECK BITS 8-15 ONLY
```

```
U 12B9, 36AC,15 ;14083          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 12BA, 361A,95 ;14084          MOV LS[T13] TO WR[1]            ;EXPECTED DATA
U 12BB, 0869,3C ;14085          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 12BC, 892A,34 ;14086          JMP [LOOP.T19.1.9]              ;ERROR LOOP IF ENABLED
U 12BD, 5B00,14 ;14087          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;14088
                ;14089  LOOP.T19.A.E:
U 12BE, 6588,15 ;14090          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 12BF, FF88,15 ;14091          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 12C0, 087D,CC ;14092          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 12C1, 3614,15 ;14093          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 12C2, B716,95 ;14094          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 12C3, 90F6,15 ;14095          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD EWR[ET2] TO -EWR[ET0]'
U 12C4, 10F6,95 ;14096          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 12C5, 087B,5C ;14097          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;14098                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 12C6, B6A8,15 ;14099          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 12C7, 36A4,95 ;14100          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 12C8, 0869,3C ;14101          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 12C9, 892B,E4 ;14102          JMP [LOOP.T19.A.E]              ;ERROR LOOP IF ENABLED
                ;14103
U 12CA, FF88,15 ;14104          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 12CB, 8877,4C ;14105          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;14106                                          ; LOWER 8 BITS
U 12CC, 087B,5C ;14107          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 12CD, B6A8,15 ;14108          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 12CE, B6A6,95 ;14109          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 12CF, 0869,3C ;14110          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 12D0, 892B,E4 ;14111          JMP [LOOP.T19.A.E]              ;ERROR LOOP IF ENABLED
U 12D1, 5B00,14 ;14112          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;14113  END.T19:
```

```
;14114  .PAGE
;14115  .TOC    "TEST 1A 2F -> B DESTINATION TEST"
;14116  ;++
;14117  ;
;14118  ;
;14119  ; Test Description:
;14120  ;
;14121  ;       The purpose of this test is to check the 2F -> B destination control
;14122  ;       lines to the 2901's in the FPA. This will be done by putting data on
;14123  ;       the direct input lines then issuing an OR of the direct inputs and
;14124  ;       zero. The destination is 2F -> B. The result will be read from the
;14125  ;       FWR designated by the B address lines back to the CPU and checked for
;14126  ;       error. The data that will be used to test the destination is a
;14127  ;       shifting 1 pattern.  What is shifted into the LSB of the data path
;14128  ;       (the LSB of the EXTENSION data path) will be tested in a later test.
;14129  ;         This test will also check that the Q reg (FQ) is not altered by
;14130  ;       loading it with an alternating 1's and 0's pattern.
;14131  ;
;14132  ; Assumptions:
;14133  ;
;14134  ;       It is assumed that all previous tests have run successfully.
;14135  ;
;14136  ; Test Steps:
;14137  ;
;14138  ;       1) Load FWR[FTO] with a shifting one pattern in each float section.
;14139  ;       2) Execute a SHF LEFT FWR[FTO] instruction and check for correct result.
;14140  ;       3) Repeat step 2 until all 32 bits have been tested.
;14141  ;       4) Load FWR[FTO] with an alternating ones and zeros pattern in each
;14142  ;          float section.
;14143  ;       5) Mov data from FWR[0] to FQ.
;14144  ;       6) Execute a SHF LEFT FWR[FTO] instruction and check that the Q reg was
;14145  ;          not altered.
;14146  ;
;14147  ; Error:
;14148  ;
;14149  ;       Note:  For the following errors, the data printed under OTHER in the
;14150  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;14151  ;       FLT) which failed.
;14152  ;
;14153  ;       Error1 - 2F -> B (shift left) failed on FWR[FTO].
;14154  ;       Error2 - 2F -> B (shift left) altered Q register.
;14155  ;
;14156  ; Debug:
;14157  ;
;14158  ;       Error1: The cause of this error is likely to be either the 2901 con-
;14159  ;               trol lines or the 2901 itself. The fraction data path control
;14160  ;               lines should be 111011100.  Of these, only the DST control (I8
;14161  ;               through I6) have not been previously tested.  If these are in-
;14162  ;               correct, trace them back to the CONTROL STORE.
;14163  ;                 If the error occurs over a 4 bit boundary, check that the
;14164  ;               shift out (pin 8) of the preceeding 2901 is going high during
;14165  ;               the SHF LEFT FWR[FTO] instruction.  If not, the preceeding 2901
;14166  ;               is probably bad.
;14167  ;                 If the shift out is OK, or any other bits are failing, sus-
;14168  ;               pect the 2901 outputting the failing bit.
```

```
                ;14169  ;
                ;14170  ;       Error2: This error can be caused by either the control lines or one of
                ;14171  ;               the 2901's.  Check the control lines as in error 1 above.  If
                ;14172  ;               these are correct, the 2901 is probably at fault.
                ;14173  ;
                ;14174  ;--
                ;14175
                ;14176  ;
                ;14177  ;       TEMPORARY LS LOCATIONS USED:
                ;14178  ;
                ;14179  ; T9    MAIN MEMORY POINTER
                ;14180  ; T10   ADDRESS OF 'SHF LEFT FWR[FT0] IN[ONE]'
                ;14181  ; T11   EXPECTED DATA FOR EACH FLOAT SECTION
                ;14182  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FQ'
                ;14183  ; T13   ADDRESS OF 'MOV FQ TO FWR[FT2]'
                ;14184  ; T14   ADDRESS OF MOV ABOVE IS PLACED HERE FOR MOV SUBROUTINE
                ;14185  ;
                ;14186  T.1A:
                ;14187
U 12D2, B65E,15 ;14188          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;14189                                          ; STATUS WORD
U 12D3, 3E80,15 ;14190          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;14191                                          ; BIT 15 INDICATES START OF TEST TO
                ;14192                                          ; CONSOLE
U 12D4, 10E0,15 ;14193          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;14194  WAIT.T1A.0:
U 12D5, 092D,54 ;14195          JMP [WAIT.T1A.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;14196                                          ; RESPOND
U 12D6, 087E,6C ;14197          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;14198                                          ; AND SIZE BITS
U 12D7, B670,15 ;14199          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;14200                                          ; OTHER DATA
U 12D8, 3E80,15 ;14201          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 12D9, B7CA,15 ;14202          MOV LS[T1A.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 12DA, BE12,15 ;14203          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 12DB, 8870,3C ;14204          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 12DC, BE14,15 ;14205          MOV WR[0] TO LS[T10]            ;LOAD LS WITH ADDRESS OF 'SHF LEFT' IN-
                ;14206                                          ; STRUCTION
U 12DD, 8870,3C ;14207          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 12DE, BE18,15 ;14208          MOV WR[0] TO LS[T12]            ;LOAD LS WITH ADDRESS OF 'MOV FWR[0] TO
                ;14209                                          ; FQ' INSTRUCTION
U 12DF, 8870,3C ;14210          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 12E0, 3E1A,15 ;14211          MOV WR[0] TO LS[T13]            ;LOAD LS WITH ADDRESS OF 'MOV FQ TO
                ;14212                                          ; FWR[0]' INSTRUCTION
U 12E1, 3725,15 ;14213          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 12E2, A0C5,15 ;14214          COM WR[2]                       ;BITS 8-15 = 1
U 12E3, C74F,15 ;14215          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;14216                                          ; BIT AND EXPONENT)
U 12E4, B725,95 ;14217          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
U 12E5, 4751,95 ;14218          BIS LS[BIT8] TO WR[3]           ;ALSO MASK OFF LSB (SHIFT INPUT FROM
                ;14219                                          ; OUTSIDE IS CHECKED IN ANOTHER TEST)
                ;14220
U 12E6, E5F8,15 ;14221          CLR LS[OS]                      ;CLEAR INDEX
                ;14222  REPEAT.T1A.1:
U 12E7, 36F6,15 ;14223          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
```

```
U 12E8, 3EA2,15 ;14224          MOV WR[0] TO LS[DATA.1]          ;FIRST FLOAT DATA
U 12E9, 3EA4,15 ;14225          MOV WR[0] TO LS[DATA.2]          ;SECOND FLOAT DATA
U 12EA, BEA6,15 ;14226          MOV WR[0] TO LS[DATA.3]          ;THIRD FLOAT DATA
U 12EB, A3C0,15 ;14227          ROL WR[0]                        ;EXPECTED DATA
U 12EC, 3E16,15 ;14228          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14229  LOOP.T1A.1:
U 12ED, 0878,FC ;14230          JSR [LOAD.TRIPLE]                ;LOAD SHIFT PATTERN INTO FWR[0]
U 12EE, 3614,15 ;14231          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF LEFT FPA INSTRUCTION
U 12EF, B716,95 ;14232          MOV LS[#300] TO WR[1]            ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 12F0, 90F6,15 ;14233          MISC2 [TRAP.ACC] WR[0]           ;FORCE SHF LEFT ON FWR[FTO]
U 12F1, 10F6,95 ;14234          MISC2 [TRAP.ACC] WR[1]           ;FORCE LOOP SELF
U 12F2, 6588,15 ;14235          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 12F3, FF88,15 ;14236          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 12F4, 087B,5C ;14237          JSR [STORE.0.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14238                                           ; AND THIRD.FLT
U 12F5, BE8B,15 ;14239          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14240                                           ; AND EXPONENT (BITS 7-14)
U 12F6, B6A8,15 ;14241          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 12F7, 3616,95 ;14242          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 12F8, 0869,3C ;14243          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 12F9, 892E,D4 ;14244          JMP [LOOP.T1A.1]                 ;ERROR LOOP IF ENABLED
                ;14245
U 12FA, FF88,15 ;14246          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 12FB, E58A,15 ;14247          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 12FC, 36AA,15 ;14248          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
U 12FD, 3616,95 ;14249          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 12FE, 0869,3C ;14250          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 12FF, 892E,D4 ;14251          JMP [LOOP.T1A.1]                 ;ERROR LOOP IF ENABLED
                ;14252
U 1300, FF88,15 ;14253          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 3
U 1301, 3E8B,95 ;14254          MOV WR[3] TO LS[ERROR.MASK]      ;CHECK BITS 8-15 ONLY
U 1302, 36AC,15 ;14255          MOV LS[THIRD.FLT] TO WR[0]       ;PUT THIRD FLT RESULT IN WR 0
U 1303, 3616,95 ;14256          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 1304, 0869,3C ;14257          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 1305, 892E,D4 ;14258          JMP [LOOP.T1A.1]                 ;ERROR LOOP IF ENABLED
                ;14259
U 1306, 7FF8,15 ;14260          INC LS[OS]                       ;INCREMENT INDEX
U 1307, B6F8,15 ;14261          MOV LS[OS] TO WR[0]              ;GET INDEX
                ;14262          CMP WR[0] WITH LS[#20],          ;DONE ALL PATTERNS?
U 1308, CF4A,35 ;14263           DT(LONG)&SET.ALU.CC             ; SET CONDITION CODES
U 1309, 892E,71 ;14264          JMP.IF[NEQ] TO [REPEAT.T1A.1]    ;REPEAT IF NOT
                ;14265
U 130A, FF82,15 ;14266          INC LS[ERROR.NUMBER]             ;ERROR 2
U 130B, B725,95 ;14267          MOV LS[#FFFF00FF] TO WR[3]       ;CHANGE ERROR MASK TO CHECK BYTE 1 ONLY
U 130C, 369A,15 ;14268          MOV LS[#55555555] TO WR[0]       ;DATA IS ALTERNATING 1'S AND 0'S PATTERM
U 130D, 3EA2,15 ;14269          MOV WR[0] TO LS[DATA.1]          ;FIRST FLOAT DATA
U 130E, 3EA4,15 ;14270          MOV WR[0] TO LS[DATA.2]          ;SECOND FLOAT DATA
U 130F, BEA6,15 ;14271          MOV WR[0] TO LS[DATA.3]          ;THIRD FLOAT DATA
U 1310, 3E16,15 ;14272          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14273  LOOP.T1A.2:
U 1311, 0878,FC ;14274          JSR [LOAD.TRIPLE]                ;LOAD ALTERNATING PATTERN INTO FWR[0]
U 1312, 3618,15 ;14275          MOV LS[T12] TO WR[0]             ;GET ADDRESS OF 'MOV FWR[0] TO FQ'
U 1313, 3E1C,15 ;14276          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 1314, 0878,AC ;14277          JSR [MOV.DATA]                   ;LOAD FQ WITH DATA
U 1315, 3614,15 ;14278          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF LEFT FPA INSTRUCTION
```

```
U 1316, B716,95 ;14279          MOV LS[#300] TO WR[1]          ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 1317, 90F6,15 ;14280          MISC2 [TRAP.ACC] WR[0]         ;FORCE SHF LEFT ON FWR[FT0]
U 1318, 10F6,95 ;14281          MISC2 [TRAP.ACC] WR[1]         ;FORCE LOOP SELF
U 1319, B61A,15 ;14282          MOV LS[T13] TO WR[0]           ;ADDRESS OF 'MOV FQ TO FWR[2]'
U 131A, 3E1C,15 ;14283          MOV WR[0] TO LS[T14]           ;SET UP FOR MOV ROUTINE
U 131B, 0878,AC ;14284          JSR [MOV.DATA]                 ;LOAD FWR[2] FROM FQ
U 131C, 6588,15 ;14285          CLR LS[ADDRESS.DATA]           ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 131D, FF88,15 ;14286          INC LS[ADDRESS.DATA]           ;1 INDICATES FIRST FLOAT FAILED
U 131E, 087A,1C ;14287          JSR [STORE.2.TRIPLE]           ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14288                                         ; AND THIRD.FLT
U 131F, BE8B,15 ;14289          MOV WR[2] TO LS[ERROR.MASK]    ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14290                                         ; AND EXPONENT (BITS 7-14)
U 1320, B6A8,15 ;14291          MOV LS[FIRST.FLT] TO WR[0]     ;PUT FIRST FLT RESULT IN WR 0
U 1321, 3616,95 ;14292          MOV LS[T11] TO WR[1]           ;EXPECTED DATA
U 1322, 0869,3C ;14293          JSR [CHECK.RESULT]             ;CHECK FOR ERRORS
U 1323, 0931,14 ;14294          JMP [LOOP.T1A.2]               ;ERROR LOOP IF ENABLED
                ;14295
U 1324, FF88,15 ;14296          INC LS[ADDRESS.DATA]           ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1325, E58A,15 ;14297          CLR LS[ERROR.MASK]             ;CHECK ALL BITS
U 1326, 36AA,15 ;14298          MOV LS[SEC.FLT] TO WR[0]       ;PUT SECOND FLT RESULT IN WR 0
U 1327, 3616,95 ;14299          MOV LS[T11] TO WR[1]           ;EXPECTED DATA
U 1328, 0869,3C ;14300          JSR [CHECK.RESULT]             ;CHECK FOR ERRORS
U 1329, 0931,14 ;14301          JMP [LOOP.T1A.2]               ;ERROR LOOP IF ENABLED
                ;14302
U 132A, FF88,15 ;14303          INC LS[ADDRESS.DATA]           ;INC PRINTOUT UNDER OTHER DATA TO 3
U 132B, 3E8B,95 ;14304          MOV WR[3] TO LS[ERROR.MASK]    ;CHECK BITS 8-15 ONLY
U 132C, 36AC,15 ;14305          MOV LS[THIRD.FLT] TO WR[0]     ;PUT THIRD FLT RESULT IN WR 0
U 132D, 3616,95 ;14306          MOV LS[T11] TO WR[1]           ;EXPECTED DATA
U 132E, 0869,3C ;14307          JSR [CHECK.RESULT]             ;CHECK FOR ERRORS
U 132F, 0931,14 ;14308          JMP [LOOP.T1A.2]               ;ERROR LOOP IF ENABLED
                ;14309  END.T1A:
```

```
;14310  .PAGE
;14311  .TOC    "TEST 1B F/2 -> B DESTINATION TEST"
;14312  ;++
;14313  ;
;14314  ; Test Description:
;14315  ;
;14316  ;       The purpose of this test is to check the F/2 -> B destination control
;14317  ;       lines to the 2901's in the FPA. This will be done by putting data on
;14318  ;       the direct input lines then issuing an OR of the direct inputs and
;14319  ;       zero. The destination is F/2 -> B. The result will be read from the
;14320  ;       FWR designated by the B address lines back to the CPU and checked for
;14321  ;       error. The data that will be used to test the destination is a
;14322  ;       shifting 1 pattern.  What is shifted into the MSB of the data path
;14323  ;       (the MSB of the FRACTION data path) will be tested in a later test.
;14324  ;         This test will also check that the Q reg (FQ) is not altered by
;14325  ;       loading it with an alternating 1's and 0's pattern.
;14326  ;
;14327  ; Assumptions:
;14328  ;
;14329  ;       It is assumed that all previous tests have run successfully.
;14330  ;
;14331  ; Test Steps:
;14332  ;
;14333  ;       1) Load FWR[FTO] with a shifting one pattern in each float section.
;14334  ;       2) Execute a SHF RIGHT FWR[FTO] instruction and check for correct result.
;14335  ;       3) Repeat step 2 until all 32 bits have been tested.
;14336  ;       4) Load FWR[FTO] with an alternating ones and zeros pattern in each
;14337  ;          float section.
;14338  ;       5) Mov data from FWR[0] to FQ.
;14339  ;       6) Execute a SHF RIGHT FWR[FTO] instruction and check that the Q reg was
;14340  ;          not altered.
;14341  ;
;14342  ; Error:
;14343  ;
;14344  ;       Note:  For the following errors, the data printed under OTHER in the
;14345  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;14346  ;       FLT) which failed.
;14347  ;
;14348  ;       Error1 - F/2 -> B (shift RIGHT) failed on FWR[FTO].
;14349  ;       Error2 - F/2 -> B (shift RIGHT) altered Q register.
;14350  ;
;14351  ; Debug:
;14352  ;
;14353  ;       Error1: The cause of this error is likely to be either the 2901 con-
;14354  ;               trol lines or the 2901 itself. The fraction data path control
;14355  ;               lines should be 101011100.  Of these, only the DST control (I8
;14356  ;               through I6) have not been previously tested.  If these are in-
;14357  ;               correct, trace them back to the CONTROL STORE.
;14358  ;                 If the error occurs over a 4 bit boundary, check that the
;14359  ;               shift out (pin 9) of the preceeding 2901 is going high during
;14360  ;               the SHF RIGHT FWR[FTO] instruction.  If not, the preceeding 2901
;14361  ;               is probably bad.
;14362  ;                 If the shift out is OK, or any other bits are failing, sus-
;14363  ;               pect the 2901 outputting the failing bit.
;14364  ;
```

```
                ;14365  ;       Error2: This error can be caused by either the control lines or one of
                ;14366  ;               the 2901's.  Check the control lines as in error 1 above.  If
                ;14367  ;               these are correct, the 2901 is probably at fault.
                ;14368  ;
                ;14369  ;--
                ;14370
                ;14371  ;
                ;14372  ;       TEMPORARY LS LOCATIONS USED:
                ;14373  ;
                ;14374  ; T9    MAIN MEMORY POINTER
                ;14375  ; T10   ADDRESS OF 'SHF RIGHT FWR[FT0] IN[MUL.SHF]'
                ;14376  ; T11   EXPECTED DATA FOR EACH FLOAT SECTION
                ;14377  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FQ'
                ;14378  ; T13   ADDRESS OF 'MOV FQ TO FWR[FT2]'
                ;14379  ; T14   ADDRESS OF MOV ABOVE IS PLACED HERE FOR MOV SUBROUTINE
                ;14380  ;
                ;14381  T.1B:
                ;14382
U 1330, B65E,15 ;14383          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;14384                                          ; STATUS WORD
U 1331, 3E80,15 ;14385          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;14386                                          ; BIT 15 INDICATES START OF TEST TO
                ;14387                                          ; CONSOLE
U 1332, 10E0,15 ;14388          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;14389  WAIT.T1B.0:
U 1333, 0933,34 ;14390          JMP [WAIT.T1B.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;14391                                          ; RESPOND
U 1334, 087E,6C ;14392          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;14393                                          ; AND SIZE BITS
U 1335, B670,15 ;14394          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;14395                                          ; OTHER DATA
U 1336, 3E80,15 ;14396          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 1337, B7CC,15 ;14397          MOV LS[T1B.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 1338, BE12,15 ;14398          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 1339, 8870,3C ;14399          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 133A, BE14,15 ;14400          MOV WR[0] TO LS[T10]            ;LOAD LS WITH ADDRESS OF 'SHF RIGHT' IN-
                ;14401                                          ; STRUCTION
U 133B, 8870,3C ;14402          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 133C, BE18,15 ;14403          MOV WR[0] TO LS[T12]            ;LOAD LS WITH ADDRESS OF 'MOV FWR[0] TO
                ;14404                                          ; FQ' INSTRUCTION
U 133D, 8870,3C ;14405          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 133E, 3E1A,15 ;14406          MOV WR[0] TO LS[T13]            ;LOAD LS WITH ADDRESS OF 'MOV FQ TO
                ;14407                                          ; FWR[0]' INSTRUCTION
U 133F, 3725,15 ;14408          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 1340, A0C5,15 ;14409          COM WR[2]                       ;BITS 8-15 = 1
U 1341, C74F,15 ;14410          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;14411                                          ; BIT AND EXPONENT)
U 1342, 474D,15 ;14412          BIS LS[BIT6] TO WR[2]           ;ALSO MASK MSB (BIT SHIFTED IN FROM
                ;14413                                          ; OUTSIDE IS CHECKED IN A LATER TEST)
U 1343, B725,95 ;14414          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;14415
U 1344, E5F8,15 ;14416          CLR LS[OS]                      ;CLEAR INDEX
                ;14417  REPEAT.T1B.1:
U 1345, 36F6,15 ;14418          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 1346, 3EA2,15 ;14419          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
```

```
U 1347, 3EA4,15 ;14420          MOV WR[0] TO LS[DATA.2]          ;SECOND FLOAT DATA
U 1348, BEA6,15 ;14421          MOV WR[0] TO LS[DATA.3]          ;THIRD FLOAT DATA
U 1349, 2340,15 ;14422          ROR WR[0]                        ;EXPECTED DATA
U 134A, 3E16,15 ;14423          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14424  LOOP.T1B.1:
U 134B, 0878,FC ;14425          JSR [LOAD.TRIPLE]                ;LOAD SHIFT PATTERN INTO FWR[0]
U 134C, 3614,15 ;14426          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF RIGHT FPA INSTRUCTION
U 134D, B716,95 ;14427          MOV LS[#300] TO WR[1]            ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 134E, 90F6,15 ;14428          MISC2 [TRAP.ACC] WR[0]           ;FORCE SHF RIGHT ON FWR[FT0]
U 134F, 10F6,95 ;14429          MISC2 [TRAP.ACC] WR[1]           ;FORCE LOOP SELF
U 1350, 6588,15 ;14430          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 1351, FF88,15 ;14431          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 1352, 087B,5C ;14432          JSR [STORE.0.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14433                                           ; AND THIRD.FLT
U 1353, BE8B,15 ;14434          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15),
                ;14435                                           ; EXPONENT (BITS 7-14), AND MSB (BIT 6)
U 1354, B6A8,15 ;14436          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 1355, 3616,95 ;14437          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 1356, 0869,3C ;14438          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 1357, 0934,B4 ;14439          JMP [LOOP.T1B.1]                 ;ERROR LOOP IF ENABLED
                ;14440
U 1358, FF88,15 ;14441          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1359, E58A,15 ;14442          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 135A, 36AA,15 ;14443          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
U 135B, 3616,95 ;14444          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 135C, 0869,3C ;14445          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 135D, 0934,B4 ;14446          JMP [LOOP.T1B.1]                 ;ERROR LOOP IF ENABLED
                ;14447
U 135E, FF88,15 ;14448          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 3
U 135F, 3E8B,95 ;14449          MOV WR[3] TO LS[ERROR.MASK]      ;CHECK BITS 8-15 ONLY
U 1360, 36AC,15 ;14450          MOV LS[THIRD.FLT] TO WR[0]       ;PUT THIRD FLT RESULT IN WR 0
U 1361, 3616,95 ;14451          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 1362, 0869,3C ;14452          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 1363, 0934,B4 ;14453          JMP [LOOP.T1B.1]                 ;ERROR LOOP IF ENABLED
                ;14454
U 1364, 7FF8,15 ;14455          INC LS[OS]                       ;INCREMENT INDEX
U 1365, B6F8,15 ;14456          MOV LS[OS] TO WR[0]              ;GET INDEX
                ;14457          CMP WR[0] WITH LS[#20],          ;DONE ALL PATTERNS?
U 1366, CF4A,35 ;14458           DT(LONG)&SET.ALU.CC             ; SET CONDITION CODES
U 1367, 8934,51 ;14459          JMP.IF[NEQ] TO [REPEAT.T1B.1]    ;REPEAT IF NOT
                ;14460
U 1368, FF82,15 ;14461          INC LS[ERROR.NUMBER]             ;ERROR 2
U 1369, C54D,15 ;14462          BIC LS[BIT6] TO WR[2]            ;CHANGE ERROR MASK TO CHECK BIT 6 AGAIN
U 136A, 369A,15 ;14463          MOV LS[#55555555] TO WR[0]       ;DATA IS ALTERNATING 1'S AND 0'S PATTERM
U 136B, 3EA2,15 ;14464          MOV WR[0] TO LS[DATA.1]          ;FIRST FLOAT DATA
U 136C, 3EA4,15 ;14465          MOV WR[0] TO LS[DATA.2]          ;SECOND FLOAT DATA
U 136D, BEA6,15 ;14466          MOV WR[0] TO LS[DATA.3]          ;THIRD FLOAT DATA
U 136E, 3E16,15 ;14467          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14468  LOOP.T1B.2:
U 136F, 0878,FC ;14469          JSR [LOAD.TRIPLE]                ;LOAD ALTERNATING PATTERN INTO FWR[0]
U 1370, 3618,15 ;14470          MOV LS[T12] TO WR[0]             ;GET ADDRESS OF 'MOV FWR[0] TO FQ'
U 1371, 3E1C,15 ;14471          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 1372, 0878,AC ;14472          JSR [MOV.DATA]                   ;LOAD FQ WITH DATA
U 1373, 3614,15 ;14473          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF RIGHT FPA INSTRUCTION
U 1374, B716,95 ;14474          MOV LS[#300] TO WR[1]            ;ADDRESS OF LOOP SELF FPA INSTRUCTION
```

```
U 1375, 90F6,15 ;14475          MISC2 [TRAP.ACC] WR[0]          ;FORCE SHF RIGHT ON FWR[FT0]
U 1376, 10F6,95 ;14476          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1377, B61A,15 ;14477          MOV LS[T13] TO WR[0]            ;ADDRESS OF 'MOV FQ TO FWR[2]'
U 1378, 3E1C,15 ;14478          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV ROUTINE
U 1379, 0878,AC ;14479          JSR [MOV.DATA]                  ;LOAD FWR[2] FROM FQ
U 137A, 6588,15 ;14480          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 137B, FF88,15 ;14481          INC LS[ADDRESS.DATA]            ;1 INDICATES FIRST FLOAT FAILED
U 137C, 087A,1C ;14482          JSR [STORE.2.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14483                                          ; AND THIRD.FLT
U 137D, BE8B,15 ;14484          MOV WR[2] TO LS[ERROR.MASK]     ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14485                                          ; AND EXPONENT (BITS 7-14)
U 137E, B6A8,15 ;14486          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 137F, 3616,95 ;14487          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 1380, 0869,3C ;14488          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1381, 0936,F4 ;14489          JMP [LOOP.T1B.2]                ;ERROR LOOP IF ENABLED
                ;14490
U 1382, FF88,15 ;14491          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1383, E58A,15 ;14492          CLR LS[ERROR.MASK]              ;CHECK ALL BITS
U 1384, 36AA,15 ;14493          MOV LS[SEC.FLT] TO WR[0]        ;PUT SECOND FLT RESULT IN WR 0
U 1385, 3616,95 ;14494          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 1386, 0869,3C ;14495          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1387, 0936,F4 ;14496          JMP [LOOP.T1B.2]                ;ERROR LOOP IF ENABLED
                ;14497
U 1388, FF88,15 ;14498          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 1389, 3E8B,95 ;14499          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 138A, 36AC,15 ;14500          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 138B, 3616,95 ;14501          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 138C, 0869,3C ;14502          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 138D, 0936,F4 ;14503          JMP [LOOP.T1B.2]                ;ERROR LOOP IF ENABLED
                ;14504  END.T1B:
```

```
;14505  .PAGE
;14506  .TOC    "TEST 1C 2F -> B, 2Q -> Q DESTINATION TEST"
;14507  ;++
;14508  ;
;14509  ;
;14510  ; Test Description:
;14511  ;
;14512  ;       The purpose of this test is to check the 2F -> B and 2Q -> Q destina-
;14513  ;       tion control lines to the 2901's in the FPA. This will be done by put-
;14514  ;       ting data on the direct input lines then issuing an OR of the direct
;14515  ;       inputs and zero. The destination is 2F -> B and 2Q -> Q.  The result
;14516  ;       will be read from the FWR designated by the B address lines back to the
;14517  ;       CPU and checked for error.  The Q reg is then moved to a FWR and check-
;14518  ;       ed similiarly.  The data that will be used to test the destination is a
;14519  ;       shifting 1 pattern.  What is shifted into the LSB of the data path
;14520  ;       (the LSB of the EXTENSION data path) will be tested in a later test.
;14521  ;
;14522  ; Assumptions:
;14523  ;
;14524  ;       It is assumed that all previous tests have run successfully.
;14525  ;
;14526  ; Test Steps:
;14527  ;
;14528  ;       1) Load FWR[FTO] with a shifting one pattern in each float section.
;14529  ;       2) Execute a SHF LEFT FWR[FTO] AND FQ instruction and check for correct
;14530  ;          result IN FWR[0]
;14531  ;       3) Move FQ to FWR[2] and check for correct result in FQ.
;14532  ;       4) Repeat steps 2 and 3 until all 32 bits have been tested.
;14533  ;
;14534  ; Error:
;14535  ;
;14536  ;       Note:  For the following errors, the data printed under OTHER in the
;14537  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;14538  ;       FLT) which failed.
;14539  ;
;14540  ;       Error1 - 2F -> B, 2Q -> Q(shift left) failed on FWR[FTO].
;14541  ;       Error2 - 2F -> B, 2Q -> Q(shift left) failed on FQ.
;14542  ;
;14543  ; Debug:
;14544  ;
;14545  ;       Error1: The cause of this error is likely to be either the 2901 con-
;14546  ;               trol lines or the 2901 itself. The fraction data path control
;14547  ;               lines should be 110011100.  Of these, only the DST control (I8
;14548  ;               through I6) have not been previously tested.  If these are in-
;14549  ;               correct, trace them back to the CONTROL STORE.
;14550  ;                 If the error occurs over a 4 bit boundary, check that the
;14551  ;               shift out (pin 8) of the preceeding 2901 is going high during
;14552  ;               the SHF LEFT FWR[FTO] AND FQ instruction.  If not, the preceed-
;14553  ;               ing 2901 is probably bad.
;14554  ;                 If the shift out is OK, or any other bits are failing, sus-
;14555  ;               pect the 2901 outputting the failing bit.
;14556  ;
;14557  ;       Error2: This error can be caused by either the control lines or one of
;14558  ;               the 2901's.  Check the control lines as in error 1 above.  If
;14559  ;               these are correct, the 2901 is probably at fault.
```

```
                ;14560  ;
                ;14561  ;--
                ;14562
                ;14563  ;
                ;14564  ;       TEMPORARY LS LOCATIONS USED:
                ;14565  ;
                ;14566  ; T9    MAIN MEMORY POINTER
                ;14567  ; T10   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[ZERO]'
                ;14568  ; T11   EXPECTED DATA FOR EACH FLOAT SECTION
                ;14569  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FQ'
                ;14570  ; T13   ADDRESS OF 'MOV FQ TO FWR[FT2]'
                ;14571  ; T14   ADDRESS OF MOV ABOVE IS PLACED HERE FOR MOV SUBROUTINE
                ;14572  ;
                ;14573  T.1C:
                ;14574
U 138E, B65E,15 ;14575          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;14576                                          ; STATUS WORD
U 138F, 3E80,15 ;14577          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;14578                                          ; BIT 15 INDICATES START OF TEST TO
                ;14579                                          ; CONSOLE
U 1390, 10E0,15 ;14580          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;14581  WAIT.T1C.0:
U 1391, 8939,14 ;14582          JMP [WAIT.T1C.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;14583                                          ; RESPOND
U 1392, 087E,6C ;14584          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;14585                                          ; AND SIZE BITS
U 1393, B670,15 ;14586          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;14587                                          ; OTHER DATA
U 1394, 3E80,15 ;14588          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 1395, 37CE,15 ;14589          MOV LS[T1C.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 1396, BE12,15 ;14590          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 1397, 8870,3C ;14591          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 1398, BE14,15 ;14592          MOV WR[0] TO LS[T10]            ;LOAD LS WITH ADDRESS OF 'SHF LEFT' IN-
                ;14593                                          ; STRUCTION
U 1399, 8870,3C ;14594          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 139A, BE18,15 ;14595          MOV WR[0] TO LS[T12]            ;LOAD LS WITH ADDRESS OF 'MOV FWR[0] TO
                ;14596                                          ; FQ' INSTRUCTION
U 139B, 8870,3C ;14597          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 139C, 3E1A,15 ;14598          MOV WR[0] TO LS[T13]            ;LOAD LS WITH ADDRESS OF 'MOV FQ TO
                ;14599                                          ; FWR[0]' INSTRUCTION
U 139D, 3725,15 ;14600          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 139E, A0C5,15 ;14601          COM WR[2]                       ;BITS 8-15 = 1
U 139F, C74F,15 ;14602          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;14603                                          ; BIT AND EXPONENT)
U 13A0, B725,95 ;14604          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
U 13A1, 4751,95 ;14605          BIS LS[BIT8] TO WR[3]           ;ALSO MASK OFF LSB (SHIFT INPUT FROM
                ;14606                                          ; OUTSIDE IS CHECKED IN ANOTHER TEST)
                ;14607
U 13A2, E5F8,15 ;14608          CLR LS[OS]                      ;CLEAR INDEX
                ;14609  REPEAT.T1C.1:
U 13A3, 36F6,15 ;14610          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 13A4, 3EA2,15 ;14611          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 13A5, 3EA4,15 ;14612          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 13A6, BEA6,15 ;14613          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 13A7, A3C0,15 ;14614          ROL WR[0]                       ;EXPECTED DATA
```

```
U 13A8, 3E16,15 ;14615          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14616  LOOP.T1C.1:
U 13A9, B640,15 ;14617          MOV LS[#1] TO WR[0]              ;1 IN WR
U 13AA, BE82,15 ;14618          MOV WR[0] TO LS[ERROR.NUMBER]    ;SET UP ERROR NUMBER IN LS
U 13AB, 0878,FC ;14619          JSR [LOAD.TRIPLE]                ;LOAD SHIFT PATTERN INTO FWR[0]
U 13AC, 3618,15 ;14620          MOV LS[T12] TO WR[0]             ;GET ADDRESS OF 'MOV FWR[0] TO FQ'
U 13AD, 3E1C,15 ;14621          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 13AE, 0878,AC ;14622          JSR [MOV.DATA]                   ;LOAD FQ WITH DATA
U 13AF, 3614,15 ;14623          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF LEFT FPA INSTRUCTION
U 13B0, B716,95 ;14624          MOV LS[#300] TO WR[1]            ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 13B1, 90F6,15 ;14625          MISC2 [TRAP.ACC] WR[0]           ;FORCE SHF LEFT ON FWR[FT0]
U 13B2, 10F6,95 ;14626          MISC2 [TRAP.ACC] WR[1]           ;FORCE LOOP SELF
U 13B3, 6588,15 ;14627          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 13B4, FF88,15 ;14628          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 13B5, 087B,5C ;14629          JSR [STORE.0.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14630                                           ; AND THIRD.FLT
U 13B6, BE8B,15 ;14631          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14632                                           ; AND EXPONENT (BITS 7-14)
U 13B7, B6A8,15 ;14633          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 13B8, 3616,95 ;14634          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 13B9, 0869,3C ;14635          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 13BA, 093A,94 ;14636          JMP [LOOP.T1C.1]                 ;ERROR LOOP IF ENABLED
                ;14637
U 13BB, FF88,15 ;14638          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 13BC, E58A,15 ;14639          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 13BD, 36AA,15 ;14640          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
U 13BE, 3616,95 ;14641          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 13BF, 0869,3C ;14642          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 13C0, 093A,94 ;14643          JMP [LOOP.T1C.1]                 ;ERROR LOOP IF ENABLED
                ;14644
U 13C1, FF88,15 ;14645          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 3
U 13C2, 3E8B,95 ;14646          MOV WR[3] TO LS[ERROR.MASK]      ;CHECK BITS 8-15 ONLY
U 13C3, 36AC,15 ;14647          MOV LS[THIRD.FLT] TO WR[0]       ;PUT THIRD FLT RESULT IN WR 0
U 13C4, 3616,95 ;14648          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 13C5, 0869,3C ;14649          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 13C6, 093A,94 ;14650          JMP [LOOP.T1C.1]                 ;ERROR LOOP IF ENABLED
                ;14651
U 13C7, FF82,15 ;14652          INC LS[ERROR.NUMBER]             ;ERROR 2
U 13C8, B61A,15 ;14653          MOV LS[T13] TO WR[0]             ;ADDRESS OF 'MOV FQ TO FWR[2]'
U 13C9, 3E1C,15 ;14654          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 13CA, 0878,AC ;14655          JSR [MOV.DATA]                   ;LOAD FWR[2] FROM FQ
U 13CB, 6588,15 ;14656          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 13CC, FF88,15 ;14657          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 13CD, 087A,1C ;14658          JSR [STORE.2.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14659                                           ; AND THIRD.FLT
U 13CE, BE8B,15 ;14660          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14661                                           ; AND EXPONENT (BITS 7-14)
U 13CF, B6A8,15 ;14662          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 13D0, 3616,95 ;14663          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 13D1, 0869,3C ;14664          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 13D2, 093A,94 ;14665          JMP [LOOP.T1C.1]                 ;ERROR LOOP IF ENABLED
                ;14666
U 13D3, FF88,15 ;14667          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 13D4, E58A,15 ;14668          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 13D5, 36AA,15 ;14669          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
```

```
U 13D6, 3616,95 ;14670          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 13D7, 0869,3C ;14671          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 13D8, 093A,94 ;14672          JMP [LOOP.T1C.1]                ;ERROR LOOP IF ENABLED
                ;14673
U 13D9, FF88,15 ;14674          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 13DA, 3E8B,95 ;14675          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 13DB, 36AC,15 ;14676          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 13DC, 3616,95 ;14677          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 13DD, 0869,3C ;14678          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 13DE, 093A,94 ;14679          JMP [LOOP.T1C.1]                ;ERROR LOOP IF ENABLED
                ;14680
U 13DF, 7FF8,15 ;14681          INC LS[OS]                      ;INCREMENT INDEX
U 13E0, B6F8,15 ;14682          MOV LS[OS] TO WR[0]             ;GET INDEX
                ;14683          CMP WR[0] WITH LS[#20],         ;DONE ALL PATTERNS?
U 13E1, CF4A,35 ;14684           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 13E2, 093A,31 ;14685          JMP.IF[NEQ] TO [REPEAT.T1C.1]   ;REPEAT IF NOT
                ;14686  END.T1C:
```

```
;14687  .PAGE
;14688  .TOC    "TEST 1D F/2 -> B, Q/2 -> Q DESTINATION TEST"
;14689  ;++
;14690  ;
;14691  ;
;14692  ; Test Description:
;14693  ;
;14694  ;       The purpose of this test is to check the F/2 -> B and Q/2 -> Q destina-
;14695  ;       tion control lines to the 2901's in the FPA. This will be done by put-
;14696  ;       ting data on the direct input lines then issuing an OR of the direct
;14697  ;       inputs and zero. The destination is F/2 -> B and Q/2 -> Q.  The result
;14698  ;       will be read from the FWR designated by the B address lines back to the
;14699  ;       CPU and checked for error.  The Q reg is then moved to a FWR and check-
;14700  ;       ed similiarly.  The data that will be used to test the destination is a
;14701  ;       shifting 1 pattern.  What is shifted into the LSB of the data path
;14702  ;       (the MSB of the FRACTION data path) will be tested in a later test.
;14703  ;
;14704  ; Assumptions:
;14705  ;
;14706  ;       It is assumed that all previous tests have run successfully.
;14707  ;
;14708  ; Test Steps:
;14709  ;
;14710  ;       1) Load FWR[FTO] with a shifting one pattern in each float section.
;14711  ;       2) Execute a SHF RIGHT FWR[FTO] AND FQ instruction and check for correct
;14712  ;          result IN FWR[0]
;14713  ;       3) Move FQ to FWR[2] and check for correct result in FQ.
;14714  ;       4) Repeat steps 2 and 3 until all 32 bits have been tested.
;14715  ;
;14716  ; Error:
;14717  ;
;14718  ;       Note:  For the following errors, the data printed under OTHER in the
;14719  ;       error report identifies the section (FIRST FLT, SECOND FLT, or HUGE
;14720  ;       FLT) which failed.
;14721  ;
;14722  ;       Error1 - F/2 -> B, Q/2 -> Q(shift right) failed on FWR[FTO].
;14723  ;       Error2 - F/2 -> B, Q/2 -> Q(shift right) failed on FQ.
;14724  ;
;14725  ; Debug:
;14726  ;
;14727  ;       Error1: The cause of this error is likely to be either the 2901 con-
;14728  ;               trol lines or the 2901 itself. The fraction data path control
;14729  ;               lines should be 100011100.  Of these, only the DST control (I8
;14730  ;               through I6) have not been previously tested.  If these are in-
;14731  ;               correct, trace them back to the CONTROL STORE.
;14732  ;                 If the error occurs over a 4 bit boundary, check that the
;14733  ;               shift out (pin 9) of the preceeding 2901 is going high during
;14734  ;               the SHF RIGHT FWR[FTO] AND FQ instruction. If not, the preceed-
;14735  ;               ing 2901 is probably bad.
;14736  ;                 If the shift out is OK, or any other bits are failing, sus-
;14737  ;               pect the 2901 outputting the failing bit.
;14738  ;
;14739  ;       Error2: This error can be caused by either the control lines or one of
;14740  ;               the 2901's.  Check the control lines as in error 1 above.  If
;14741  ;               these are correct, the 2901 is probably at fault.
```

```
                ;14742  ;
                ;14743  ;--
                ;14744
                ;14745  ;
                ;14746  ;       TEMPORARY LS LOCATIONS USED:
                ;14747  ;
                ;14748  ; T9    MAIN MEMORY POINTER
                ;14749  ; T10   ADDRESS OF 'SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]
                ;14750  ; T11   EXPECTED DATA FOR EACH FLOAT SECTION
                ;14751  ; T12   ADDRESS OF 'MOV FWR[FT0] TO FQ'
                ;14752  ; T13   ADDRESS OF 'MOV FQ TO FWR[FT2]'
                ;14753  ; T14   ADDRESS OF MOV ABOVE IS PLACED HERE FOR MOV SUBROUTINE
                ;14754  ;
                ;14755  T.1D:
                ;14756
U 13E3, B65E,15 ;14757          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;14758                                          ; STATUS WORD
U 13E4, 3E80,15 ;14759          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;14760                                          ; BIT 15 INDICATES START OF TEST TO
                ;14761                                          ; CONSOLE
U 13E5, 10E0,15 ;14762          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;14763  WAIT.T1D.0:
U 13E6, 893E,64 ;14764          JMP [WAIT.T1D.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;14765                                          ; RESPOND
U 13E7, 087E,6C ;14766          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;14767                                          ; AND SIZE BITS
U 13E8, B670,15 ;14768          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;14769                                          ; OTHER DATA
U 13E9, 3E80,15 ;14770          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 13EA, 37D0,15 ;14771          MOV LS[T1D.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 13EB, BE12,15 ;14772          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 13EC, 8870,3C ;14773          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 13ED, BE14,15 ;14774          MOV WR[0] TO LS[T10]            ;LOAD LS WITH ADDRESS OF 'SHF RIGHT' IN-
                ;14775                                          ; STRUCTION
U 13EE, 8870,3C ;14776          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 13EF, BE18,15 ;14777          MOV WR[C] TO LS[T12]            ;LOAD LS WITH ADDRESS OF 'MOV FWR[0] TO
                ;14778                                          ; FQ' INSTRUCTION
U 13F0, 8870,3C ;14779          JSR [L0]                        ;LOAD WR0 WITH MEMORY DATA
U 13F1, 3E1A,15 ;14780          MOV WR[0] TO LS[T13]            ;LOAD LS WITH ADDRESS OF 'MOV FQ TO
                ;14781                                          ; FWR[0]' INSTRUCTION
U 13F2, 3725,15 ;14782          MOV LS[#FFFF00FF] TO WR[2]      ;BITS 8-15 = 0
U 13F3, A0C5,15 ;14783          COM WR[2]                       ;BITS 8-15 = 1
U 13F4, C74F,15 ;14784          BIS LS[BIT7] TO WR[2]           ;ERROR MASK TO MASK OFF BITS 7-15 (SIGN
                ;14785                                          ; BIT AND EXPONENT)
U 13F5, 474D,15 ;14786          BIS LS[BIT6] TO WR[2]           ;ALSO MASK MSB (BIT SHIFTED IN FROM
                ;14787                                          ; OUTSIDE IS CHECKED IN A LATER TEST)
U 13F6, B725,95 ;14788          MOV LS[#FFFF00FF] TO WR[3]      ;SET UP ERROR MASK TO CHECK BYTE 1 ONLY
                ;14789
U 13F7, E5F8,15 ;14790          CLR LS[OS]                      ;CLEAR INDEX
                ;14791  REPEAT.T1D.1:
U 13F8, 36F6,15 ;14792          MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;LOAD DATA PATTERN IN WR
U 13F9, 3EA2,15 ;14793          MOV WR[0] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 13FA, 3EA4,15 ;14794          MOV WR[0] TO LS[DATA.2]         ;SECOND FLOAT DATA
U 13FB, BEA6,15 ;14795          MOV WR[0] TO LS[DATA.3]         ;THIRD FLOAT DATA
U 13FC, 2340,15 ;14796          ROR WR[0]                       ;EXPECTED DATA
```

```
U 13FD, 3E16,15 ;14797          MOV WR[0] TO LS[T11]             ;STORE EXPECTED DATA IN LS
                ;14798  LOOP.T1D.1:
U 13FE, B640,15 ;14799          MOV LS[#1] TO WR[0]              ;1 IN WR
U 13FF, BE82,15 ;14800          MOV WR[0] TO LS[ERROR.NUMBER]    ;SET UP ERROR NUMBER IN LS
U 1400, 0878,FC ;14801          JSR [LOAD.TRIPLE]                ;LOAD SHIFT PATTERN INTO FWR[0]
U 1401, 3618,15 ;14802          MOV LS[T12] TO WR[0]             ;GET ADDRESS OF 'MOV FWR[0] TO FQ'
U 1402, 3E1C,15 ;14803          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 1403, 0878,AC ;14804          JSR [MOV.DATA]                   ;LOAD FQ WITH DATA
U 1404, 3614,15 ;14805          MOV LS[T10] TO WR[0]             ;ADDRESS OF SHF RIGHT FPA INSTRUCTION
U 1405, B716,95 ;14806          MOV LS[#300] TO WR[1]            ;ADDRESS OF LOOP SELF FPA INSTRUCTION
U 1406, 90F6,15 ;14807          MISC2 [TRAP.ACC] WR[0]           ;FORCE SHF RIGHT ON FWR[FT0]
U 1407, 10F6,95 ;14808          MISC2 [TRAP.ACC] WR[1]           ;FORCE LOOP SELF
U 1408, 6588,15 ;14809          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 1409, FF88,15 ;14810          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 140A, 087B,5C ;14811          JSR [STORE.0.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14812                                           ; AND THIRD.FLT
U 140B, BE8B,15 ;14813          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14814                                           ; AND EXPONENT (BITS 7-14)
U 140C, B6A8,15 ;14815          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 140D, 3616,95 ;14816          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 140E, 0869,3C ;14817          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 140F, 893F,E4 ;14818          JMP [LOOP.T1D.1]                 ;ERROR LOOP IF ENABLED
                ;14819
U 1410, FF88,15 ;14820          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1411, E58A,15 ;14821          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 1412, 36AA,15 ;14822          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
U 1413, 3616,95 ;14823          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 1414, 0869,3C ;14824          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 1415, 893F,E4 ;14825          JMP [LOOP.T1D.1]                 ;ERROR LOOP IF ENABLED
                ;14826
U 1416, FF88,15 ;14827          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 3
U 1417, 3E8B,95 ;14828          MOV WR[3] TO LS[ERROR.MASK]      ;CHECK BITS 8-15 ONLY
U 1418, 36AC,15 ;14829          MOV LS[THIRD.FLT] TO WR[0]       ;PUT THIRD FLT RESULT IN WR 0
U 1419, 3616,95 ;14830          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 141A, 0869,3C ;14831          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 141B, 893F,E4 ;14832          JMP [LOOP.T1D.1]                 ;ERROR LOOP IF ENABLED
                ;14833
U 141C, FF82,15 ;14834          INC LS[ERROR.NUMBER]             ;ERROR 2
U 141D, B61A,15 ;14835          MOV LS[T13] TO WR[0]             ;ADDRESS OF 'MOV FQ TO FWR[2]'
U 141E, 3E1C,15 ;14836          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV ROUTINE
U 141F, 0878,AC ;14837          JSR [MOV.DATA]                   ;LOAD FWR[2] FROM FQ
U 1420, 6588,15 ;14838          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 1421, FF88,15 ;14839          INC LS[ADDRESS.DATA]             ;1 INDICATES FIRST FLOAT FAILED
U 1422, 087A,1C ;14840          JSR [STORE.2.TRIPLE]             ;GET RESULT IN LS FIRST.FLT, SEC.FLT
                ;14841                                           ; AND THIRD.FLT
U 1423, BE8B,15 ;14842          MOV WR[2] TO LS[ERROR.MASK]      ;DISABLE CHECKING OF SIGN BIT (BIT 15)
                ;14843                                           ; AND EXPONENT (BITS 7-14)
U 1424, B6A8,15 ;14844          MOV LS[FIRST.FLT] TO WR[0]       ;PUT FIRST FLT RESULT IN WR 0
U 1425, 3616,95 ;14845          MOV LS[T11] TO WR[1]             ;EXPECTED DATA
U 1426, 0869,3C ;14846          JSR [CHECK.RESULT]               ;CHECK FOR ERRORS
U 1427, 893F,E4 ;14847          JMP [LOOP.T1D.1]                 ;ERROR LOOP IF ENABLED
                ;14848
U 1428, FF88,15 ;14849          INC LS[ADDRESS.DATA]             ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1429, E58A,15 ;14850          CLR LS[ERROR.MASK]               ;CHECK ALL BITS
U 142A, 36AA,15 ;14851          MOV LS[SEC.FLT] TO WR[0]         ;PUT SECOND FLT RESULT IN WR 0
```

```
U 142B, 3616,95 ;14852          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 142C, 0869,3C ;14853          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 142D, 893F,E4 ;14854          JMP [LOOP.T1D.1]                ;ERROR LOOP IF ENABLED
                ;14855
U 142E, FF88,15 ;14856          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 3
U 142F, 3E8B,95 ;14857          MOV WR[3] TO LS[ERROR.MASK]     ;CHECK BITS 8-15 ONLY
U 1430, 36AC,15 ;14858          MOV LS[THIRD.FLT] TO WR[0]      ;PUT THIRD FLT RESULT IN WR 0
U 1431, 3616,95 ;14859          MOV LS[T11] TO WR[1]            ;EXPECTED DATA
U 1432, 0869,3C ;14860          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1433, 893F,E4 ;14861          JMP [LOOP.T1D.1]                ;ERROR LOOP IF ENABLED
                ;14862
U 1434, 7FF8,15 ;14863          INC LS[OS]                      ;INCREMENT INDEX
U 1435, B6F8,15 ;14864          MOV LS[OS] TO WR[0]             ;GET INDEX
                ;14865          CMP WR[0] WITH LS[#20],         ;DONE ALL PATTERNS?
U 1436, CF4A,35 ;14866           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 1437, 893F,81 ;14867          JMP.IF[NEQ] TO [REPEAT.T1D.1]   ;REPEAT IF NOT
                ;14868  END.T1D:
```

```
;14869  .PAGE
;14870  .TOC    "TEST 1E OR FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)"
;14871  ;++
;14872  ;
;14873  ;
;14874  ; Test Description:
;14875  ;
;14876  ;       This test checks the 'OR' function in the exponent data path using
;14877  ;       an OR EWR[ET1] WITH EWR[ET4] instruction.  The source is A.B, the
;14878  ;       function is A OR B (EXP.CTL), and the destination is WRITE B.  The
;14879  ;       OR function was used previously with the source D.0, so this test
;14880  ;       makes sure the OR works with non-zero operands on the B input to the
;14881  ;       ALU.  It also checks the EXP DECODE PAL for the EXP.CTL OR function
;14882  ;       which was not used previously (the D OR 0 operation used the EXP.CTL
;14883  ;       LOAD function).  The data path is to write EWR[0] and the upper bits
;14884  ;       of the FRAC path, shift this data into the 16 EXP bits, and move the
;14885  ;       data into EWR[1] or EWR[4].  After the OR instruction, the result is
;14886  ;       moved from EWR[4] to EQ, and from EQ to EWR[0] to be checked.  Data
;14887  ;       patterns are 0's, 1's, and shifting 1's.
;14888  ;
;14889  ; Assumptions:
;14890  ;
;14891  ;       It is assumed that all previous tests have run successfully.
;14892  ;
;14893  ; Test Steps:
;14894  ;
;14895  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;14896  ;          EXP data path and mov EWR[0] to EWR[1].
;14897  ;       2) Load EWR[0] again with 0's, shift this data into the EXP data path,
;14898  ;          and mov EWR[0] to EWR[4].
;14899  ;       3) Execute OR EWR[1] WITH EWR[4] instruction, mov EWR[4] to EQ, and
;14900  ;          mov EQ to EWR[0].
;14901  ;       4) Check result in EWR[0] for 0's.
;14902  ;       5) Repeat steps 2 and 3 with data of 1's in EWR[4].  Check result for
;14903  ;          1's.
;14904  ;       6) Repeat steps 1 through 3 with data of 1's in EWR[1].  Check result
;14905  ;          for 1's.
;14906  ;       7) Repeat steps 2 and 3 with 1's in EWR[4] and check result for 1's.
;14907  ;       8) Repeat step 1 to get 0's in EWR[1]
;14908  ;       9) Repeat steps 2 and 3 with shifting ones in EWR[4] until all 16 bits
;14909  ;          in the EXP data path have been tested.
;14910  ;
;14911  ; Error:
;14912  ;
;14913  ;       Note:  For the following errors, the data printed under OTHER in the
;14914  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;14915  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;14916  ;
;14917  ;       Error1 - OR EWR[ET1] WITH EWR[ET4] failed with 0's in EWR[1] and 0's
;14918  ;                in EWR[4].
;14919  ;       Error2 - OR EWR[ET1] WITH EWR[ET4] failed with 0's in EWR[1] and 1's
;14920  ;                in EWR[4].
;14921  ;       Error3 - OR EWR[ET1] WITH EWR[ET4] failed with 1's in EWR[1] and 0's
;14922  ;                in EWR[4].
;14923  ;       Error4 - OR EWR[ET1] WITH EWR[ET4] failed with 1's in EWR[1] and 1's
```

```
                    ;14924  ;                   in EWR[4].
                    ;14925  ;       Error5 - OR EWR[ET1] WITH EWR[ET4] failed with 0's in EWR[1] and
                    ;14926  ;                   shifting ones in EWR[4].
                    ;14927  ;
                    ;14928  ; Debug:
                    ;14929  ;
                    ;14930  ;       Error1 - The most likely cause of this error is the EXP DECODE PAL.
                    ;14931  ;       The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
                    ;14932  ;       0100(B) respectively during the OR instruction.  If these are incor-
                    ;14933  ;       rect, they can be traced back to the control store.
                    ;14934  ;         If these inputs are OK, check the output for a 01 1001 on I5 through
                    ;14935  ;       I0 respectively.  If these are incorrect, the PAL is bad.
                    ;14936  ;         If the EXP DECODE PAL is ok, suspect the 2901.
                    ;14937  ;       Errors 2-5 - Same as error 1, except the 2901 is more likely to be the
                    ;14938  ;       problem.
                    ;14939  ;
                    ;14940  ;--
                    ;14941
                    ;14942  ;
                    ;14943  ;       TEMPORARY LS LOCATIONS USED:
                    ;14944  ;
                    ;14945  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                    ;14946  ; T9    MAIN MEMORY POINTER
                    ;14947  ; T10   ADDRESS OF 'OR EWR[ET1] WITH EWR[ET4]'
                    ;14948  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                    ;14949  ; T12   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET4]'
                    ;14950  ; T13   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET1]'
                    ;14951  ; T14   ADDRESS OF MOVE GOES HERE FOR MOV SUBROUTINE
                    ;14952  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                    ;14953  ; T57   ADDRESS OF 'MOV EWR[ET4] TO EQ'
                    ;14954  ; T58   ADDRESS OF 'MOV EQ TO EWR[ET0]'
                    ;14955  ;
                    ;14956  T.1E:
                    ;14957
U 1438, B65E,15     ;14958          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;14959                                          ; STATUS WORD
U 1439, 3E80,15     ;14960          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;14961                                          ; BIT 15 INDICATES START OF TEST TO
                    ;14962                                          ; CONSOLE
U 143A, 10E0,15     ;14963          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;14964  WAIT.T1E.0:
U 143B, 0943,B4     ;14965          JMP [WAIT.T1E.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;14966                                          ; RESPOND
U 143C, 087E,6C     ;14967          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;14968                                          ; AND SIZE BITS
U 143D, B670,15     ;14969          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                    ;14970                                          ; OTHER DATA
U 143E, 3E80,15     ;14971          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 143F, B724,15     ;14972          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 1440, C75E,15     ;14973          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 1441, C54E,15     ;14974          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 1442, 3E8A,15     ;14975          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 1443, B7D2,15     ;14976          MOV LS[T1E.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 1444, BE12,15     ;14977          MOV WR[0] TO LS[T9]              ;SET UP POINTER IN LS
U 1445, 37EA,15     ;14978          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
```

```
                ;14979                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 1446, 3E10,15 ;14980          MOV WR[0] TO LS[T8]              ;STORE IN LS T8
U 1447, 8870,3C ;14981          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'OR' ROUTINE
U 1448, BE14,15 ;14982          MOV WR[0] TO LS[T10]             ;STORE IN LS T10
U 1449, 37EC,15 ;14983          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;14984                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 144A, 3E16,15 ;14985          MOV WR[0] TO LS[T11]             ;STORE IN LS T11
U 144B, 8870,3C ;14986          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;14987                                          ; TO EWR[4]' INSTRUCTION
U 144C, BE18,15 ;14988          MOV WR[0] TO LS[T12]             ;STORE IN LS T12
U 144D, 8870,3C ;14989          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;14990                                          ; TO EWR[1]' INSTRUCTION
U 144E, 3E1A,15 ;14991          MOV WR[0] TO LS[T13]             ;STORE IN LS T13
U 144F, 8870,3C ;14992          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[4]
                ;14993                                          ; TO EQ' INSTRUCTION
U 1450, 3EAE,15 ;14994          MOV WR[0] TO LS[T57]             ;STORE IN LS T57
U 1451, 8870,3C ;14995          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EQ TO
                ;14996                                          ; EWR[0]' INSTRUCTION
U 1452, 3EB0,15 ;14997          MOV WR[0] TO LS[T58]             ;STORE IN LS T58
U 1453, B7EE,15 ;14998          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;14999                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 1454, BE1E,15 ;15000          MOV WR[0] TO LS[T15]             ;STORE IN LS T15
                ;15001
U 1455, E5A2,15 ;15002          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 1456, E5A4,15 ;15003          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 1457, 65A6,15 ;15004          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 1458, 087D,CC ;15005          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1459, B61A,15 ;15006          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EWR[1]
U 145A, 3E1C,15 ;15007          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV SUBROUTINE
U 145B, 0878,AC ;15008          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[1]
U 145C, 8948,CC ;15009          JSR [LOOP.T1E.1.5]              ;DO 'OR' OF 0 WITH 0
                ;15010
U 145D, FF82,15 ;15011          INC LS[ERROR.NUMBER]            ;ERROR 2
U 145E, FDA2,15 ;15012          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 145F, FDA4,15 ;15013          COM LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 1460, 7DA6,15 ;15014          COM LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 1461, 8948,CC ;15015          JSR [LOOP.T1E.1.5]              ;DO 'OR' OF 0 WITH 1'S
                ;15016                                          ; SET
                ;15017
U 1462, FF82,15 ;15018          INC LS[ERROR.NUMBER]            ;ERROR 3
U 1463, 087D,CC ;15019          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1464, B61A,15 ;15020          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EWR[1]
U 1465, 3E1C,15 ;15021          MOV WR[0] TO LS[T14]             ;SET UP FOR MOV SUBROUTINE
U 1466, 0878,AC ;15022          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[1] (ALL 1'S)
U 1467, E5A2,15 ;15023          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1468, 8948,CC ;15024          JSR [LOOP.T1E.1.5]              ;DO 'OR' OF 1'S WITH 0'S
                ;15025
U 1469, FF82,15 ;15026          INC LS[ERROR.NUMBER]            ;ERROR 4
U 146A, FDA2,15 ;15027          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 146B, 8948,CC ;15028          JSR [LOOP.T1E.1.5]              ;DO 'OR' OF 1'S WITH 1'S
                ;15029
U 146C, FF82,15 ;15030          INC LS[ERROR.NUMBER]            ;ERROR 5
U 146D, B64A,15 ;15031          MOV LS[#20] TO WR[0]            ;PUT AN 20 IN WR
U 146E, 2100,15 ;15032          DEC WR[0]                       ;1F(H) NOW IN WR
U 146F, 3EF8,15 ;15033          MOV WR[0] TO LS[OS]              ;START INDEX AT 1F (POINTS TO BIT 31
```

```
                ;15034                                          ; SET).  BIT 31 WILL END UP IN LSB OF
                ;15035                                          ; EXPONENT.
U 1470, E5A2,15 ;15036          CLR LS[DATA.1]                  ;FIRST FLOAT DATA
U 1471, 087D,CC ;15037          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1472, B61A,15 ;15038          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EWR[1]
U 1473, 3E1C,15 ;15039          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1474, 0878,AC ;15040          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[1] (ALL 0'S)
U 1475, 65A6,15 ;15041          CLR LS[DATA.3]                  ;EXPECTED DATA WILL BE 0 IN UPPER
                ;15042                                          ; 8 BITS OF EXP
U 1476, B64F,15 ;15043          MOV LS[BIT7] TO WR[2]           ;EXPECTED DATA FOR LOWER 8 BITS
                ;15044  REPEAT.T1E.5A:
U 1477, BEA5,15 ;15045          MOV WR[2] TO LS[DATA.2]         ;STORE EXPECTED DATA IN LS
                ;15046
U 1478, B6F6,95 ;15047          MOV LS[SHIFT.OS(4-0)] TO WR[1]  ;LOAD DATA PATTERN IN WR
U 1479, BEA2,95 ;15048          MOV WR[1] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 147A, 8948,CC ;15049          JSR [LOOP.T1E.1.5]              ;GO OR 0'S WITH SHIFTING 1'S
U 147B, 7FF8,15 ;15050          INC LS[OS]                      ;INCREMENT INDEX
U 147C, 23C1,15 ;15051          ROL WR[2]                       ;NEXT EXPECTED DATA FOR LOWER 8 BITS
                ;15052          BIT LS[BIT15] WITH WR[2],       ;WILL DATA BE IN UPPER 8 BITS?
U 147D, D95F,35 ;15053           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 147E, 0947,79 ;15054          JMP.IF[EQL] TO [REPEAT.T1E.5A]  ;REPEAT IF NOT
                ;15055
U 147F, E5A4,15 ;15056          CLR LS[DATA.2]                  ;EXPECTED DATA WILL BE 0 IN LOWER
                ;15057                                          ; 8 BITS OF EXP
U 1480, B64F,15 ;15058          MOV LS[BIT7] TO WR[2]           ;EXPECTED DATA FOR UPPER 8 BITS
                ;15059  REPEAT.T1E.5B:
U 1481, 3EA7,15 ;15060          MOV WR[2] TO LS[DATA.3]         ;STORE EXPECTED DATA IN LS
                ;15061
U 1482, B6F6,95 ;15062          MOV LS[SHIFT.OS(4-0)] TO WR[1]  ;LOAD DATA PATTERN IN WR
U 1483, BEA2,95 ;15063          MOV WR[1] TO LS[DATA.1]         ;FIRST FLOAT DATA
U 1484, 8948,CC ;15064          JSR [LOOP.T1E.1.5]              ;GO OR 0'S WITH SHIFTING 1'S
U 1485, 7FF8,15 ;15065          INC LS[OS]                      ;INCREMENT INDEX
U 1486, 23C1,15 ;15066          ROL WR[2]                       ;NEXT EXPECTED DATA FOR LOWER 8 BITS
U 1487, B64A,15 ;15067          MOV LS[#20] TO WR[0]            ;GET DATA OF 20
U 1488, 4748,15 ;15068          BIS LS[#10] TO WR[0]            ;NOW HAVE 30 IN WR
                ;15069          CMP WR[0] WITH LS[OS],          ;DONE ALL PATTERNS?
U 1489, CFF8,35 ;15070           DT(LONG)&SET.ALU.CC            ; SET CONDITION CODES
U 148A, 8948,11 ;15071          JMP.IF[NEQ] TO [REPEAT.T1E.5B]  ;REPEAT IF NOT
                ;15072
U 148B, 894A,94 ;15073          JMP [END.T1E]                   ;DONE WITH 'OR' TEST
                ;15074
                ;15075  LOOP.T1E.1.5:
U 148C, 6588,15 ;15076          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 148D, FF88,15 ;15077          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 148E, 087D,CC ;15078          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 148F, 3618,15 ;15079          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EWR[4]
U 1490, 3E1C,15 ;15080          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1491, 0878,AC ;15081          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[4]
U 1492, 3614,15 ;15082          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'OR' INSTRUCTION
U 1493, B716,95 ;15083          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 1494, 90F6,15 ;15084          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'OR EWR[ET1] WITH EWR[ET4]'
U 1495, 10F6,95 ;15085          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1496, B6AE,15 ;15086          MOV LS[T57] TO WR[0]            ;ADDRESS OF MOV EWR[4] TO EQ
U 1497, 3E1C,15 ;15087          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1498, 0878,AC ;15088          JSR [MOV.DATA]                  ;MOV EWR[4] TO EQ
```

```
U 1499, B6B0,15 ;15089          MOV LS[T58] TO WR[0]          ;ADDRESS OF MOV EQ TO EWR[ET0]
U 149A, 3E1C,15 ;15090          MOV WR[0] TO LS[T14]          ;SET UP FOR MOV SUBROUTINE
U 149B, 0878,AC ;15091          JSR [MOV.DATA]                ;MOV EQ TO EWR[0]
                ;15092
U 149C, 087B,5C ;15093          JSR [STORE.0.TRIPLE]          ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15094                                        ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 149D, B6A8,15 ;15095          MOV LS[FIRST.FLT] TO WR[0]    ;PUT FIRST FLT RESULT IN WR 0
U 149E, 36A4,95 ;15096          MOV LS[DATA.2] TO WR[1]       ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 149F, 0869,3C ;15097          JSR [CHECK.RESULT]            ;CHECK FOR ERRORS
U 14A0, 0948,C4 ;15098          JMP [LOOP.T1E.1.5]            ;ERROR LOOP IF ENABLED
                ;15099
U 14A1, FF88,15 ;15100          INC LS[ADDRESS.DATA]          ;INC PRINTOUT UNDER OTHER DATA TO 2
U 14A2, 8877,4C ;15101          JSR [SHIFT.RIGHT.8]           ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15102                                        ; LOWER 8 BITS
U 14A3, 087B,5C ;15103          JSR [STORE.0.TRIPLE]          ;GET RESULT AGAIN
U 14A4, B6A8,15 ;15104          MOV LS[FIRST.FLT] TO WR[0]    ;PUT FIRST FLT RESULT IN WR 0
U 14A5, B6A6,95 ;15105          MOV LS[DATA.3] TO WR[1]       ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 14A6, 0869,3C ;15106          JSR [CHECK.RESULT]            ;CHECK FOR ERRORS
U 14A7, 0948,C4 ;15107          JMP [LOOP.T1E.1.5]            ;ERROR LOOP IF ENABLED
U 14A8, 5B00,14 ;15108          RETURN                        ;DONE WITH THIS DATA PATTERN
                ;15109  END.T1E:
```

```
;15110  .PAGE
;15111  .TOC    "TEST 1F ADD FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)"
;15112  ;++
;15113  ;
;15114  ; Test Description:
;15115  ;
;15116  ;       This test checks the 'ADD' function in the exponent data path using
;15117  ;       an ADD EWR[ET2] WITH EWR[ET0] instruction.  The source is A.B, the
;15118  ;       function is ADD (EXP.CTL), and the destination is WRITE B.  The 2901
;15119  ;       ADD function was used previously with the instruction A+B+FCOUT.  This
;15120  ;       test checks that the EXP DECODE PAL is decoding the ADD function cor-
;15121  ;       rectly.  The data path is to write EWR[0] and the upper bits of the
;15122  ;       FRAC path, shift this data into the 16 EXP bits, and either move the
;15123  ;       data into EWR[2] or leave it in EWR[0].  After the ADD instruction, the
;15124  ;       result is stored from EWR[0] and checked.  Data patterns are 0's and
;15125  ;       1's.  Additional patterns are not necessary since the 2901 operation
;15126  ;       has been checked previously.
;15127  ;
;15128  ; Assumptions:
;15129  ;
;15130  ;       It is assumed that all previous tests have run successfully.
;15131  ;
;15132  ; Test Steps:
;15133  ;
;15134  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;15135  ;          EXP data path and mov EWR[0] to EWR[2].
;15136  ;       2) Load EWR[0] again with 0's and shift this data into the EXP data
;15137  ;          path.
;15138  ;       3) Execute ADD EWR[2] TO EWR[0] instruction.
;15139  ;       4) Check result in EWR[0] for 0's.
;15140  ;       5) Repeat steps 2 and 3 with data of 1's in EWR[0].  Check result for
;15141  ;          1's.
;15142  ;       6) Repeat steps 1 through 3 with data of 1's in EWR[2].  Check result
;15143  ;          for 1's.
;15144  ;       7) Repeat steps 2 and 3 with 1's in EWR[2] and check result for 1's in
;15145  ;          the upper 8 bits and FE(H) in the lower 8 bits.
;15146  ;
;15147  ; Error:
;15148  ;
;15149  ;       Note:  For the following errors, the data printed under OTHER in the
;15150  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15151  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15152  ;
;15153  ;       Error1 - ADD EWR[2] TO EWR[0] failed with 0's in EWR[2] and 0's in
;15154  ;                EWR[0].
;15155  ;       Error2 - ADD EWR[2] TO EWR[0] failed with 0's in EWR[2] and 1's in
;15156  ;                EWR[0].
;15157  ;       Error3 - ADD EWR[2] TO EWR[0] failed with 1's in EWR[2] and 0's in
;15158  ;                EWR[0].
;15159  ;       Error4 - ADD EWR[2] TO EWR[0] failed with 1's in EWR[2] and 1's in
;15160  ;                EWR[0].
;15161  ;
;15162  ; Debug:
;15163  ;
;15164  ;       Error1-4 - The most likely cause of this error is the EXP DECODE PAL.
```

```
                ;15165  ;       The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
                ;15166  ;       0011(B) respectively during the ADD instruction.  If these are incor-
                ;15167  ;       rect, they can be traced back to the control store.
                ;15168  ;         If these inputs are OK, check the output for a 00 0001 on I5 through
                ;15169  ;       I0 respectively.  If these are incorrect, the PAL is bad.
                ;15170  ;
                ;15171  ;--
                ;15172
                ;15173  ;
                ;15174  ;       TEMPORARY LS LOCATIONS USED:
                ;15175  ;
                ;15176  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;15177  ; T9    MAIN MEMORY POINTER
                ;15178  ; T10   ADDRESS OF 'ADD EWR[ET2] TO EWR[ET0]'
                ;15179  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;15180  ; T14   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET2]'
                ;15181  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;15182  ;
                ;15183  T.1F:
                ;15184
U 14A9, B65E,15 ;15185          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;15186                                          ; STATUS WORD
U 14AA, 3E80,15 ;15187          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;15188                                          ; BIT 15 INDICATES START OF TEST TO
                ;15189                                          ; CONSOLE
U 14AB, 10E0,15 ;15190          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;15191  WAIT.T1F.0:
U 14AC, 894A,C4 ;15192          JMP [WAIT.T1F.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;15193                                          ; RESPOND
U 14AD, 087E,6C ;15194          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;15195                                          ; AND SIZE BITS
U 14AE, B670,15 ;15196          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;15197                                          ; OTHER DATA
U 14AF, 3E80,15 ;15198          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 14B0, B724,15 ;15199          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 14B1, C75E,15 ;15200          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 14B2, C54E,15 ;15201          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 14B3, 3E8A,15 ;15202          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 14B4, B7D4,15 ;15203          MOV LS[T1F.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 14B5, BE12,15 ;15204          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 14B6, 37EA,15 ;15205          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;15206                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 14B7, 3E10,15 ;15207          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 14B8, 8870,3C ;15208          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'ADD' ROUTINE
U 14B9, BE14,15 ;15209          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 14BA, 37EC,15 ;15210          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15211                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 14BB, 3E16,15 ;15212          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 14BC, 8870,3C ;15213          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15214                                          ; TO EWR[2]' INSTRUCTION
U 14BD, 3E1C,15 ;15215          MOV WR[0] TO LS[T14]            ;STORE IN LS T14
U 14BE, B7EE,15 ;15216          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15217                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 14BF, BE1E,15 ;15218          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
                ;15219
```

```
U 14C0, E5A2,15 ;15220          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 14C1, E5A4,15 ;15221          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 14C2, 65A6,15 ;15222          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 14C3, 087D,CC ;15223          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 14C4, 0878,AC ;15224          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[2]
U 14C5, 094D,7C ;15225          JSR [LOOP.T1F.1.4]              ;DO 'ADD' OF 0 TO 0
                ;15226
U 14C6, FF82,15 ;15227          INC LS[ERROR.NUMBER]            ;ERROR 2
U 14C7, FDA2,15 ;15228          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 14C8, FDA4,15 ;15229          COM LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 14C9, 7DA6,15 ;15230          COM LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 14CA, 094D,7C ;15231          JSR [LOOP.T1F.1.4]              ;DO 'ADD' OF 0 TO 1'S
                ;15232                                          ; SET
                ;15233
U 14CB, FF82,15 ;15234          INC LS[ERROR.NUMBER]            ;ERROR 3
U 14CC, 087D,CC ;15235          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 14CD, 0878,AC ;15236          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[2] (ALL 1'S)
U 14CE, E5A2,15 ;15237          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 14CF, 094D,7C ;15238          JSR [LOOP.T1F.1.4]              ;DO 'ADD' OF 1'S TO 0'S
                ;15239
U 14D0, FF82,15 ;15240          INC LS[ERROR.NUMBER]            ;ERROR 4
U 14D1, FDA2,15 ;15241          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 14D2, B69E,15 ;15242          MOV LS[ONES] TO WR[0]           ;ALL 1'S IN WR 0
U 14D3, C54E,15 ;15243          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 14D4, 3EA4,15 ;15244          MOV WR[0] TO LS[DATA.2]         ;EXPECTED DATA FOR LOWER 8 BITS (FE(H)
                ;15245                                          ; IN BITS 7-14)
U 14D5, 094D,7C ;15246          JSR [LOOP.T1F.1.4]              ;DO 'ADD' OF 1'S TO 1'S
                ;15247
U 14D6, 894E,B4 ;15248          JMP [END.T1F]                   ;DONE WITH 'ADD' TEST
                ;15249
                ;15250  LOOP.T1F.1.4:
U 14D7, 6588,15 ;15251          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 14D8, FF88,15 ;15252          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 14D9, 087D,CC ;15253          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 14DA, 3614,15 ;15254          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'ADD' INSTRUCTION
U 14DB, B716,95 ;15255          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 14DC, 90F6,15 ;15256          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'ADD EWR[ET2] TO EWR[ET0]'
U 14DD, 10F6,95 ;15257          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
                ;15258
U 14DE, 087B,5C ;15259          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15260                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 14DF, B6A8,15 ;15261          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 14E0, 36A4,95 ;15262          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 14E1, 0869,3C ;15263          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 14E2, 894D,74 ;15264          JMP [LOOP.T1F.1.4]              ;ERROR LOOP IF ENABLED
                ;15265
U 14E3, FF88,15 ;15266          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 14E4, 8877,4C ;15267          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15268                                          ; LOWER 8 BITS
U 14E5, 087B,5C ;15269          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 14E6, B6A8,15 ;15270          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 14E7, B6A6,95 ;15271          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 14E8, 0869,3C ;15272          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 14E9, 894D,74 ;15273          JMP [LOOP.T1F.1.4]              ;ERROR LOOP IF ENABLED
U 14EA, 5B00,14 ;15274          RETURN                          ;DONE WITH THIS DATA PATTERN
```

```
                ;15275  END.T1F:
```

```
;15276  .PAGE
;15277  .TOC    "TEST 20 Q+1 FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)"
;15278  ;++
;15279  ;
;15280  ; Test Description:
;15281  ;
;15282  ;       This test checks the 'INC EQ' function in the EXP DECODE PAL using
;15283  ;       an INC EQ instruction.  The source is 0.Q, the function is R PLUS S,
;15284  ;       and the destination is WRITE Q.  CIN is forced high by the EXP DECODE
;15285  ;       PAL to produce the increment.  The ADD function of the exponent data
;15286  ;       path and 2901's was tested previously.  This test checks that the EXP
;15287  ;       DECODE PAL will set the 2901 control lines properly during an INC EQ
;15288  ;       instruction and will set CIN EXP 0 to a 1.  The data path is to write
;15289  ;       EWR[0] and the upper bits of the FRAC path, shift this data into the
;15290  ;       16 EXP bits, and move the data to EQ.  After the INC EQ instruction,
;15291  ;       the result is moved from EQ to EWR[0] to be checked.  Data patterns are
;15292  ;       0's and 1's.
;15293  ;
;15294  ; Assumptions:
;15295  ;
;15296  ;       It is assumed that all previous tests have run successfully.
;15297  ;
;15298  ; Test Steps:
;15299  ;
;15300  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;15301  ;          EXP data path and mov EWR[0] to EQ.
;15302  ;       2) Execute INC EQ instruction, and mov EQ to EWR[0].
;15303  ;       3) Check result in EWR[0] for a 1 (LSB set).
;15304  ;       4) Repeat steps 1 and 2 with data of 1's in EWR[0].  Check result for
;15305  ;          0's.
;15306  ;
;15307  ; Error:
;15308  ;
;15309  ;       Note:  For the following errors, the data printed under OTHER in the
;15310  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15311  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15312  ;
;15313  ;       Error1 - INC EQ with EQ=0's failed.
;15314  ;       Error1 - INC EQ with EQ=1's failed.
;15315  ;
;15316  ; Debug:
;15317  ;
;15318  ;       Errors 1 and 2 - The most likely cause of this error is the EXP DECODE
;15319  ;       PAL.  The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
;15320  ;       1001(B) respectively during the INC EQ instruction.  If these are in-
;15321  ;       correct, they can be traced back to the control store.
;15322  ;         If these inputs are OK, check the output for a 00 0010 on I5 through
;15323  ;       I0 respectively during the INC EQ instruction.  Also check that CIN
;15324  ;       EXP0 H is high.  If these are incorrect, the PAL is bad.
;15325  ;
;15326  ;--
;15327
;15328  ;
;15329  ;       TEMPORARY LS LOCATIONS USED:
;15330  ;
```

```
                ;15331  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;15332  ; T9    MAIN MEMORY POINTER
                ;15333  ; T10   ADDRESS OF 'INC EQ'
                ;15334  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;15335  ; T12   ADDRESS OF 'MOV EWR[ET0] TO EQ'
                ;15336  ; T13   ADDRESS OF 'MOV EQ TO EWR[ET0]'
                ;15337  ; T14   ADDRESS OF MOVE GOES HERE FOR MOV SUBROUTINE
                ;15338  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;15339  ;
                ;15340  T.20:
                ;15341
U 14EB, B65E,15 ;15342          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;15343                                          ; STATUS WORD
U 14EC, 3E80,15 ;15344          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;15345                                          ; BIT 15 INDICATES START OF TEST TO
                ;15346                                          ; CONSOLE
U 14ED, 10E0,15 ;15347          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;15348  WAIT.T20.0:
U 14EE, 894E,E4 ;15349          JMP [WAIT.T20.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;15350                                          ; RESPOND
U 14EF, 087E,6C ;15351          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;15352                                          ; AND SIZE BITS
U 14F0, B670,15 ;15353          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;15354                                          ; OTHER DATA
U 14F1, 3E80,15 ;15355          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 14F2, B724,15 ;15356          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 14F3, C75E,15 ;15357          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 14F4, C54E,15 ;15358          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 14F5, 3E8A,15 ;15359          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 14F6, 37D6,15 ;15360          MOV LS[T20.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 14F7, BE12,15 ;15361          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 14F8, 37EA,15 ;15362          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;15363                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 14F9, 3E10,15 ;15364          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 14FA, 8870,3C ;15365          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'INC EQ'
                ;15366                                          ; ROUTINE
U 14FB, BE14,15 ;15367          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 14FC, 37EC,15 ;15368          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15369                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 14FD, 3E16,15 ;15370          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 14FE, 8870,3C ;15371          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15372                                          ; TO EQ' INSTRUCTION
U 14FF, BE18,15 ;15373          MOV WR[0] TO LS[T12]            ;STORE IN LS T12
U 1500, 8870,3C ;15374          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EQ TO
                ;15375                                          ; EWR[ET0]' INSTRUCTION
U 1501, 3E1A,15 ;15376          MOV WR[0] TO LS[T13]            ;STORE IN LS T13
U 1502, B7EE,15 ;15377          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15378                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 1503, BE1E,15 ;15379          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
                ;15380
U 1504, E5A2,15 ;15381          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 1505, 364E,15 ;15382          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 IN WR
U 1506, 3EA4,15 ;15383          MOV WR[0] TO LS[DATA.2]         ;01(H) IN EXPECTED LOWER 8 BITS OF EXP
                ;15384                                          ; (EXP LSB IS BIT 7 OF RESULT)
U 1507, 65A6,15 ;15385          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
```

```
U 1508, 0950,EC ;15386          JSR [LOOP.T20.1.2]              ;DO 'INC EQ' WITH EQ=0'S
                ;15387
U 1509, FF82,15 ;15388          INC LS[ERROR.NUMBER]            ;ERROR 2
U 150A, FDA2,15 ;15389          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 150B, E5A4,15 ;15390          CLR LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 150C, 0950,EC ;15391          JSR [LOOP.T20.1.2]              ;DO 'INC EQ' WITH EQ=1'S
                ;15392
U 150D, 0952,84 ;15393          JMP [END.T20]                   ;DONE WITH INC EQ TEST
                ;15394
                ;15395  LOOP.T20.1.2:
U 150E, 6588,15 ;15396          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 150F, FF88,15 ;15397          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 1510, 087D,CC ;15398          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 1511, 3618,15 ;15399          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EQ
U 1512, 3E1C,15 ;15400          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1513, 0878,AC ;15401          JSR [MOV.DATA]                  ;MOV EWR[0] TO EQ
U 1514, 3614,15 ;15402          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'INC EQ' INSTRUCTION
U 1515, B716,95 ;15403          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 1516, 90F6,15 ;15404          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'INC EQ'
U 1517, 10F6,95 ;15405          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1518, B61A,15 ;15406          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EQ TO EWR[0]
U 1519, 3E1C,15 ;15407          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 151A, 0878,AC ;15408          JSR [MOV.DATA]                  ;MOV EQ TO EWR[0]
                ;15409
U 151B, 087B,5C ;15410          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15411                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 151C, B6A8,15 ;15412          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 151D, 36A4,95 ;15413          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 151E, 0869,3C ;15414          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 151F, 8950,E4 ;15415          JMP [LOOP.T20.1.2]              ;ERROR LOOP IF ENABLED
                ;15416
U 1520, FF88,15 ;15417          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1521, 8877,4C ;15418          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15419                                          ; LOWER 8 BITS
U 1522, 087B,5C ;15420          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1523, B6A8,15 ;15421          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1524, B6A6,95 ;15422          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1525, 0869,3C ;15423          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1526, 8950,E4 ;15424          JMP [LOOP.T20.1.2]              ;ERROR LOOP IF ENABLED
U 1527, 5B00,14 ;15425          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;15426  END.T20:
```

```
;15427  .PAGE
;15428  .TOC    "TEST 21 Q-1 FUNCTION ON EXPONENT DATA PATH TEST(M8389 FPA MODULE)"
;15429  ;++
;15430  ;
;15431  ; Test Description:
;15432  ;
;15433  ;       This test checks the 'DEC EQ' function in the EXP DECODE PAL using
;15434  ;       an DEC EQ instruction.  The source is 0.Q, the function is R MINUS S,
;15435  ;       and the destination is WRITE Q.  CIN is forced low by the EXP DECODE
;15436  ;       PAL to produce the decrement.  The SUB function of the exponent data
;15437  ;       path and 2901's was tested previously but CIN was forced high for that
;15438  ;       test.  This test checks that the EXP DECODE PAL will set the 2901
;15439  ;       control lines properly during an DEC EQ instruction and will set CIN
;15440  ;       EXP 0 to a 0.  This test also checks that the SUB function of the ex-
;15441  ;       ponent data path works properly when CIN is a 0.  The data path is to
;15442  ;       write EWR[0] and the upper bits of the FRAC path, shift this data into
;15443  ;       the 16 EXP bits, and move the data to EQ.  After the INC EQ instruc-
;15444  ;       tion, the result is moved from EQ to EWR[0] to be checked.  Data pat-
;15445  ;       terns are 0's and 1's.
;15446  ;
;15447  ; Assumptions:
;15448  ;
;15449  ;       It is assumed that all previous tests have run successfully.
;15450  ;
;15451  ; Test Steps:
;15452  ;
;15453  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;15454  ;          EXP data path and mov EWR[0] to EQ.
;15455  ;       2) Execute DEC EQ instruction, and mov EQ to EWR[0].
;15456  ;       3) Check result in EWR[0] for all 1's.
;15457  ;       4) Repeat steps 1 and 2 with data of 1's in EWR[0].  Check result for
;15458  ;          FFFE(H).
;15459  ;
;15460  ; Error:
;15461  ;
;15462  ;       Note:  For the following errors, the data printed under OTHER in the
;15463  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15464  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15465  ;
;15466  ;       Error1 - DEC EQ with EQ=0's failed.
;15467  ;       Error1 - DEC EQ with EQ=1's failed.
;15468  ;
;15469  ; Debug:
;15470  ;
;15471  ;       Errors 1 and 2 - The most likely cause of this error is the EXP DECODE
;15472  ;       PAL.  The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
;15473  ;       1000(B) respectively during the DEC EQ instruction.  If these are in-
;15474  ;       correct, they can be traced back to the control store.
;15475  ;         If these inputs are OK, check the output for a 00 1010 on I5 through
;15476  ;       I0 respectively during the DEC EQ instruction.  Also check that CIN
;15477  ;       EXP0 H is LOW.  If these are incorrect, the PAL is bad.
;15478  ;         A less likely possibility is that the 2901 itself is bad.  This would
;15479  ;       most likely show up in the lower 8 bits (OTHER=1) and would indicate
;15480  ;       that the 2901 associated with bits 3-0 of the exponent data path is
;15481  ;       bad.  The SUB function was not tested with CIN EXP0=0 previously.
```

```
                    ;15482  ;
                    ;15483  ;--
                    ;15484
                    ;15485  ;
                    ;15486  ;       TEMPORARY LS LOCATIONS USED:
                    ;15487  ;
                    ;15488  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                    ;15489  ; T9    MAIN MEMORY POINTER
                    ;15490  ; T10   ADDRESS OF 'DEC EQ'
                    ;15491  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                    ;15492  ; T12   ADDRESS OF 'MOV EWR[ET0] TO EQ'
                    ;15493  ; T13   ADDRESS OF 'MOV EQ TO EWR[ET0]'
                    ;15494  ; T14   ADDRESS OF MOVE GOES HERE FOR MOV SUBROUTINE
                    ;15495  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                    ;15496  ;
                    ;15497  T.21:
                    ;15498
U 1528, B65E,15     ;15499          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;15500                                          ; STATUS WORD
U 1529, 3E80,15     ;15501          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;15502                                          ; BIT 15 INDICATES START OF TEST TO
                    ;15503                                          ; CONSOLE
U 152A, 10E0,15     ;15504          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;15505  WAIT.T21.0:
U 152B, 0952,B4     ;15506          JMP [WAIT.T21.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;15507                                          ; RESPOND
U 152C, 087E,6C     ;15508          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;15509                                          ; AND SIZE BITS
U 152D, B670,15     ;15510          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                    ;15511                                          ; OTHER DATA
U 152E, 3E80,15     ;15512          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 152F, B724,15     ;15513          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 1530, C75E,15     ;15514          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 1531, C54E,15     ;15515          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 1532, 3E8A,15     ;15516          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 1533, B7D8,15     ;15517          MOV LS[T21.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 1534, BE12,15     ;15518          MOV WR[0] TO LS[T9]              ;SET UP POINTER IN LS
U 1535, 37EA,15     ;15519          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                    ;15520                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 1536, 3E10,15     ;15521          MOV WR[0] TO LS[T8]              ;STORE IN LS T8
U 1537, 8870,3C     ;15522          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'DEC EQ'
                    ;15523                                          ; ROUTINE
U 1538, BE14,15     ;15524          MOV WR[0] TO LS[T10]             ;STORE IN LS T10
U 1539, 37EC,15     ;15525          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                    ;15526                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 153A, 3E16,15     ;15527          MOV WR[0] TO LS[T11]             ;STORE IN LS T11
U 153B, 8870,3C     ;15528          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                    ;15529                                          ; TO EQ' INSTRUCTION
U 153C, BE18,15     ;15530          MOV WR[0] TO LS[T12]             ;STORE IN LS T12
U 153D, 8870,3C     ;15531          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EQ TO
                    ;15532                                          ; EWR[ET0]' INSTRUCTION
U 153E, 3E1A,15     ;15533          MOV WR[0] TO LS[T13]             ;STORE IN LS T13
U 153F, B7EE,15     ;15534          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                    ;15535                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 1540, BE1E,15     ;15536          MOV WR[0] TO LS[T15]             ;STORE IN LS T15
```

```
                ;15537
U 1541, E5A2,15 ;15538          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 1542, B69E,15 ;15539          MOV LS[ONES] TO WR[0]           ;ONES IN WR 0
U 1543, 3EA4,15 ;15540          MOV WR[0] TO LS[DATA.2]         ;ONES IN EXPECTED LOWER 8 BITS OF EXP
U 1544, BEA6,15 ;15541          MOV WR[0] TO LS[DATA.3]         ;ONES IN EXPECTED UPPER 8 BITS OF EXP
U 1545, 0954,CC ;15542          JSR [LOOP.T21.1.2]              ;DO 'DEC EQ' WITH EQ=0'S
                ;15543
U 1546, FF82,15 ;15544          INC LS[ERROR.NUMBER]            ;ERROR 2
U 1547, FDA2,15 ;15545          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1548, 5F4E,15 ;15546          MCOM LS[BIT7] TO WR[0]          ;DATA OF FFFFFF7F IN WR 0
U 1549, 3EA4,15 ;15547          MOV WR[0] TO LS[DATA.2]         ;FE(H) IN EXPECTED LOWER 8 BITS OF EXP
                ;15548                                          ; (LSB OF EXP IS BIT 7 OF RESULT)
U 154A, 0954,CC ;15549          JSR [LOOP.T21.1.2]              ;DO 'DEC EQ' WITH EQ=1'S
                ;15550
U 154B, 0956,64 ;15551          JMP [END.T21]                   ;DONE WITH DEC EQ TEST
                ;15552
                ;15553  LOOP.T21.1.2:
U 154C, 6588,15 ;15554          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 154D, FF88,15 ;15555          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 154E, 087D,CC ;15556          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 154F, 3618,15 ;15557          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EQ
U 1550, 3E1C,15 ;15558          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1551, 0878,AC ;15559          JSR [MOV.DATA]                  ;MOV EWR[0] TO EQ
U 1552, 3614,15 ;15560          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'DEC EQ' INSTRUCTION
U 1553, B716,95 ;15561          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 1554, 90F6,15 ;15562          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'DEC EQ'
U 1555, 10F6,95 ;15563          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 1556, B61A,15 ;15564          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EQ TO EWR[0]
U 1557, 3E1C,15 ;15565          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 1558, 0878,AC ;15566          JSR [MOV.DATA]                  ;MOV EQ TO EWR[0]
                ;15567
U 1559, 087B,5C ;15568          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15569                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 155A, B6A8,15 ;15570          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 155B, 36A4,95 ;15571          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 155C, 0869,3C ;15572          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 155D, 8954,C4 ;15573          JMP [LOOP.T21.1.2]              ;ERROR LOOP IF ENABLED
                ;15574
U 155E, FF88,15 ;15575          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 155F, 8877,4C ;15576          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15577                                          ; LOWER 8 BITS
U 1560, 087B,5C ;15578          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1561, B6A8,15 ;15579          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1562, B6A6,95 ;15580          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1563, 0869,3C ;15581          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1564, 8954,C4 ;15582          JMP [LOOP.T21.1.2]              ;ERROR LOOP IF ENABLED
U 1565, 5B00,14 ;15583          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;15584  END.T21:
```

```
;15585  .PAGE
;15586  .TOC    "TEST 22 Q-A FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)"
;15587  ;++
;15588  ;
;15589  ; Test Description:
;15590  ;
;15591  ;       This test checks the 'Q-A' function in the exponent data path using
;15592  ;       a SUB EWR[ET5] FROM EQ TO EWR[ET0] instruction.  The source is A,Q, the
;15593  ;       function is Q-A (EXP.CTL), and the destination is WRITE B.  The 2901
;15594  ;       SUB function was used previously with the instruction SUB.  This test
;15595  ;       checks that the EXP DECODE PAL is decoding the Q-A function correctly.
;15596  ;       The data path is to write EWR[0] and the upper bits of the FRAC path,
;15597  ;       shift this data into the 16 EXP bits, and either move the data into
;15598  ;       EWR[5] or into EQ.  After the Q-A instruction, the result is stored
;15599  ;       from EWR[0] and checked.  Data patterns are 0's and 1's.  Additional
;15600  ;       patterns are not necessary since the 2901 operation has been checked
;15601  ;       previously.
;15602  ;
;15603  ; Assumptions:
;15604  ;
;15605  ;       It is assumed that all previous tests have run successfully.
;15606  ;
;15607  ; Test Steps:
;15608  ;
;15609  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;15610  ;          EXP data path and mov EWR[0] to EWR[5].
;15611  ;       2) Load EWR[0] again with 0's, shift this data into the EXP data path,
;15612  ;          and mov EWR[0] to EQ.
;15613  ;       3) Execute SUB EWR[5] FROM EQ TO EWR[0] instruction.
;15614  ;       4) Check result in EWR[0] for 0's.
;15615  ;       5) Repeat steps 2 and 3 with data of 1's in EQ.  Check result for 1's.
;15616  ;       6) Repeat steps 1 through 3 with data of 1's in EWR[5].  Check result
;15617  ;          for a 1 in lsb of exponent.
;15618  ;       7) Repeat steps 2 and 3 with 1's in EWR[5] and check result for 0's.
;15619  ;
;15620  ; Error:
;15621  ;
;15622  ;       Note:  For the following errors, the data printed under OTHER in the
;15623  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15624  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15625  ;
;15626  ;       Error1 - SUB EWR[5] FROM EQ TO EWR[0] failed with 0's in EWR[5] and 0's
;15627  ;                EQ.
;15628  ;       Error2 - SUB EWR[5] FROM EQ TO EWR[0] failed with 0's in EWR[5] and 1's
;15629  ;                EQ.
;15630  ;       Error3 - SUB EWR[5] FROM EQ TO EWR[0] failed with 1's in EWR[5] and 0's
;15631  ;                EQ.
;15632  ;       Error4 - SUB EWR[5] FROM EQ TO EWR[0] failed with 1's in EWR[5] and 1's
;15633  ;                EQ.
;15634  ;
;15635  ; Debug:
;15636  ;
;15637  ;       Error1-4 - The most likely cause of this error is the EXP DECODE PAL.
;15638  ;       The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
;15639  ;       1110(B) respectively during the SUB instruction.  If these are incor-
```

```
                ;15640  ;       rect, they can be traced back to the control store.
                ;15641  ;         If these inputs are OK, check the output for a 00 1000 on I5 through
                ;15642  ;       I0 respectively and a high on CIN EXP0 H.  If these are incorrect, the
                ;15643  ;       PAL is probably bad.
                ;15644  ;
                ;15645  ;--
                ;15646
                ;15647  ;
                ;15648  ;       TEMPORARY LS LOCATIONS USED:
                ;15649  ;
                ;15650  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;15651  ; T9    MAIN MEMORY POINTER
                ;15652  ; T10   ADDRESS OF 'SUB EWR[ET5] FROM EQ TO EWR[ET0]'
                ;15653  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;15654  ; T12   ADDRESS OF 'MOV EWR[ET0] TO EQ'
                ;15655  ; T13   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET5]'
                ;15656  ; T14   ADDRESS OF MOV PUT HERE FOR MOV ROUTINE
                ;15657  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;15658  ;
                ;15659  T.22:
                ;15660
U 1566, B65C,15 ;15661          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;15662                                          ; STATUS WORD
U 1567, 3E80,15 ;15663          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;15664                                          ; BIT 15 INDICATES START OF TEST TO
                ;15665                                          ; CONSOLE
U 1568, 10E0,15 ;15666          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;15667  WAIT.T22.0:
U 1569, 0956,94 ;15668          JMP [WAIT.T22.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;15669                                          ; RESPOND
U 156A, 087E,6C ;15670          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;15671                                          ; AND SIZE BITS
U 156B, B670,15 ;15672          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;15673                                          ; OTHER DATA
U 156C, 3E80,15 ;15674          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 156D, B724,15 ;15675          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 156E, C75E,15 ;15676          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 156F, C54E,15 ;15677          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 1570, 3E8A,15 ;15678          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 1571, 37DA,15 ;15679          MOV LS[T22.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 1572, BE12,15 ;15680          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 1573, 37EA,15 ;15681          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;15682                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 1574, 3E10,15 ;15683          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 1575, 8870,3C ;15684          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'SUB' ROUTINE
U 1576, BE14,15 ;15685          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 1577, 37EC,15 ;15686          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15687                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 1578, 3E16,15 ;15688          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 1579, 8870,3C ;15689          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15690                                          ; TO EQ' INSTRUCTION
U 157A, BE18,15 ;15691          MOV WR[0] TO LS[T12]            ;STORE IN LS T12
U 157B, 8870,3C ;15692          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15693                                          ; TO EWR[5]' INSTRUCTION
U 157C, 3E1A,15 ;15694          MOV WR[0] TO LS[T13]            ;STORE IN LS T13
```

```
U 157D, B7EE,15 ;15695          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15696                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 157E, BE1E,15 ;15697          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
                ;15698
U 157F, E5A2,15 ;15699          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 1580, E5A4,15 ;15700          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 1581, 65A6,15 ;15701          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 1582, 087D,CC ;15702          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 1583, B61A,15 ;15703          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR0 TO EWR5
U 1584, 3E1C,15 ;15704          MOV WR[0] TO LS[T14]            ;PUT IN LS 14 FOR MOV ROUTINE
U 1585, 0878,AC ;15705          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[5]
U 1586, 0959,BC ;15706          JSR [SUB.T22.1.4]               ;DO 'SUB' OF 0 FROM 0
                ;15707
U 1587, FF82,15 ;15708          INC LS[ERROR.NUMBER]            ;ERROR 2
U 1588, FDA2,15 ;15709          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 1589, FDA4,15 ;15710          COM LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 158A, 7DA6,15 ;15711          COM LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 158B, 0959,BC ;15712          JSR [SUB.T22.1.4]               ;DO 'SUB' OF 0 FROM 1'S
                ;15713
U 158C, FF82,15 ;15714          INC LS[ERROR.NUMBER]            ;ERROR 3
U 158D, 087D,CC ;15715          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 158E, B61A,15 ;15716          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR0 TO EWR5
U 158F, 3E1C,15 ;15717          MOV WR[0] TO LS[T14]            ;PUT IN LS 14 FOR MOV ROUTINE
U 1590, 0878,AC ;15718          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[5]
U 1591, E5A2,15 ;15719          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 1592, 364E,15 ;15720          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 (WILL BE LSB OF RESULT)
U 1593, 3EA4,15 ;15721          MOV WR[0] TO LS[DATA.2]         ;EXPECTED OF 01 IN LOWER 8 BITS OF EXP
U 1594, 65A6,15 ;15722          CLR LS[DATA.3]                  ;EXPECTED OF 0'S IN UPPER 8 BITS OF EXP
U 1595, 0959,BC ;15723          JSR [SUB.T22.1.4]               ;DO 'SUB' OF 1'S FROM 0'S
                ;15724
U 1596, FF82,15 ;15725          INC LS[ERROR.NUMBER]            ;ERROR 4
U 1597, FDA2,15 ;15726          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 1598, E5A4,15 ;15727          CLR LS[DATA.2]                  ;EXPECTED OF 0'S IN LOWER 8 BITS
U 1599, 0959,BC ;15728          JSR [SUB.T22.1.4]               ;DO 'SUB' OF 1'S FROM 1'S
                ;15729
U 159A, 095B,24 ;15730          JMP [END.T22]                   ;DONE WITH 'SUB' TEST
                ;15731
                ;15732  SUB.T22.1.4:
U 159B, 6588,15 ;15733          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 159C, FF88,15 ;15734          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 159D, 087D,CC ;15735          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 159E, 3618,15 ;15736          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EQ
U 159F, 3E1C,15 ;15737          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 15A0, 0878,AC ;15738          JSR [MOV.DATA]                  ;MOV EWR[0] TO EQ
                ;15739  LOOP.T22.1.4:
U 15A1, 3614,15 ;15740          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 15A2, B716,95 ;15741          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 15A3, 90F6,15 ;15742          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'SUB EWR[5] FROM EQ TO EWR[0]'
U 15A4, 10F6,95 ;15743          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
                ;15744
U 15A5, 087B,5C ;15745          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15746                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 15A6, B6A8,15 ;15747          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 15A7, 36A4,95 ;15748          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 15A8, 0869,3C ;15749          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
```

```
U 15A9, 895A,14 ;15750          JMP [LOOP.T22.1.4]              ;ERROR LOOP IF ENABLED
                ;15751
U 15AA, FF88,15 ;15752          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 15AB, 8877,4C ;15753          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15754                                          ; LOWER 8 BITS
U 15AC, 087B,5C ;15755          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 15AD, B6A8,15 ;15756          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 15AE, B6A6,95 ;15757          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 15AF, 0869,3C ;15758          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 15B0, 895A,14 ;15759          JMP [LOOP.T22.1.4]              ;ERROR LOOP IF ENABLED
U 15B1, 5B00,14 ;15760          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;15761  END.T22:
```

```
;15762  .PAGE
;15763  .TOC    "TEST 23 A-Q FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)"
;15764  ;++
;15765  ;
;15766  ; Test Description:
;15767  ;
;15768  ;       This test checks the 'A-Q' function in the exponent data path using
;15769  ;       a SUB EQ FROM EWR[ET4] TO EWR[ET1] instruction.  The source is A.Q, the
;15770  ;       function is A-Q (EXP.CTL), and the destination is WRITE B.  The 2901
;15771  ;       SUB function was used previously with the instruction SUB.  This test
;15772  ;       checks that the EXP DECODE PAL is decoding the A-Q function correctly.
;15773  ;       The data path is to write EWR[0] and the upper bits of the FRAC path,
;15774  ;       shift this data into the 16 EXP bits, and either move the data into
;15775  ;       EWR[4] or into EQ.  After the A-Q instruction, the result is moved from
;15776  ;       EWR[ET1] to EQ, from EQ to EWR[ET0], and stored from EWR[0] to be
;15777  ;       checked.  Data patterns are 0's and 1's.  Additional patterns are not
;15778  ;       necessary since the 2901 operation has been checked previously.
;15779  ;
;15780  ; Assumptions:
;15781  ;
;15782  ;       It is assumed that all previous tests have run successfully.
;15783  ;
;15784  ; Test Steps:
;15785  ;
;15786  ;       1) Load EWR[0] and FWR[0] with 0's.  Shift this data into the 16 bit
;15787  ;          EXP data path and mov EWR[0] to EWR[4].
;15788  ;       2) Load EWR[0] again with 0's, shift this data into the EXP data path,
;15789  ;          and mov EWR[0] to EQ.
;15790  ;       3) Execute SUB EQ FROM EWR[4] TO EWR[1] instruction.  Mov result to EQ,
;15791  ;          then from EQ to EWR[0]
;15792  ;       4) Check result in EWR[0] for 0's.
;15793  ;       5) Repeat steps 2 and 3 with data of 1's in EQ.  Check result for a 1
;15794  ;          in LSB of exponent.
;15795  ;       6) Repeat steps 1 through 3 with data of 1's in EWR[4].  Check result
;15796  ;          for a 1's.
;15797  ;       7) Repeat steps 2 and 3 with 1's in EWR[4] and check result for 0's.
;15798  ;
;15799  ; Error:
;15800  ;
;15801  ;       Note:  For the following errors, the data printed under OTHER in the
;15802  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15803  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15804  ;
;15805  ;       Error1 - SUB EQ FROM EWR[4] TO EWR[1] failed with 0's in EWR[4] and 0's
;15806  ;                EQ.
;15807  ;       Error2 - SUB EQ FROM EWR[4] TO EWR[1] failed with 0's in EWR[4] and 1's
;15808  ;                EQ.
;15809  ;       Error3 - SUB EQ FROM EWR[4] TO EWR[1] failed with 1's in EWR[4] and 0's
;15810  ;                EQ.
;15811  ;       Error4 - SUB EQ FROM EWR[4] TO EWR[1] failed with 1's in EWR[4] and 1's
;15812  ;                EQ.
;15813  ;
;15814  ; Debug:
;15815  ;
;15816  ;       Error1-4 - The most likely cause of this error is the EXP DECODE PAL.
```

```
                ;15317  ;       The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
                ;15818  ;       0110(B) respectively during the SUB instruction.  If these are incor-
                ;15819  ;       rect, they can be traced back to the control store.
                ;15820  ;         If these inputs are OK, check the output for a 01 0000 on I5 through
                ;15821  ;       I0 respectively and a high on CIN EXP0 H.  If these are incorrect, the
                ;15822  ;       PAL is probably bad.
                ;15823  ;
                ;15824  ;--
                ;15825
                ;15826  ;
                ;15827  ;       TEMPORARY LS LOCATIONS USED:
                ;15828  ;
                ;15829  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
                ;15830  ; T9    MAIN MEMORY POINTER
                ;15831  ; T10   ADDRESS OF 'SUB EQ FROM EWR[ET4] TO EWR[ET1]'
                ;15832  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                ;15833  ; T12   ADDRESS OF 'MOV EWR[ET0] TO EQ'
                ;15834  ; T13   ADDRESS OF 'MOV EWR[ET0] TO EWR[ET4]'
                ;15835  ; T14   ADDRESS OF MOV PUT HERE FOR MOV ROUTINE
                ;15836  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                ;15837  ; T57   ADDRESS OF 'MOV EQ TO EWR[ET0]'
                ;15838  ; T58   ADDRESS OF 'MOV EWR[ET1] TO EQ'
                ;15839  ;
                ;15840  T.23:
                ;15841
U 15B2, B65E,15 ;15842          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;15843                                          ; STATUS WORD
U 15B3, 3E80,15 ;15844          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;15845                                          ; BIT 15 INDICATES START OF TEST TO
                ;15846                                          ; CONSOLE
U 15B4, 10E0,15 ;15847          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;15848  WAIT.T23.0:
U 15B5, 895B,54 ;15849          JMP [WAIT.T23.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;15850                                          ; RESPOND
U 15B6, 087E,6C ;15851          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;15852                                          ; AND SIZE BITS
U 15B7, B670,15 ;15853          MOV LS[OTHER.DATA] TO WR[0]     ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                ;15854                                          ; OTHER DATA
U 15B8, 3E80,15 ;15855          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 15B9, B724,15 ;15856          MOV LS[#FFFF00FF] TO WR[0]      ;BITS 8-15 = 0
U 15BA, C75E,15 ;15857          BIS LS[BIT15] TO WR[0]          ;SET BIT 15
U 15BB, C54E,15 ;15858          BIC LS[BIT7] TO WR[0]           ;CLEAR BIT 7
U 15BC, 3E8A,15 ;15859          MOV WR[0] TO LS[ERROR.MASK]     ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 15BD, 37DC,15 ;15860          MOV LS[T23.POINTER] TO WR[0]    ;GET POINTER TO MAIN MEMORY ADDRESS
U 15BE, BE12,15 ;15861          MOV WR[0] TO LS[T9]             ;SET UP POINTER IN LS
U 15BF, 37EA,15 ;15862          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                ;15863                                          ; EWR[ET0] AND EQ' INSTRUCTION
U 15C0, 3E10,15 ;15864          MOV WR[0] TO LS[T8]             ;STORE IN LS T8
U 15C1, 8870,3C ;15865          JSR [L0]                        ;LOAD WR 0 WITH ADDRESS OF 'SUB' ROUTINE
U 15C2, BE14,15 ;15866          MOV WR[0] TO LS[T10]            ;STORE IN LS T10
U 15C3, 37EC,15 ;15867          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15868                                          ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 15C4, 3E16,15 ;15869          MOV WR[0] TO LS[T11]            ;STORE IN LS T11
U 15C5, 8870,3C ;15870          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15871                                          ; TO EQ' INSTRUCTION
```

```
U 15C6, BE18,15 ;15872          MOV WR[0] TO LS[T12]             ;STORE IN LS T12
U 15C7, 8870,3C ;15873          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[0]
                ;15874                                          ; TO EWR[4]' INSTRUCTION
U 15C8, 3E1A,15 ;15875          MOV WR[0] TO LS[T13]            ;STORE IN LS T13
U 15C9, B7EE,15 ;15876          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                ;15877                                          ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 15CA, BE1E,15 ;15878          MOV WR[0] TO LS[T15]            ;STORE IN LS T15
U 15CB, 8870,3C ;15879          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EQ TO
                ;15880                                          ; EWR[0]' INSTRUCTION
U 15CC, 3EAE,15 ;15881          MOV WR[0] TO LS[T57]            ;STORE IN LS T57
U 15CD, 8870,3C ;15882          JSR [L0]                        ;LOAD WR[0] WITH ADDRESS OF 'MOV EWR[1]
                ;15883                                          ; TO EQ' INSTRUCTION
U 15CE, 3EB0,15 ;15884          MOV WR[0] TO LS[T58]            ;STORE IN LS T58
                ;15885
U 15CF, E5A2,15 ;15886          CLR LS[DATA.1]                  ;ZEROS IN FIRST FLOAT DATA
U 15D0, E5A4,15 ;15887          CLR LS[DATA.2]                  ;ZEROS IN EXPECTED LOWER 8 BITS OF EXP
U 15D1, 65A6,15 ;15888          CLR LS[DATA.3]                  ;ZEROS IN EXPECTED UPPER 8 BITS OF EXP
U 15D2, 087D,CC ;15889          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 15D3, B61A,15 ;15890          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR0 TO EWR4
U 15D4, 3E1C,15 ;15891          MOV WR[0] TO LS[T14]            ;PUT IN LS 14 FOR MOV ROUTINE
U 15D5, 0878,AC ;15892          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[4]
U 15D6, 095E,CC ;15893          JSR [SUB.T23.1.4]               ;DO 'SUB' OF 0 FROM 0
                ;15894
U 15D7, FF82,15 ;15895          INC LS[ERROR.NUMBER]            ;ERROR 2
U 15D8, FDA2,15 ;15896          COM LS[DATA.1]                  ;FIRST FLOAT DATA OF ONES
U 15D9, 364E,15 ;15897          MOV LS[BIT7] TO WR[0]           ;SET BIT 7 (WILL BE LSB OF RESULT)
U 15DA, 3EA4,15 ;15898          MOV WR[0] TO LS[DATA.2]         ;EXPECTED OF 01 IN LOWER 8 BITS OF EXP
U 15DB, 095E,CC ;15899          JSR [SUB.T23.1.4]               ;DO 'SUB' OF 1'S FROM 0
                ;15900
U 15DC, FF82,15 ;15901          INC LS[ERROR.NUMBER]            ;ERROR 3
U 15DD, 087D,CC ;15902          JSR [LOAD.SHIFT.EWR0]           ;LOAD 16 BITS OF EXPONENT IN EWR[0]
U 15DE, B61A,15 ;15903          MOV LS[T13] TO WR[0]            ;ADDRESS OF MOV EWR0 TO EWR4
U 15DF, 3E1C,15 ;15904          MOV WR[0] TO LS[T14]            ;PUT IN LS 14 FOR MOV ROUTINE
U 15E0, 0878,AC ;15905          JSR [MOV.DATA]                  ;MOV EWR[0] TO EWR[4]
U 15E1, E5A2,15 ;15906          CLR LS[DATA.1]                  ;DATA OF 0'S IN FIRST FLOAT
U 15E2, E5A4,15 ;15907          CLR LS[DATA.2]                  ;CLR DATA.2
U 15E3, FDA4,15 ;15908          COM LS[DATA.2]                  ;EXPECTED DATA IN LOWER 8 BITS OF EXP
U 15E4, 7DA6,15 ;15909          COM LS[DATA.3]                  ;EXPECTED DATA IN UPPER 8 BITS OF EXP
U 15E5, 095E,CC ;15910          JSR [SUB.T23.1.4]               ;DO 'SUB' OF 0 FROM 1'S
                ;15911
U 15E6, FF82,15 ;15912          INC LS[ERROR.NUMBER]            ;ERROR 4
U 15E7, FDA2,15 ;15913          COM LS[DATA.1]                  ;DATA OF 1'S IN FIRST FLOAT
U 15E8, E5A4,15 ;15914          CLR LS[DATA.2]                  ;EXPECTED OF 0'S IN LOWER 8 BITS
U 15E9, 65A6,15 ;15915          CLR LS[DATA.3]                  ;EXPECTED OF 0'S IN UPPER 8 BITS
U 15EA, 095E,CC ;15916          JSR [SUB.T23.1.4]               ;DO 'SUB' OF 1'S FROM 1'S
                ;15917
U 15EB, 0960,94 ;15918          JMP [END.T23]                   ;DONE WITH 'SUB' TEST
                ;15919
                ;15920  SUB.T23.1.4:
U 15EC, 6588,15 ;15921          CLR LS[ADDRESS.DATA]            ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 15ED, FF88,15 ;15922          INC LS[ADDRESS.DATA]            ;1 INDICATES LOWER 8 BITS FAILED
U 15EE, 087D,CC ;15923          JSR [LOAD.SHIFT.EWR0]           ;GO LOAD EXPONENT INTO EWR[ET0]
U 15EF, 3618,15 ;15924          MOV LS[T12] TO WR[0]            ;ADDRESS OF MOV EWR[0] TO EQ
U 15F0, 3E1C,15 ;15925          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 15F1, 0878,AC ;15926          JSR [MOV.DATA]                  ;MOV EWR[0] TO EQ
```

```
U 15F2, 3614,15 ;15927          MOV LS[T10] TO WR[0]            ;GET ADDRESS OF FPA 'SUB' INSTRUCTION
U 15F3, B716,95 ;15928          MOV LS[#300] TO WR[1]           ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 15F4, 90F6,15 ;15929          MISC2 [TRAP.ACC] WR[0]          ;FORCE 'SUB EQ FROM EWR[ET4] TO EWR[1]'
U 15F5, 10F6,95 ;15930          MISC2 [TRAP.ACC] WR[1]          ;FORCE LOOP SELF
U 15F6, B6B0,15 ;15931          MOV LS[58] TO WR[0]             ;ADDRESS OF MOV EWR[1] TO EQ
U 15F7, 3E1C,15 ;15932          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 15F8, 0878,AC ;15933          JSR [MOV.DATA]                  ;MOV EWR[1] TO EQ
U 15F9, B6AE,15 ;15934          MOV LS[T57] TO WR[0]            ;ADDRESS OF MOV EQ TO EWR[0]
U 15FA, 3E1C,15 ;15935          MOV WR[0] TO LS[T14]            ;SET UP FOR MOV SUBROUTINE
U 15FB, 0878,AC ;15936          JSR [MOV.DATA]                  ;MOV EQ TO EWR[0]
                ;15937
U 15FC, 087B,5C ;15938          JSR [STORE.0.TRIPLE]            ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;15939                                          ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 15FD, B6A8,15 ;15940          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 15FE, 36A4,95 ;15941          MOV LS[DATA.2] TO WR[1]         ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 15FF, 0869,3C ;15942          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1600, 895E,C4 ;15943          JMP [SUB.T23.1.4]               ;ERROR LOOP IF ENABLED
                ;15944
U 1601, FF88,15 ;15945          INC LS[ADDRESS.DATA]            ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1602, 8877,4C ;15946          JSR [SHIFT.RIGHT.8]             ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;15947                                          ; LOWER 8 BITS
U 1603, 087B,5C ;15948          JSR [STORE.0.TRIPLE]            ;GET RESULT AGAIN
U 1604, B6A8,15 ;15949          MOV LS[FIRST.FLT] TO WR[0]      ;PUT FIRST FLT RESULT IN WR 0
U 1605, B6A6,95 ;15950          MOV LS[DATA.3] TO WR[1]         ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1606, 0869,3C ;15951          JSR [CHECK.RESULT]              ;CHECK FOR ERRORS
U 1607, 895E,C4 ;15952          JMP [SUB.T23.1.4]               ;ERROR LOOP IF ENABLED
U 1608, 5B00,14 ;15953          RETURN                          ;DONE WITH THIS DATA PATTERN
                ;15954  END.T23:
```

```
;15955  .PAGE
;15956  .TOC    "TEST 24 CLR FUNCTION ON EXP DECODE PAL TEST(M8389 FPA MODULE)"
;15957  ;++
;15958  ;
;15959  ; Test Description:
;15960  ;
;15961  ;       This test checks the 'CLR' function in the EXP DECODE PAL using a
;15962  ;       CLR EWR[ET0] instruction.  The source is 0.A, the function is R AND S,
;15963  ;       and the destination is WRITE B. This test checks that the EXP DECODE
;15964  ;       PAL will set the 2901 control lines properly during a CLR EWR instruc-
;15965  ;       tion.  The data path is to write EWR[0] and the upper bits of the FRAC
;15966  ;       path and shift this data into the 16 EXP bits.  After the CLR EWR[ET0]
;15967  ;       instruction, the result is checked for a zero.  Data patterns are 0's
;15968  ;       and 1's.
;15969  ;
;15970  ; Assumptions:
;15971  ;
;15972  ;       It is assumed that all previous tests have run successfully.
;15973  ;
;15974  ; Test Steps:
;15975  ;
;15976  ;       1) Load EWR[0] and FWR[0] with 1's.  Shift this data into the 16 bit
;15977  ;          EXP data path.
;15978  ;       2) Execute CLR EWR[ET0] instruction.
;15979  ;       3) Check result in EWR[0] for all 0's
;15980  ;       4) Repeat steps 1 and 2 with data of 0's in EWR[0].  Check result for
;15981  ;          0's
;15982  ;
;15983  ; Error:
;15984  ;
;15985  ;       Note:  For the following errors, the data printed under OTHER in the
;15986  ;       error report indicates whether the lower 8 bits of exponent (OTHER=1)
;15987  ;       or the upper 8 bits of exponent (OTHER=2) failed.
;15988  ;
;15989  ;       Error1 - CLR EWR[ET0] with EWR[0]=1'S failed.
;15990  ;       Error1 - CLR EWR[ET0] with EWR[0]=0'S failed.
;15991  ;
;15992  ; Debug:
;15993  ;
;15994  ;       Errors 1 and 2 - The most likely cause of this error is the EXP DECODE
;15995  ;       PAL.  The inputs on EXP CODE3 (1) H through EXP CODE0 (1) H should be
;15996  ;       1100(B) respectively during the CLR EWR instruction.  If these are in-
;15997  ;       correct, they can be traced back to the control store.
;15998  ;         If these inputs are OK, check the output for a 10 0100 on I5 through
;15999  ;       I0 respectively during the CLR EWR instruction.  If these are incor-
;16000  ;       rect, the PAL is bad.
;16001  ;
;16002  ;--
;16003
;16004  ;
;16005  ;       TEMPORARY LS LOCATIONS USED:
;16006  ;
;16007  ; T8    ADDRESS OF 'SHF RIGHT EWR[ET0] AND EQ'
;16008  ; T9    MAIN MEMORY POINTER
;16009  ; T10   ADDRESS OF 'CLR EWR[ET0]'
```

```
                 ;16010  ; T11   ADDRESS OF 'SHF LEFT EWR[ET0] AND EQ IN[FRAC]'
                 ;16011  ; T15   ADDRESS OF 'SHF LEFT FWR[FT0] AND FQ IN[DIV.SHF]
                 ;16012  ;
                 ;16013  T.24:
                 ;16014
U 1609, B65E,15  ;16015          MOV LS[BEGIN.TEST] TO WR[0]      ;SET BIT 15 IN WR[0] FOR CONTROL AND
                 ;16016                                           ; STATUS WORD
U 160A, 3E80,15  ;16017          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                 ;16018                                           ; BIT 15 INDICATES START OF TEST TO
                 ;16019                                           ; CONSOLE
U 160B, 10E0,15  ;16020          MISC [SET.CP.ATTN]               ;ISSUE CPU ATTN TO CONSOLE
                 ;16021  WAIT.T24.0:
U 160C, 0960,C4  ;16022          JMP [WAIT.T24.0]                 ;LOOP HERE WAITING FOR CONSOLE TO
                 ;16023                                           ; RESPOND
U 160D, 087E,6C  ;16024          JSR [SETUP]                      ;SET UP ERROR INFO AND CLEAR INSTRUC
                 ;16025                                           ; AND SIZE BITS
U 160E, B670,15  ;16026          MOV LS[OTHER.DATA] TO WR[0]      ;SET BIT TO MAKE ERROR REPORT PRINT UNDER
                 ;16027                                           ; OTHER DATA
U 160F, 3E80,15  ;16028          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT IN CONTROL WORD
U 1610, B724,15  ;16029          MOV LS[#FFFF00FF] TO WR[0]       ;BITS 8-15 = 0
U 1611, C75E,15  ;16030          BIS LS[BIT15] TO WR[0]           ;SET BIT 15
U 1612, C54E,15  ;16031          BIC LS[BIT7] TO WR[0]            ;CLEAR BIT 7
U 1613, 3E8A,15  ;16032          MOV WR[0] TO LS[ERROR.MASK]      ;SET UP MASK TO CHECK BITS 7-14 ONLY
U 1614, B7DE,15  ;16033          MOV LS[T24.POINTER] TO WR[0]     ;GET POINTER TO MAIN MEMORY ADDRESS
U 1615, BE12,15  ;16034          MOV WR[0] TO LS[T9]              ;SET UP POINTER IN LS
U 1616, 37EA,15  ;16035          MOV LS[SHIFT.RIGHT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF RIGHT
                 ;16036                                           ; EWR[ET0] AND EQ' INSTRUCTION
U 1617, 3E10,15  ;16037          MOV WR[0] TO LS[T8]              ;STORE IN LS T8
U 1618, 8870,3C  ;16038          JSR [LO]                         ;LOAD WR 0 WITH ADDRESS OF 'CLR EWR[ET0]'
                 ;16039                                           ; ROUTINE
U 1619, BE14,15  ;16040          MOV WR[0] TO LS[T10]             ;STORE IN LS T10
U 161A, 37EC,15  ;16041          MOV LS[SHIFT.LEFT.EWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                 ;16042                                           ; EWR[ET0] AND EQ IN[FRAC]' INSTRUCTION
U 161B, 3E16,15  ;16043          MOV WR[0] TO LS[T11]             ;STORE IN LS T11
U 161C, B7EE,15  ;16044          MOV LS[SHIFT.LEFT.FWR] TO WR[0] ;LOAD WR 0 WITH ADDRESS OF 'SHF LEFT
                 ;16045                                           ; FWR[FT0] AND FQ IN[DIV.SHF]' INSTRUCTION
U 161D, BE1E,15  ;16046          MOV WR[0] TO LS[T15]             ;STORE IN LS T15
                 ;16047
U 161E, E5A2,15  ;16048          CLR LS[DATA.1]                   ;ZEROS IN LS
U 161F, FDA2,15  ;16049          COM LS[DATA.1]                   ;ONES IN FIRST FLOAT DATA
U 1620, E5A4,15  ;16050          CLR LS[DATA.2]                   ;ZEROS IN EXPECTED DATA FOR LOWER 8 BITS
U 1621, 65A6,15  ;16051          CLR LS[DATA.3]                   ;ZEROS IN EXPECTED DATA FOR UPPER 8 BITS
U 1622, 8962,7C  ;16052          JSR [LOOP.T24.1.2]               ;DO 'CLR EWR[0]' WITH EWR=1'S
                 ;16053
U 1623, FF82,15  ;16054          INC LS[ERROR.NUMBER]             ;ERROR 2
U 1624, E5A2,15  ;16055          CLR LS[DATA.1]                   ;FIRST FLOAT DATA OF ZEROS
U 1625, 8962,7C  ;16056          JSR [LOOP.T24.1.2]               ;DO 'CLR EWR[0]' WITH EQ=0'S
                 ;16057
U 1626, 8963,B4  ;16058          JMP [END.T24]                    ;DONE WITH CLR EWR[0] TEST
                 ;16059
                 ;16060  LOOP.T24.1.2:
U 1627, 6588,15  ;16061          CLR LS[ADDRESS.DATA]             ;DATA TO PRINT UNDER OTHER IN ERROR REPORT
U 1628, FF88,15  ;16062          INC LS[ADDRESS.DATA]             ;1 INDICATES LOWER 8 BITS FAILED
U 1629, 087D,CC  ;16063          JSR [LOAD.SHIFT.EWR0]            ;GO LOAD EXPONENT INTO EWR[ET0]
U 162A, 3614,15  ;16064          MOV LS[T10] TO WR[0]             ;GET ADDRESS OF FPA 'CLR EWR[0]'
```

```
                ;16065                                  ; INSTRUCTION
U 162B, B716,95 ;16066          MOV LS[#300] TO WR[1]   ;SECOND FPA ADDRESS TO FORCE (LOOP SELF)
U 162C, 90F6,15 ;16067          MISC2 [TRAP.ACC] WR[0]  ;FORCE 'CLR EWR[0]'
U 162D, 10F6,95 ;16068          MISC2 [TRAP.ACC] WR[1]  ;FORCE LOOP SELF
                ;16069
U 162E, 087B,5C ;16070          JSR [STORE.0.TRIPLE]    ;GET RESULT IN LS FIRST.FLT, SEC.FLT,
                ;16071                                  ; THIRD.FLT AND EXPONENT (BITS 7-14)
U 162F, B6A8,15 ;16072          MOV LS[FIRST.FLT] TO WR[0] ;PUT FIRST FLT RESULT IN WR 0
U 1630, 36A4,95 ;16073          MOV LS[DATA.2] TO WR[1] ;EXPECTED DATA FOR LOWER 8 BITS OF EXP
U 1631, 0869,3C ;16074          JSR [CHECK.RESULT]      ;CHECK FOR ERRORS
U 1632, 0962,74 ;16075          JMP [LOOP.T24.1.2]      ;ERROR LOOP IF ENABLED
                ;16076
U 1633, FF88,15 ;16077          INC LS[ADDRESS.DATA]    ;INC PRINTOUT UNDER OTHER DATA TO 2
U 1634, 8877,4C ;16078          JSR [SHIFT.RIGHT.8]     ;SHIFT UPPER 8 BITS OF EXPONENT INTO
                ;16079                                  ; LOWER 8 BITS
U 1635, 087B,5C ;16080          JSR [STORE.0.TRIPLE]    ;GET RESULT AGAIN
U 1636, B6A8,15 ;16081          MOV LS[FIRST.FLT] TO WR[0] ;PUT FIRST FLT RESULT IN WR 0
U 1637, B6A6,95 ;16082          MOV LS[DATA.3] TO WR[1] ;EXPECTED DATA FOR UPPER 8 BITS OF EXP
U 1638, 0869,3C ;16083          JSR [CHECK.RESULT]      ;CHECK FOR ERRORS
U 1639, 0962,74 ;16084          JMP [LOOP.T24.1.2]      ;ERROR LOOP IF ENABLED
U 163A, 5B00,14 ;16085          RETURN                  ;DONE WITH THIS DATA PATTERN
                ;16086  END.T24:
```

```
;16087  .page
;16088  .toc    "TEST 25 96 BIT FLOATING DATA TEST(M8389 FPA MODULE)"
;16089  ;++
;16090  ;
;16091  ;
;16092  ; Test Description:
;16093  ;
;16094  ;       The purpose of this test is to check that the three floats of data
;16095  ;       are going to the correct 2901's and only those 2901's. This will be
;16096  ;       verified by loading the three floats of data with a shifting one
;16097  ;       pattern through the three floats of data one at a time and reading
;16098  ;       all three floats of data back to the CPU to verify that the right
;16099  ;       float of data is being written and only that float.
;16100  ;
;16101  ; Assumptions:
;16102  ;
;16103  ;       If is assumed that the working registers have been tested previously.
;16104  ;
;16105  ; Test Steps:
;16106  ;
;16107  ;       1) Get the data pattern to be loaded.
;16108  ;       2) Load the data into the FPA.
;16109  ;       3) Read the data back from the FPA to the CPU.
;16110  ;       4) Check all three floats of data for error.
;16111  ;       5) If the last pattern has been tested then go on to the next test
;16112  ;          else get the next pattern and repeat the prvious steps.
;16113  ;
;16114  ; Error:
;16115  ;
;16116  ;       NOTE: When other data is one bits 32 - 56 of the fraction data path and
;16117  ;             bits 0 - 7 of the exponent data path. When other data is 2 then
;16118  ;             the data is bit 0 - 31 of the fraction data path. When other data
;16119  ;             is 3 then the data is bits 0 - 7 of the extension data path.
;16120  ;       Error1 - Data failed in the first float
;16121  ;       Error2 - Data failed in the second float
;16122  ;       Error3 - Data failed in the third float
;16123  ;
;16124  ; Debug:
;16125  ;
;16126  ;       Error1: If this error occurs the destination should be checked. On the
;16127  ;               2901's that should be written there should be a write back and
;16128  ;               on the 2901's that aren't being written there shouldn't be.
;16129  ;               If there isn't a write back on the destination lines then the
;16130  ;               destination bits are in error and the DESTINATION PROM should
;16131  ;               be checked for error. If the destination lines are not in error
;16132  ;               then it's likely that one of the other floats is being written
;16133  ;               into the 2901's when it shouldn't be. The easiest way to deter-
;16134  ;               mine if this is the case is to look at the excpected data for
;16135  ;               the other two floats and if the received data matches then it's
;16136  ;               likely that the float with the matching data is overwriting the
;16137  ;               data in the first float.
;16138  ;       Error2: Same as error 1.
;16139  ;       Error3: Same as error 1.
;16140  ;
;16141  ;
```

```
                ;16142  ;--
                ;16143  T.25:
U 163B, B65E,15 ;16144          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;16145                                          ; STATUS WORD
U 163C, 3E80,15 ;16146          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;16147                                          ; BIT 15 INDICATES START OF TEST TO
                ;16148                                          ; CONSOLE
U 163D, 10E0,15 ;16149          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;16150  WAIT.T25.0:
U 163E, 8963,E4 ;16151          JMP [WAIT.T25.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;16152                                          ; RESPOND
U 163F, 087E,6C ;16153          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;16154                                          ; AND SIZE BITS
U 1640, B670,15 ;16155          MOV LS[OTHER.DATA] TO WR[0]     ;Setup other data
U 1641, 3E80,15 ;16156          MOV WR[0] TO LS[CONTROL.STATUS]
U 1642, 365F,15 ;16157          MOV LS[BIT15] TO WR[2]          ;Setup for first float error mask
U 1643, B725,95 ;16158          MOV LS[#FFFF00FF] TO WR[3]
U 1644, E5F8,15 ;16159          CLR LS[OS]                      ;Clear the index into LS
U 1645, 36F6,15 ;16160  FF:     MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;Index into LS to get data
U 1646, 3EA2,15 ;16161          MOV WR[0] TO LS[DATA.1]         ;Put data in DATA.1
U 1647, E5A4,15 ;16162          CLR LS[DATA.2]                  ;Put zeros in the other floats
U 1648, 65A6,15 ;16163          CLR LS[DATA.3]
U 1649, 0878,FC ;16164          JSR [LOAD.TRIPLE]               ;Load data into WR[0]
U 164A, 087B,5C ;16165          JSR [STORE.0.TRIPLE]            ;Store data to the CPU
U 164B, 887C,6C ;16166          JSR [CHECK.SUB]                 ;Check for error
U 164C, 8964,54 ;16167          JMP [FF]                        ;Loop on error
U 164D, 7FF8,15 ;16168          INC LS[OS]                      ;Add one to the index
U 164E, B6F8,15 ;16169          MOV LS[OS] TO WR[0]
                ;16170          CMP WR[0] WITH LS[#20],         ;If not 32 go on to next float
U 164F, CF4A,35 ;16171           DT(LONG)&SET.ALU.CC
U 1650, 8964,51 ;16172          JMP.IF[NEQ] TO [FF]             ;else loop to FF
U 1651, E5F8,15 ;16173          CLR LS[OS]
U 1652, FF82,15 ;16174          INC LS[ERROR.NUMBER]            ;Go onto check second float
U 1653, 36F6,15 ;16175  SF:     MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;Index into LS to get data
U 1654, 3EA4,15 ;16176          MOV WR[0] TO LS[DATA.2]         ;Put data in DATA.2
U 1655, E5A2,15 ;16177          CLR LS[DATA.1]                  ;Put zeros in the other floats
U 1656, 65A6,15 ;16178          CLR LS[DATA.3]
U 1657, 0878,FC ;16179          JSR [LOAD.TRIPLE]               ;Load data into WR[0]
U 1658, 087B,5C ;16180          JSR [STORE.0.TRIPLE]            ;Store data to the CPU
U 1659, 887C,6C ;16181          JSR [CHECK.SUB]                 ;Check for error
U 165A, 0965,34 ;16182          JMP [SF]                        ;Loop on error
U 165B, 7FF8,15 ;16183          INC LS[OS]                      ;Add one to the index
U 165C, B6F8,15 ;16184          MOV LS[OS] TO WR[0]
                ;16185          CMP WR[0] WITH LS[#20],         ;If not 32 go on to next float
U 165D, CF4A,35 ;16186           DT(LONG)&SET.ALU.CC
U 165E, 0965,31 ;16187          JMP.IF[NEQ] TO [SF]             ;else loop to SF
U 165F, E5F8,15 ;16188          CLR LS[OS]
U 1660, FF82,15 ;16189          INC LS[ERROR.NUMBER]            ;Go onto check third float
U 1661, 36F6,15 ;16190  TF:     MOV LS[SHIFT.OS(4-0)] TO WR[0]  ;Index into LS to get data
U 1662, BEA6,15 ;16191          MOV WR[0] TO LS[DATA.3]         ;Put data in DATA.3
U 1663, E5A4,15 ;16192          CLR LS[DATA.2]                  ;Put zeros in the other floats
U 1664, E5A2,15 ;16193          CLR LS[DATA.1]
U 1665, 0878,FC ;16194          JSR [LOAD.TRIPLE]               ;Load data into WR[0]
U 1666, 087B,5C ;16195          JSR [STORE.0.TRIPLE]            ;Store data to the CPU
U 1667, 887C,6C ;16196          JSR [CHECK.SUB]                 ;Check for error
```

```
U 1668, 8966,14 ;16197          JMP [TF]                        ;Loop on error
U 1669, 7FF8,15 ;16198          INC LS[OS]                      ;Add one to the index
U 166A, B6F8,15 ;16199          MOV LS[OS] TO WR[0]
                ;16200          CMP WR[0] WITH LS[#20],         ;If not 32 go on to next float
U 166B, CF4A,35 ;16201           DT(LONG)&SET.ALU.CC
U 166C, 8966,11 ;16202          JMP.IF[NEQ] TO [TF]             ;else loop to TF
                ;16203  T25.END:
```

```
;16204  .PAGE
;16205  .TOC    "TEST 26 SHIFT FIELD TEST(M8389 FPA MODULE)"
;16206  ;++
;16207  ;
;16208  ;
;16209  ; Test Description:
;16210  ;
;16211  ;       This test check the what is shifted into the most significant bit
;16212  ;       (right shift) of the fraction data path and the least significant
;16213  ;       bit (left shift) of both the fraction data path an the exponent data
;16214  ;       path. Zeros are always shifted in from the right to the MSB of the
;16215  ;       exponent data path. The cases for right shift into the fraction data
;16216  ;       path will be tested first. Set the clock field to ALTER FRACTION
;16217  ;       and the shift field to 01, then shift right two times. If bits 55 and
;16218  ;       54 of the Q and WR selected by the address lines are both one then
;16219  ;       there is no error. This method of testing will continue until all of
;16220  ;       the possibilities have been exhausted. Then the same approach will
;16221  ;       be taken with the left shifts. Since bit 55 of the fraction data path
;16222  ;       will only be tranfered back to the CPU on a 3rd huge store the data
;16223  ;       will be read back to the CPU when the shift field is 11. The table
;16224  ;       below contains the what is shifted right and left under the varying
;16225  ;       conditions of the clock field and the shift field. The numbers in the
;16226  ;       right hand corner correspond to the test step # and the error # where
;16227  ;       the condition is being tested.
;16228  ;
;16229  ;
;16230  ;       Right Shift
;16231  ;
;16232  ;               Clock Field = 010
;16233  ;
;16234  ; Shifted into:         FRAC55 Q3                       FRAC55 R3
;16235  ;
;16236  ; From:
;16237  ;   Shift field                                                        Test #
;16238  ;
;16239  ;       00              EXPONENT Q0 *                   EXPONENT R0     6
;16240  ;       01              EXTENSION R0                    FRAC COUT       7
;16241  ;       10              0                               EXT00 R0 SAVE   5
;16242  ;       11              EXTENSION R0                    0               4
;16243  ;
;16244  ;               Clock field = 011
;16245  ;
;16246  ;       00              EXTENSION R0                    EXPONENT R0     3
;16247  ;       11              1                               1               1
;16248  ;       10              0 *                             EXT00 R0 SAVE * *
;16249  ;       01              0                               0               2
;16250  ;
;16251  ;
;16252  ;       Left Shift
;16253  ;
;16254  ;               Exponent                                        Fraction
;16255  ;
;16256  ;  Shift #      Q0              R0                      Q0              R0      T#
;16257  ;
;16258  ;   00          FRAC55 Q3       FRAC55 R3               0               0       10
```

```
;16259  ;   01          0 *             0 *             0 *             FRAC55 R3 SAVE          9
;16260  ;   10          1               1               1               1                       8
;16261  ;   11          FRAC55 Q3       FRAC55 R3       QIN             FRAC55 Q3               11
;16262  ;
;16263  ; * - Can't be checked because the word is not used in the FPA microcode.
;16264  ;
;16265  ;
;16266  ; Assumptions:
;16267  ;
;16268  ;       It is assumed that all of the control lines on the 2901 have been
;16269  ;       tested.
;16270  ;
;16271  ; Test Steps:
;16272  ;
;16273  ;       1) Load all registers with zeros. Issue a MISC instruction that will
;16274  ;          OR a WR with zero and shift right with the clock fiald set to alter
;16275  ;          fraction and the shift field set to 11. If the WR and the Q reg
;16276  ;          don't have a one in the MSB then issue error1 for FWR and error2 for
;16277  ;          FQ.
;16278  ;       2) Load all registers with ones. The clock field should be alter frac-
;16279  ;          tion and the shift field should be 11. Issue a MISC instruction that
;16280  ;          will OR a WR with zero and shift right. The clock field should be
;16281  ;          set to alter fraction and the shift field should be set to 01. If
;16282  ;          the WR and the Q reg don't have a zero in the MSB then issue error3
;16283  ;          for FWR and error 4 for FQ.
;16284  ;       3) Load zeros into all registers with the exception of the two LSB's
;16285  ;          bits of the extension data path and the exponent data path of the
;16286  ;          WR which should be loaded with 10. The clock field should be alter
;16287  ;          fraction and the shift field should be 00. Issue two MISC instruc-
;16288  ;          tions to OR the WR with 0 and shift right twice. If the two MSB's of
;16289  ;          Q and WR fraction data path aren't 10 then issue error 5 for FWR and
;16290  ;          error 6 for FQ.
;16291  ;       4) Load ones into all registers with the exception of the two LSB's
;16292  ;          of the WR extension data path which should be loaded with 01. The
;16293  ;          clock field should be a nop and the shift field should be 11. Issue
;16294  ;          two MISC instructions to OR the WR with zero and shift the WR right
;16295  ;          twice. If the MSB's of the Q register isn't 01 and the MSB's of the
;16296  ;          WR of the fraction data path aren't 00 then issue error 7 for FWR
;16297  ;          and error 8 for FQ.
;16298  ;       5) Load ones into all the registers with the exception of the two LSB's
;16299  ;          of the WR of the extension data path which should be 01. The clock
;16300  ;          field should be nop and the shift field should be 10. Issue three
;16301  ;          MISC isntructions to OR the WR with zero and shift right three times
;16302  ;          The three MSB's of the Q register of the fraction data path should
;16303  ;          be 0 and the two MSB's of the WR of the fraction data path should
;16304  ;          be 01, if not issue error 9 for FWR and error A for FQ.
;16305  ;       6) Load zeros into all of the registers except the two LSB's of the
;16306  ;          exponent data path for both the Q and the WR which should be 10.
;16307  ;          The clock field should be nop and the shift field should be 00.
;16308  ;          Issue two MISC isntructions to OR the with 0 and shift right twice.
;16309  ;          If the MSB's of the Q reg and the WR aren't 10 then issue error B.
;16310  ;       7) Load two WR's and the Q register with zero's except for the LSB's
;16311  ;          of the extension data path of one WR which should be 10. As a result
;16312  ;          of the hidden bit, bit 55 of the fraction data path will be set.
;16313  ;          This means that if an add of the two WR's is executed FRAC COUT
```

```
;16314  ;          will be set for the following cycle. The result of the add is one
;16315  ;          WR with zeros in all bits except the LSB's of the extension data
;16316  ;          path which are 10. An OR of the resultant WR with zero and a shift
;16317  ;          right in the destination should result in the bit 55 of the QR being
;16318  ;          zero and bit 55 of the WR being 1. A second shift right should
;16319  ;          result 1n FRAC55 Q3 = 1 and FRAC55 R3 = 0. If either of these are in
;16320  ;          then issue error C for FWR and error D for FQ.
;16321  ;       9) Load a WR and the QR with zeros. The clock field should be a nop
;16322  ;          and the shift field should be 10. Issue a MISC instruction to OR
;16323  ;          the WR with 0 and shift left. If the LSB of the Q register and the
;16324  ;          WR of both the fraction and exponent data path are 1 then there's no
;16325  ;          error, else issue error E if FWR fails, F if FQ fails, 10 if EWR
;16326  ;          fails, and 11 if EQ fails.
;16327  ;      10) Load all registers with ones except 10 in the MSB's of the fraction
;16328  ;          data path. The clock field should be a nop and the shift field
;16329  ;          should be 01. Issue a MISC instruction to OR the WR with 0 and shift
;16330  ;          left three times. If the three LSB's of Q register of both the frac-
;16331  ;          tion and exponent data paths and the WR of the exponent data path
;16332  ;          are zero and the two LSB's of the WR of the fraction data path are
;16333  ;          10 then there's no error, else issue error 12 if FWR fails.
;16334  ;      11) Load all the registers with ones except the MSB's of the Q register
;16335  ;          and the WR of the exponent data path which should be 01. The clock
;16336  ;          filed should be a nop and the shift field should be 00. Issue two
;16337  ;          MISC instructions to OR the WR with 0 and shift left twice. If the
;16338  ;          LSB's of the registers in the exponent data path aren't 01 and the
;16339  ;          LSB's of the registers in the fraction data path aren't 00 then
;16340  ;          issue error 13 if FWR or EWR fails and error 14 if FQ or EQ fails.
;16341  ;      12) Load all the registers with 0's except the MSB's of the exponent
;16342  ;          data paths and the WR of the fraction data path which should be 01.
;16343  ;          The instruction encode bits should be anything but DIV or DIVL which
;16344  ;          corresponds to 00101 or 11010. The clock field should be nop and the
;16345  ;          shift field should be 11. Issue a MISC instruction to OR the WR with
;16346  ;          zero and shift left. The LSB of all the data paths except the QR of
;16347  ;          the fraction data path should be should be 1. A second shift left
;16348  ;          should cause all of the LSB's of the data paths to be 0. This shift
;16349  ;          test does not test the case when a DIV ro DIV L is being executed.
;16350  ;          These two cases for the five different data types will be tested in
;16351  ;          specific test to check the MUL/DIV logic. If FWR or EWR fails then
;16352  ;          issue error 15 and if FQ or EQ fails then issue error 16.
;16353  ;
;16354  ; Error:
;16355  ;
;16356  ;          NOTE: If other data is 1 then bits 31 - 56 of the fraction data path
;16357  ;                and bits 0 - 7 of the exponent data path are being checked. If
;16358  ;                other data is 2 then bits then bits 0 - 31 of the fraction data
;16359  ;                path are being checked for error. If other data is 3 then bits
;16360  ;                0 - 7 of the extension data path are being checked for error.
;16361  ;          Error1 - A one failed to be shifted in from the right of FWR.
;16362  ;          Error2 - A one failed to be shifted in from the right of FQ.
;16363  ;          Error3 - A zero failed to be shifted in from the right of FWR.
;16364  ;          Error4 - A zero failed to be shifted in from the right of FQ.
;16365  ;          Error5 - The LSB of the extension data path failed to be shifted into
;16366  ;                   the MSB of the QR or the LSB of the exponent data path failed
;16367  ;                   to be shifted into the MSB of the WR.
;16368  ;          Error6 - The LSB of the extension data path failed to be shifted into
```

```
;16369  ;                the MSB of the QR or the LSB of the exponent data path failed
;16370  ;                to be shifted into the MSB of the QR.
;16371  ;       Error7 - The LSB of the WR of the extension data path failed to be
;16372  ;                shifted into the MSB of the QR in the fraction data path or
;16373  ;                0's failed to be shifted into the WR of the fraction data path
;16374  ;       Error8 - The LSB of the WR of the extension data path failed to be
;16375  ;                shifted into the MSB of the QR in the fraction data path or
;16376  ;                0's failed to be shifted into the QR of the fraction data path
;16377  ;       Error9 - EXTOO R0 SAVE failed to be shifted into FRAC55 R3 or 0 failed
;16378  ;                to be shifted into FRAC55 Q3.
;16379  ;       ErrorA - EXPONENT R0 failed to be shifted into FRAC55 Q3 or EXPONENT R0
;16380  ;                failed to be shifted into FRAC55 R3 of the WR.
;16381  ;       ErrorB - EXPONENT R0 failed to be shifted into FRAC55 Q3 or EXPONENT R0
;16382  ;                failed to be shifted into FRAC55 R3 of the QR.
;16383  ;       ErrorC - FRAC COUT failed to be shifted into FRAC55 R3.
;16384  ;       ErrorD - EXTENSION R0 failed to be shifted into FRAC55 Q3
;16385  ;       ErrorE - Ones failed to be shifted left into the registers in both data
;16386  ;                paths.
;16387  ;       ErrorF - Same as error E.
;16388  ;       Error10 - Same as error E.
;16389  ;       Error11 - Same as error E.
;16390  ;       Error12 - Zeros failed to be shifted into the registers of the exponent
;16391  ;                 data path and the QR of the fraction data path or FRAC R3 SAV
;16392  ;                 failed to be shifted into R0.
;16393  ;       Error13 - Zeros failed to be shifted into the registers of the fraction
;16394  ;                 data path or FRAC55 Q3 failed to be shifted into Q0 or FRAC55
;16395  ;                 R3 failed to be shifted into R0.
;16396  ;       Error14 - Same as error 13.
;16397  ;       Error15 - FRAC55 Q3 failed to be shifted into Q0 or FRAC55 R3 failed to
;16398  ;                 be shifted into R0 or for the fraction data path, Qin failed
;16399  ;                 to be shifted into Q0 or FRAC55 Q3 failed to be shifted into
;16400  ;                 R0.
;16401  ;       Error16 - Same as error 15.
;16402  ;
;16403  ; Debug:
;16404  ;
;16405  ;       Error1: If there's an error then the FRAC SHIFT CTL PAL should be
;16406  ;               checked for error. The outputs FPAL FRAC55 R3 H and FPAL FRAC55
;16407  ;               Q3 H should be both be asserted and FPAM ENB MUL SHF H should
;16408  ;               be unasserted. For this paticular test FPAE SHF (1) H should be
;16409  ;               unasserted, FPAE SHF (0) H should be asserted, FPAD EXTEND CLK
;16410  ;               (1) H should be unasserted and FPAC ENB CLK3 L should be
;16411  ;               asserted. The destination bits of both the fraction and expo-
;16412  ;               nent data path are set for a right shift which means that FPAE
;16413  ;               FRAC I8 (1) H is asserted and FPAE FRAC I7 (1) H is unasserted.
;16414  ;               If the shift signals are in error then the latch they came from
;16415  ;               should be checked for error. If the clock extend signal in
;16416  ;               error then the latch it came from should be checked for error.
;16417  ;               If the latch is not in error then the clock gates should be
;16418  ;               checked for error. If FPAC ENB CLK3 L is in error then the
;16419  ;               clock decoder should be checked for error.
;16420  ;       Error2: Same as error 1 except it's the QR instead of the FWR.
;16421  ;       Error3: Same as error 1 except that the shift field should be 11.
;16422  ;       Error4: Same as error 3 except it's the QR instead of the FWR.
;16423  ;       Error5: Same as error1 except that the shift field should be 00 and
```

```
;16424  ;              FPAF EXT00 R0 H and FPAM E0 R0 H should be alternately high
;16425  ;              and then low after each shift. Both of these signals are out-
;16426  ;              puts from other 2901 so those signals should be checked.
;16427  ;       Error6: Same as the error 7 except it's the QR instead of the FWR.
;16428  ;       Error7: For the remaining errors FPAD EXTEND CLK (1)  can be anything
;16429  ;              but FPAC ENB CLK3 L must be unasserted. The rest is the same
;16430  ;              as error 7.
;16431  ;       Error8: Same as error 9 except it's the QR instead of the FWR.
;16432  ;       Error9: Same as error 5 except that FPAD EXTEND CLK (1) can be anything
;16433  ;              and FPAC ENB CLK3 L must be unasserted.
;16434  ;       ErrorA: Same as error 7 except that FPAD EXTEND CLK (1) can be anything
;16435  ;              and FPAC ENB CLK3 L must be unasserted and the out put from
;16436  ;              the 2901 to be checked is EXPONENT Q0.
;16437  ;       ErrorB: Same as error C excepts it's the QR instead of the FWR.
;16438  ;       ErrorC: Same as error one except that FPAD EXTEND CLK (1) can be any-
;16439  ;              thing, FPAC ENB CLK3 L must be unasserted and the output from
;16440  ;              the LSB of the extension data path is being shifted into the
;16441  ;              MSB of the QR of the fraction data path. If what is shifted in-
;16442  ;              to the MSB of the WR of the fraction is in error then the gates
;16443  ;              going into the most signifigant 2901 in the fraction data path
;16444  ;              should be checked for error.
;16445  ;       ErrorD: Same as error C except that instead of FWR it's the FQ.
;16446  ;       ErrorE: If this error occured then the FRAC SHIFT CTL PAL should be
;16447  ;              checked for error. Q0 and R0 of the both data paths should be
;16448  ;              one. The shift inputs should be 10, enable clk3 should be
;16449  ;              unasserted and the destination inputs for the control logic
;16450  ;              should be 11. If any of these inputs are in error then see
;16451  ;              error 1.
;16452  ;       ErrorF: Same as error E.
;16453  ;       Error10: Same as error E.
;16454  ;       Error11: Same as error E.
;16455  ;       Error12: If this error occurs then the FRAC SHIFT CTL PAL should be
;16456  ;              checked for error. Q0 of both data paths and R0 of the expo-
;16457  ;              nent data path should be zeros. R0 of the fraction data path
;16458  ;              should be 1 after the second shift and 0 after the third shift
;16459  ;              The shift inputs should be 01, enable clk3 should be unas-
;16460  ;              serted and the destination inputs for the control logic should
;16461  ;              be 11. If either of the fraction inputs fail then the
;16462  ;              EXTENSION DATA PATH MUX should be checked for error. If FPAC
;16463  ;              FRAC55 R3 SAVE H is in error then the STATUS REG 1 LATCH
;16464  ;              should be checked for error.
;16465  ;       Error13: If this error occurs then the FRAC SHIFT CTL PAL should be
;16466  ;              checked for error. Q0 and R0 of the fraction data path should
;16467  ;              be zeros. Q0 of the exponent data path should be FRAC55 Q3
;16468  ;              and R0 of the fraction data should be FRAC55 R3. The shift
;16469  ;              field should be 00. The destination inputs should both be one,
;16470  ;              and enb ckl3 should be unasserted. If any of the fraction
;16471  ;              inputs are in error then the EXTENSION DATA PATH MUX should be
;16472  ;              checked for error. If any other signals are in error see error
;16473  ;              1.
;16474  ;       Error14: Same as error 13.
;16475  ;       Error15: If this error occurs then the LSB of both registers of the
;16476  ;              exponent data path and the QR of the fraction data path should
;16477  ;              be the MSB of the respctive data path and register. Q0 should
;16478  ;              be QIN. If QIN is in error then the instruction encode bits
```

```
                ;16479  ;               should be 00101 (from MSB to LSB), the MOD field should be 10.
                ;16480  ;               The shift field should be 11, the destination control bits
                ;16481  ;               should be 11, and enb clk3 should be unasserted. If any of
                ;16482  ;               these signals are in error see error16.
                ;16483  ;       Error16: Same as error 15.
                ;16484  ;
                ;16485  ;
                ;16486  ;--
                ;16487  T.26:
U 166D, B65E,15 ;16488          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;16489                                          ; STATUS WORD
U 166E, 3E80,15 ;16490          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;16491                                          ; BIT 15 INDICATES START OF TEST TO
                ;16492                                          ; CONSOLE
U 166F, 10E0,15 ;16493          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;16494  WAIT.T26.0:
U 1670, 8967,04 ;16495          JMP [WAIT.T26.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;16496                                          ; RESPOND
U 1671, 087E,6C ;16497          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;16498                                          ; AND SIZE BITS
U 1672, B670,15 ;16499          MOV LS[OTHER.DATA] TO WR[0]     ;set up the other data field
U 1673, 3E80,15 ;16500          MOV WR[0] TO LS[CONTROL.STATUS]
U 1674, 365F,15 ;16501          MOV LS[BIT15] TO WR[2]          ;Setup first float mask
U 1675, B725,95 ;16502          MOV LS[#FFFF00FF] TO WR[3]      ;Setup third float mask
U 1676, B7AC,15 ;16503          MOV LS[T26.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1677, BE12,15 ;16504          MOV WR[0] TO LS[T9]
U 1678, 8870,3C ;16505          JSR [LO]                        ;Get address of SHF RIGHT FWR[FT1]
U 1679, 3E10,15 ;16506          MOV WR[0] TO LS[T8]              ; AND FQ IN[ONE]X
U 167A, 8870,3C ;16507          JSR [LO]                        ;Get address of SHF RIGHT FWR[FT0]
U 167B, BE14,15 ;16508          MOV WR[0] TO LS[T10]             ; AND FQ IN[ZERO]X
U 167C, 8870,3C ;16509          JSR [LO]                        ;Get address of SHF RIGHT FWR[FT0]
U 167D, 3E16,15 ;16510          MOV WR[0] TO LS[T11]             ; AND FQ IN[HUGE.ADJ]X
U 167E, 8870,3C ;16511          JSR [LO]                        ;Get address of STORE FIRST FLT FT1
U 167F, 3EAE,15 ;16512          MOV WR[0] TO LS[T57]
U 1680, 8870,3C ;16513          JSR [LO]                        ;Get address of STORE SECOND FLT FT1
U 1681, 3EB0,15 ;16514          MOV WR[0] TO LS[T58]
U 1682, 8870,3C ;16515          JSR [LO]                        ;Get address of STORE THIRD FLT FT1
U 1683, BEB2,15 ;16516          MOV WR[0] TO LS[T59]
U 1684, 8870,3C ;16517          JSR [LO]                        ;Get address of MOV FT0 TO FT1
U 1685, 3E1C,15 ;16518          MOV WR[0] TO LS[T14]
U 1686, 8870,3C ;16519          JSR [LO]                        ;Get address of MOV FT0 TO FQ
U 1687, 3F90,15 ;16520          MOV WR[0] TO LS[TC8]
U 1688, 8870,3C ;16521          JSR [LO]                        ;Get address of MOV FQ TO FT2
U 1689, BF92,15 ;16522          MOV WR[0] TO LS[TC9]
U 168A, B640,15 ;16523  T26.1:  MOV LS[#1] TO WR[0]             ;1 in WR
U 168B, BE82,15 ;16524          MOV WR[0] TO LS[ERROR.NUMBER]   ;Set up error number in LS.
U 168C, 087C,1C ;16525          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 168D, 0878,AC ;16526          JSR [MOV.DATA]                  ;MOV FT0 TO FT1
U 168E, B790,15 ;16527          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 168F, B716,95 ;16528          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1690, 90F6,15 ;16529          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 1691, 10F6,95 ;16530          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1692, B610,15 ;16531          MOV LS[T8] TO WR[0]             ;Setup to SHF RIGHT FWR[FT1] AND FQ
                ;16532                                          ; IN[ONE]X
U 1693, 90F6,15 ;16533          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT1] AND FQ IN[ONE]X
```

```
U 1694, 90F6,15 ;16534          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT1] AND FQ IN[ONE]X
U 1695, 90F6,15 ;16535          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT1] AND FQ IN[ONE]X
U 1696, 10F6,95 ;16536          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1697, 087A,BC ;16537          JSR [STORE.X.TRIPLE]            ;Store FT1 to the CPU
U 1698, B64C,15 ;16538          MOV LS[BIT6] TO WR[0]           ;Setup data for check.sub
U 1699, C74A,15 ;16539          BIS LS[BIT5] TO WR[0]
U 169A, 4748,15 ;16540          BIS LS[BIT4] TO WR[0]
U 169B, 3EA2,15 ;16541          MOV WR[0] TO LS[DATA.1]
U 169C, 887C,6C ;16542          JSR [CHECK.SUB]                 ;Check for error
U 169D, 8968,A4 ;16543          JMP [T26.1]                     ;Loop on error
U 169E, FF82,15 ;16544          INC LS[ERROR.NUMBER]            ;Go onto error2
U 169F, 3792,15 ;16545          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 16A0, B716,95 ;16546          MOV LS[#300] TO WR[1]           ;Setup to loop self
U 16A1, 90F6,15 ;16547          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 16A2, 10F6,95 ;16548          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16A3, 087B,5C ;16549          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 16A4, 887C,6C ;16550          JSR [CHECK.SUB]                 ;Check for error
U 16A5, 8968,A4 ;16551          JMP [T26.1]                     ;Loop on error
U 16A6, 36C0,15 ;16552  T26.2:  MOV LS[#3(H)] TO WR[0]          ;Set error number to 3
U 16A7, BE82,15 ;16553          MOV WR[0] TO LS[ERROR.NUMBER]
U 16A8, B69E,15 ;16554          MOV LS[ONES] TO WR[0]           ;Setup data
U 16A9, 3EA2,15 ;16555          MOV WR[0] TO LS[DATA.1]
U 16AA, 3EA4,15 ;16556          MOV WR[0] TO LS[DATA.2]
U 16AB, BEA6,15 ;16557          MOV WR[0] TO LS[DATA.3]
U 16AC, 0878,FC ;16558          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 16AD, B790,15 ;16559          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 16AE, B716,95 ;16560          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 16AF, 90F6,15 ;16561          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 16B0, 10F6,95 ;16562          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16B1, B69E,15 ;16563          MOV LS[ONES] TO WR[0]           ;Setup data
U 16B2, 4552,15 ;16564          BIC LS[BIT9] TO WR[0]
U 16B3, BEA6,15 ;16565          MOV WR[0] TO LS[DATA.3]
U 16B4, 0878,FC ;16566          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 16B5, 3614,15 ;16567          MOV LS[T10] TO WR[0]            ;Setup to SHF RIGHT FWR[FT0] AND FQ
                ;16568                                          ; AND IN[ZERO]X
U 16B6, B716,95 ;16569          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 16B7, 90F6,15 ;16570          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[ZERO]X
U 16B8, 90F6,15 ;16571          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[ZERO]X
U 16B9, 90F6,15 ;16572          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[ZERO]X
U 16BA, 10F6,95 ;16573          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16BB, 087B,5C ;16574          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 16BC, B69E,15 ;16575          MOV LS[ONES] TO WR[0]           ;Setup data
U 16BD, 3EA4,15 ;16576          MOV WR[0] TO LS[DATA.2]
U 16BE, BEA6,15 ;16577          MOV WR[0] TO LS[DATA.3]
U 16BF, 454A,15 ;16578          BIC LS[BIT5] TO WR[0]
U 16C0, 454C,15 ;16579          BIC LS[BIT6] TO WR[0]
U 16C1, 3EA2,15 ;16580          MOV WR[0] TO LS[DATA.1]
U 16C2, 887C,6C ;16581          JSR [CHECK.SUB]                 ;Check for error
U 16C3, 096A,64 ;16582          JMP [T26.2]                     ;Loop on error
U 16C4, FF82,15 ;16583          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 16C5, 3792,15 ;16584          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 16C6, B716,95 ;16585          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 16C7, 90F6,15 ;16586          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 16C8, 10F6,95 ;16587          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16C9, 087B,5C ;16588          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
```

```
U 16CA, 887C,6C ;16589          JSR [CHECK.SUB]                 ;Check for error
U 16CB, 096A,64 ;16590          JMP [T26.2]                     ;Loop on error
U 16CC, 3644,15 ;16591  T26.3:  MOV LS[#4] TO WR[0]             ;Set error # to 5
U 16CD, 2040,15 ;16592          INC WR[0]
U 16CE, BE82,15 ;16593          MOV WR[0] TO LS[ERROR.NUMBER]
U 16CF, B69E,15 ;16594          MOV LS[ONES] TO WR[0]           ;Setup data
U 16D0, 3EA2,15 ;16595          MOV WR[0] TO LS[DATA.1]
U 16D1, 3EA4,15 ;16596          MOV WR[0] TO LS[DATA.2]
U 16D2, BEA6,15 ;16597          MOV WR[0] TO LS[DATA.3]
U 16D3, 0878,FC ;16598          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 16D4, B790,15 ;16599          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 16D5, B716,95 ;16600          MOV LS[#300] TO WR[1]           ;Setup to loop self
U 16D6, 90F6,15 ;16601          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 16D7, 10F6,95 ;16602          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16D8, B616,15 ;16603          MOV LS[T11] TO WR[0]            ;Setup to SHF RIGHT FWR[FT0] AND FQ
                ;16604                                          ; IN[HUGE.ADJ]X
U 16D9, B716,95 ;16605          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 16DA, 90F6,15 ;16606          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[HUGE.ADJ]X
U 16DB, 90F6,15 ;16607          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[HUGE.ADJ]X
U 16DC, 90F6,15 ;16608          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[HUGE.ADJ]X
U 16DD, 10F6,95 ;16609          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16DE, 087B,5C ;16610          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 16DF, B69E,15 ;16611          MOV LS[ONES] TO WR[0]
U 16E0, 3EA4,15 ;16612          MOV WR[0] TO LS[DATA.2]
U 16E1, BEA6,15 ;16613          MOV WR[0] TO LS[DATA.3]
U 16E2, C55C,15 ;16614          BIC LS[BIT14] TO WR[0]
U 16E3, C55A,15 ;16615          BIC LS[BIT13] TO WR[0]
U 16E4, 4558,15 ;16616          BIC LS[BIT12] TO WR[0]
U 16E5, 3EA2,15 ;16617          MOV WR[0] TO LS[DATA.1]
U 16E6, 887C,6C ;16618          JSR [CHECK.SUB]                 ;Check for error
U 16E7, 096C,C4 ;16619          JMP [T26.3]                     ;Loop on error
U 16E8, FF82,15 ;16620          INC LS[ERROR.NUMBER]            ;Go onto error 6
U 16E9, 3792,15 ;16621          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 16EA, B716,95 ;16622          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 16EB, 90F6,15 ;16623          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 16EC, 10F6,95 ;16624          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 16ED, 087B,5C ;16625          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 16EE, 887C,6C ;16626          JSR [CHECK.SUB]                 ;Check for error
U 16EF, 096C,C4 ;16627          JMP [T26.3]                     ;Loop on error
U 16F0, FF82,15 ;16628          INC LS[ERROR.NUMBER]            ;Go onto error7
U 16F1, 8870,3C ;16629          JSR [L0]                        ;Get address of SHF RIGHT FWR[FT0] AND
U 16F2, 3E10,15 ;16630          MOV WR[0] TO LS[T8]              ; FQ IN[EXT.ROT] AND DEC EQ
U 16F3, 8870,3C ;16631          JSR [L0]                        ;Get address of SHF RIGHT FWR[FT5] AND
U 16F4, BE14,15 ;16632          MOV WR[0] TO LS[T10]              ; FQ IN[DELAY.ROT]
U 16F5, 8870,3C ;16633          JSR [L0]                        ;Get address of SHF RIGHT FWR[FT0]
U 16F6, 3E16,15 ;16634          MOV WR[0] TO LS[T11]              ; IN[EXP.SHF]
U 16F7, 8870,3C ;16635          JSR [L0]                        ;Get address of SHF RIGHT FWR[FT0]
U 16F8, BE18,15 ;16636          MOV WR[0] TO LS[T12]              ; AND FQ
U 16F9, 8870,3C ;16637          JSR [L0]                        ;Get address of MOV FT0 TO FT5
U 16FA, 3E1C,15 ;16638          MOV WR[0] TO LS[T14]
U 16FB, 8870,3C ;16639          JSR [L0]                        ;Get address of Store first flt FT6
U 16FC, 3EAE,15 ;16640          MOV WR[0] TO LS[T57]
U 16FD, 8870,3C ;16641          JSR [L0]                        ;Get address of Store second flt FT6
U 16FE, 3EB0,15 ;16642          MOV WR[0] TO LS[T58]
U 16FF, 8870,3C ;16643          JSR [L0]                        ;Get address of Store third flt FT7
```

```
U 1700, BEB2,15 ;16644          MOV WR[0] TO LS[T59]
U 1701, 8870,3C ;16645          JSR [L0]                        ;Get address of MOV FT5 TO FT6
U 1702, BE1E,15 ;16646          MOV WR[0] TO LS[T15]
U 1703, 8870,3C ;16647          JSR [L0]                        ;Get address of ADD SHFR FWR[FT2] TO
U 1704, 3E1A,15 ;16648          MOV WR[0] TO LS[T13]            ; FWR[FT0] + FCOUT
U 1705, 8870,3C ;16649          JSR [L0]                        ;Get address of MOV FT0 TO FT2
U 1706, BF08,15 ;16650          MOV WR[0] TO LS[T84]
U 1707, 8870,3C ;16651          JSR [L0]                        ;Get address of CLR FT0
U 1708, BF8C,15 ;16652          MOV WR[0] TO LS[TC5]
U 1709, B646,15 ;16653  T26.4:  MOV LS[#8] TO WR[0]             ;Set error # to 7
U 170A, 2100,15 ;16654          DEC WR[0]
U 170B, BE82,15 ;16655          MOV WR[0] TO LS[ERROR.NUMBER]
U 170C, B69E,15 ;16656          MOV LS[ONES] TO WR[0]           ;Setup data
U 170D, 3EA2,15 ;16657          MOV WR[0] TO LS[DATA.1]
U 170E, 3EA4,15 ;16658          MOV WR[0] TO LS[DATA.2]
U 170F, BEA6,15 ;16659          MOV WR[0] TO LS[DATA.3]
U 1710, 0878,FC ;16660          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 1711, B790,15 ;16661          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 1712, B716,95 ;16662          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1713, 90F6,15 ;16663          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 1714, 10F6,95 ;16664          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1715, B69E,15 ;16665          MOV LS[ONES] TO WR[0]           ;Setup data
U 1716, 4552,15 ;16666          BIC LS[BIT9] TO WR[0]
U 1717, BEA6,15 ;16667          MOV WR[0] TO LS[DATA.3]
U 1718, 0878,FC ;16668          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 1719, B610,15 ;16669          MOV LS[T8] TO WR[0]             ;Setup SHF RIGHT FWR[FT0] AND FQ
                ;16670                                          ; IN[EXT.ROT]
U 171A, B716,95 ;16671          MOV LS[#300] TO WR[1]           ;Setup loop self
U 171B, 90F6,15 ;16672          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]
                ;16673                                          ; AND DEC EQ
U 171C, 90F6,15 ;16674          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]
                ;16675                                          ; AND DEC EQ
U 171D, 90F6,15 ;16676          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT0] AND FQ IN[EXT.ROT]
                ;16677                                          ; AND DEC EQ
U 171E, 10F6,95 ;16678          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 171F, 087B,5C ;16679          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 1720, B69E,15 ;16680          MOV LS[ONES] TO WR[0]           ;Setup data
U 1721, 3EA4,15 ;16681          MOV WR[0] TO LS[DATA.2]
U 1722, BEA6,15 ;16682          MOV WR[0] TO LS[DATA.3]
U 1723, 454C,15 ;16683          BIC LS[BIT6] TO WR[0]
U 1724, 454A,15 ;16684          BIC LS[BIT5] TO WR[0]
U 1725, 3EA2,15 ;16685          MOV WR[0] TO LS[DATA.1]
U 1726, 887C,6C ;16686          JSR [CHECK.SUB]                 ;Check for error
U 1727, 8970,94 ;16687          JMP [T26.4]                     ;Loop on error
U 1728, FF82,15 ;16688          INC LS[ERROR.NUMBER]            ;Go onto error 8
U 1729, 3792,15 ;16689          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 172A, B716,95 ;16690          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 172B, 90F6,15 ;16691          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 172C, 10F6,95 ;16692          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 172D, 087B,5C ;16693          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 172E, B69E,15 ;16694          MOV LS[ONES] TO WR[0]           ;Setup data
U 172F, 454C,15 ;16695          BIC LS[BIT6] TO WR[0]
U 1730, C54E,15 ;16696          BIC LS[BIT7] TO WR[0]
U 1731, C550,15 ;16697          BIC LS[BIT8] TO WR[0]
U 1732, 3EA2,15 ;16698          MOV WR[0] TO LS[DATA.1]
```

```
U 1733, 887C,6C ;16699          JSR [CHECK.SUB]                 ;Check for error
U 1734, 8970,94 ;16700          JMP [T26.4]                     ;Loop on error
U 1735, FF82,15 ;16701          INC LS[ERROR.NUMBER]            ;Go onto error 9
U 1736, C749,15 ;16702          BIS LS[BIT4] TO WR[2]           ;Change error mask to mask out bit
                ;16703                                          ;53 for EXTRO SAVE shift
U 1737, B646,15 ;16704  T26.5:  MOV LS[#8] TO WR[0]             ;Set error # to 9
U 1738, 2040,15 ;16705          INC WR[0]
U 1739, BE82,15 ;16706          MOV WR[0] TO LS[ERROR.NUMBER]
U 173A, B69E,15 ;16707          MOV LS[ONES] TO WR[0]           ;Setup data
U 173B, 3EA2,15 ;16708          MOV WR[0] TO LS[DATA.1]
U 173C, 3EA4,15 ;16709          MOV WR[0] TO LS[DATA.2]
U 173D, BEA6,15 ;16710          MOV WR[0] TO LS[DATA.3]
U 173E, 0878,FC ;16711          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 173F, B790,15 ;16712          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 1740, B716,95 ;16713          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1741, 90F6,15 ;16714          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 1742, 10F6,95 ;16715          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1743, B69E,15 ;16716          MOV LS[ONES] TO WR[0]           ;Setup data
U 1744, 4552,15 ;16717          BIC LS[BIT9] TO WR[0]
U 1745, BEA6,15 ;16718          MOV WR[0] TO LS[DATA.3]
U 1746, 0878,FC ;16719          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 1747, 0878,AC ;16720          JSR [MOV.DATA]                  ;MOV FT0 TO FT5
U 1748, 3614,15 ;16721          MOV LS[T10] TO WR[0]            ;Setup to SHF RIGHT FWR[FT5] AND FQ
                ;16722                                          ; IN[DELAY.ROT]
U 1749, B716,95 ;16723          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 174A, 90F6,15 ;16724          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT5] AND FQ IN[DELAY.ROT]
U 174B, 90F6,15 ;16725          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT5] AND FQ IN[DELAY.ROT]
U 174C, 90F6,15 ;16726          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT5] AND FQ IN[DELAY.ROT]
U 174D, 90F6,15 ;16727          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT FWR[FT5] AND FQ IN[DELAY.ROT]
U 174E, 10F6,95 ;16728          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 174F, 361E,15 ;16729          MOV LS[T15] TO WR[0]            ;Setup to MOV FT5 TO FT6
U 1750, B716,95 ;16730          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1751, 90F6,15 ;16731          MISC2 [TRAP.ACC] WR[0]          ;MOV FT5 TO FT6
U 1752, 10F6,95 ;16732          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1753, 087A,BC ;16733          JSR [STORE.X.TRIPLE]            ;Store FT0 to the CPU
U 1754, B69E,15 ;16734          MOV LS[ONES] TO WR[0]           ;Setup data
U 1755, 3EA4,15 ;16735          MOV WR[0] TO LS[DATA.2]
U 1756, BEA6,15 ;16736          MOV WR[0] TO LS[DATA.3]
U 1757, 454C,15 ;16737          BIC LS[BIT6] TO WR[0]
U 1758, C548,15 ;16738          BIC LS[BIT4] TO WR[0]
U 1759, 3EA2,15 ;16739          MOV WR[0] TO LS[DATA.1]
U 175A, 887C,6C ;16740          JSR [CHECK.SUB]                 ;Check for error
U 175B, 0973,74 ;16741          JMP [T26.5]                     ;Loop on error
U 175C, FF82,15 ;16742          INC LS[ERROR.NUMBER]            ;Go onto error A
U 175D, 3792,15 ;16743          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 175E, B716,95 ;16744          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 175F, 90F6,15 ;16745          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 1760, 10F6,95 ;16746          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1761, 087B,5C ;16747          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 1762, B6A2,15 ;16748          MOV LS[DATA.1] TO WR[0]         ;Setup data
U 1763, 454A,15 ;16749          BIC LS[BIT5] TO WR[0]
U 1764, 3EA2,15 ;16750          MOV WR[0] TO LS[DATA.1]
U 1765, 887C,6C ;16751          JSR [CHECK.SUB]                 ;Check for error
U 1766, 0973,74 ;16752          JMP [T26.5]                     ;Loop on error
U 1767, FF82,15 ;16753          INC LS[ERROR.NUMBER]            ;Go onto error B
```

```
U 1768, 4549,15 ;16754          BIC LS[BIT4] TO WR[2]          ;Set error mask back
U 1769, B69E,15 ;16755  T26.6:  MOV LS[ONES] TO WR[0]          ;Setup data
U 176A, 3EA2,15 ;16756          MOV WR[0] TO LS[DATA.1]
U 176B, 3EA4,15 ;16757          MOV WR[0] TO LS[DATA.2]
U 176C, BEA6,15 ;16758          MOV WR[0] TO LS[DATA.3]
U 176D, 0878,FC ;16759          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 176E, B790,15 ;16760          MOV LS[TC8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 176F, B716,95 ;16761          MOV LS[#300] TO WR[1]          ;Setup to Loop self
U 1770, 90F6,15 ;16762          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 1771, 10F6,95 ;16763          MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 1772, B69E,15 ;16764          MOV LS[ONES] TO WR[0]          ;Setup data
U 1773, C54E,15 ;16765          BIC LS[BIT7] TO WR[0]
U 1774, 3EA2,15 ;16766          MOV WR[0] TO LS[DATA.1]
U 1775, 0878,FC ;16767          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 1776, B616,15 ;16768          MOV LS[T11] TO WR[0]           ;Setup to SHF RIGHT FWR[FT0] IN[EXP.SHF]
U 1777, B716,95 ;16769          MOV LS[#300] TO WR[1]          ;Setup to loop self
U 1778, 90F6,15 ;16770          MISC2 [TRAP.ACC] WR[0]         ;SHF RIGHT FWR[FT0] INEXP.SHF]
U 1779, 90F6,15 ;16771          MISC2 [TRAP.ACC] WR[0]         ;SHF RIGHT FWR[FT0] INEXP.SHF]
U 177A, 90F6,15 ;16772          MISC2 [TRAP.ACC] WR[0]         ;SHF RIGHT FWR[FT0] INEXP.SHF]
U 177B, 10F6,95 ;16773          MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 177C, 087B,5C ;16774          JSR [STORE.0.TRIPLE]           ;Store FT0 to the CPU
U 177D, B69E,15 ;16775          MOV LS[ONES] TO WR[0]          ;Setup data
U 177E, 454A,15 ;16776          BIC LS[BIT5] TO WR[0]
U 177F, C55C,15 ;16777          BIC LS[BIT14] TO WR[0]
U 1780, C55A,15 ;16778          BIC LS[BIT13] TO WR[0]
U 1781, 4558,15 ;16779          BIC LS[BIT12] TO WR[0]
U 1782, 3EA2,15 ;16780          MOV WR[0] TO LS[DATA.1]
U 1783, 887C,6C ;16781          JSR [CHECK.SUB]                ;Check for error
U 1784, 8976,94 ;16782          JMP [T26.6]                    ;Loop self
U 1785, FF82,15 ;16783          INC LS[ERROR.NUMBER]           ;Go onto error C
U 1786, 087C,1C ;16784  T26.7:  JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 1787, 3708,15 ;16785          MOV LS[T84] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 1788, B716,95 ;16786          MOV LS[#300] TO WR[1]          ;Setup to Loop self
U 1789, 90F6,15 ;16787          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 178A, 10F6,95 ;16788          MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 178B, B61A,15 ;16789          MOV LS[T13] TO WR[0]           ;Setup to ADD SHFR FWR[FT2] TO
                ;16790                                         ;FWR[FT0] + FCOUT
U 178C, B716,95 ;16791          MOV LS[#300] TO WR[1]          ;Setup to Loop self
U 178D, 90F6,15 ;16792          MISC2 [TRAP.ACC] WR[0]         ;ADD SHFR FWR[FT2] TO FWR[FT0] + FCOUT
U 178E, 10F6,95 ;16793          MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 178F, 3618,15 ;16794          MOV LS[T12] TO WR[0]           ;Setup to SHF RIGHT HUGE FWR[FT0] AND FQ
U 1790, 90F6,15 ;16795          MISC2 [TRAP.ACC] WR[0]         ;SHF RIGHT HUGE FWR[FT0] AND FQ
U 1791, 10F6,95 ;16796          MISC2 [TRAP.ACC] WR[1]         ;Loop self
U 1792, 087B,5C ;16797          JSR [STORE.0.TRIPLE]           ;Store FT0 to the CPU
U 1793, 65A6,15 ;16798          CLR LS[DATA.3]                 ;Setup data
U 1794, E5A4,15 ;16799          CLR LS[DATA.2]
U 1795, B64C,15 ;16800          MOV LS[BIT6] TO WR[0]
U 1796, 3EA2,15 ;16801          MOV WR[0] TO LS[DATA.1]
U 1797, 887C,6C ;16802          JSR [CHECK.SUB]                ;Check for error
U 1798, 0978,64 ;16803          JMP [T26.7]                    ;Loop on error
                ;16804                                         ;Change data but testing the same shift
                ;16805                                         ; so the same error # is used.
U 1799, 378C,15 ;16806  T26.7A: MOV LS[TC5] TO WR[0]           ;Setup to CLR FT0
U 179A, B716,95 ;16807          MOV LS[#300] TO WR[1]          ;Setup to Loop self
U 179B, 90F6,15 ;16808          MISC2 [TRAP.ACC] WR[0]         ;CLR FT0
```

```
U 179C, 10F6,95 ;16809          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 179D, 3708,15 ;16810          MOV LS[T84] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 179E, B716,95 ;16811          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 179F, 90F6,15 ;16812          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 17A0, 10F6,95 ;16813          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17A1, B61A,15 ;16814          MOV LS[T13] TO WR[0]            ;Setup to ADD SHFR FWR[FT2] TO
                ;16815                                          ;FWR[FT0] + FCOUT
U 17A2, B716,95 ;16816          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 17A3, 90F6,15 ;16817          MISC2 [TRAP.ACC] WR[0]          ;ADD SHFR FWR[FT2] TO FWR[FT0] + FCOUT
U 17A4, 10F6,95 ;16818          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17A5, 3618,15 ;16819          MOV LS[T12] TO WR[0]            ;Setup to SHF RIGHT HUGE FWR[FT0] AND FQ
U 17A6, 90F6,15 ;16820          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT HUGE FWR[FT0] AND FQ
U 17A7, 10F6,95 ;16821          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17A8, 087B,5C ;16822          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 17A9, 65A6,15 ;16823          CLR LS[DATA.3]                  ;Setup data
U 17AA, E5A4,15 ;16824          CLR LS[DATA.2]
U 17AB, E5A2,15 ;16825          CLR LS[DATA.1]
U 17AC, 887C,6C ;16826          JSR [CHECK.SUB]                 ;Check for error
U 17AD, 8979,94 ;16827          JMP [T26.7A]                    ;Loop on error
U 17AE, FF82,15 ;16828          INC LS[ERROR.NUMBER]            ;Go onto error D
U 17AF, 087C,1C ;16829  T26.7B: JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 17B0, B790,15 ;16830          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 17B1, B716,95 ;16831          MOV LS[#300] TO WR[1]           ;Setup Loop self
U 17B2, 90F6,15 ;16832          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 17B3, 10F6,95 ;16833          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17B4, B652,15 ;16834          MOV LS[BIT9] TO WR[0]
U 17B5, BEA6,15 ;16835          MOV WR[0] TO LS[DATA.3]
U 17B6, 0878,FC ;16836          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 17B7, 3618,15 ;16837          MOV LS[T12] TO WR[0]            ;Setup to SHF RIGHT HUGE FWR[FT0]
                ;16838                                          ; AND FQ
U 17B8, B716,95 ;16839          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 17B9, 90F6,15 ;16840          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT HUGE FWR[FT0] AND FQ
U 17BA, 90F6,15 ;16841          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT HUGE FWR[FT0] AND FQ
U 17BB, 90F6,15 ;16842          MISC2 [TRAP.ACC] WR[0]          ;SHF RIGHT HUGE FWR[FT0] AND FQ
U 17BC, 10F6,95 ;16843          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17BD, 3792,15 ;16844          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 17BE, B716,95 ;16845          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 17BF, 90F6,15 ;16846          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 17C0, 10F6,95 ;16847          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17C1, 087B,5C ;16848          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 17C2, 3648,15 ;16849          MOV LS[BIT4] TO WR[0]           ;Setup data
U 17C3, C74C,15 ;16850          BIS LS[BIT6] TO WR[0]
U 17C4, 475C,15 ;16851          BIS LS[BIT14] TO WR[0]
U 17C5, 475A,15 ;16852          BIS LS[BIT13] TO WR[0]
U 17C6, C758,15 ;16853          BIS LS[BIT12] TO WR[0]
U 17C7, 4756,15 ;16854          BIS LS[BIT11] TO WR[0]
U 17C8, C754,15 ;16855          BIS LS[BIT10] TO WR[0]
U 17C9, C752,15 ;16856          BIS LS[BIT9] TO WR[0]
U 17CA, 474E,15 ;16857          BIS LS[BIT7] TO WR[0]
U 17CB, 3EA2,15 ;16858          MOV WR[0] TO LS[DATA.1]
U 17CC, E5A4,15 ;16859          CLR LS[DATA.2]
U 17CD, 65A6,15 ;16860          CLR LS[DATA.3]
U 17CE, 887C,6C ;16861          JSR [CHECK.SUB]                 ;Check for error
U 17CF, 897A,F4 ;16862          JMP [T26.7B]                    ;Loop on error
U 17D0, FF82,15 ;16863          INC LS[ERROR.NUMBER]            ;Go onto error E
```

```
U 17D1, 8870,3C ;16864          JSR [LO]                        ;Get address of MOV FTO TO D.RND
U 17D2, 3E1C,15 ;16865          MOV WR[0] TO LS[T14]
U 17D3, 8870,3C ;16866          JSR [LO]                        ;Get address of STORE FIRST FLT FTE
U 17D4, 3EAE,15 ;16867          MOV WR[0] TO LS[T57]
U 17D5, 8870,3C ;16868          JSR [LO]                        ;Get address of STORE SEC FLT FTE
U 17D6, 3EB0,15 ;16869          MOV WR[0] TO LS[T58]
U 17D7, 8870,3C ;16870          JSR [LO]                        ;Get address of STORE THIRD FLT FTE
U 17D8, BEB2,15 ;16871          MOV WR[0] TO LS[T59]
U 17D9, 8870,3C ;16872          JSR [LO]                        ;Get address of SHF LEFT FWR[D.RND] AND
U 17DA, 3E1A,15 ;16873          MOV WR[0] TO LS[T13]            ; FQ IN[ONE]
U 17DB, 3648,15 ;16874  T26.8:  MOV LS[#10] TO WR[0]            ;Set up error # to E
U 17DC, 2100,15 ;16875          DEC WR[0]
U 17DD, 2100,15 ;16876          DEC WR[0]
U 17DE, BE82,15 ;16877          MOV WR[0] TO LS[ERROR.NUMBER]
U 17DF, 378C,15 ;16878          MOV LS[TC5] TO WR[0]            ;Setup to CLR FTO
U 17E0, B716,95 ;16879          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 17E1, 90F6,15 ;16880          MISC2 [TRAP.ACC] WR[0]          ;CLR FTO
U 17E2, 10F6,95 ;16881          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17E3, B790,15 ;16882          MOV LS[TC8] TO WR[0]            ;Setup to MOV FTO TO FQ
U 17E4, 90F6,15 ;16883          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FQ
U 17E5, 10F6,95 ;16884          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17E6, 0878,AC ;16885          JSR [MOV.DATA]                  ;MOV FTO TO FTE
U 17E7, B61A,15 ;16886          MOV LS[T13] TO WR[0]            ;Setup to SHF LEFT FWR[D.RND] AND FQ
                ;16887                                          ; IN[ONE], DEC EQ
U 17E8, 90F6,15 ;16888          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT FWR[D.RND] AND FQ IN[ONE,
                ;16889                                          ; DEC EQ
U 17E9, 90F6,15 ;16890          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT FWR[D.RND] AND FQ IN[ONE,
                ;16891                                          ; DEC EQ
U 17EA, 10F6,95 ;16892          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17EB, 087A,BC ;16893          JSR [STORE.X.TRIPLE]            ;Store FTE to the CPU
U 17EC, 3792,15 ;16894          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FTO
U 17ED, B716,95 ;16895          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 17EE, 90F6,15 ;16896          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FTO
U 17EF, 10F6,95 ;16897          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 17F0, E5A4,15 ;16898          CLR LS[DATA.2]                  ;Setup data
U 17F1, E5A2,15 ;16899          CLR LS[DATA.1]
U 17F2, 3650,15 ;16900          MOV LS[BIT8] TO WR[0]
U 17F3, C752,15 ;16901          BIS LS[BIT9] TO WR[0]
U 17F4, BEA6,15 ;16902          MOV WR[0] TO LS[DATA.3]
U 17F5, 887C,6C ;16903          JSR [CHECK.SUB]                 ;Check for error
U 17F6, 897D,B4 ;16904          JMP [T26.8]                     ;Loop on error
U 17F7, FF82,15 ;16905          INC LS[ERROR.NUMBER]            ;Go onto error F
U 17F8, 087B,5C ;16906          JSR [STORE.0.TRIPLE]            ;Store FTO to the CPU
U 17F9, 365C,15 ;16907          MOV LS[BIT14] TO WR[0]          ;Setup data
U 17FA, 475A,15 ;16908          BIS LS[BIT13] TO WR[0]
U 17FB, C758,15 ;16909          BIS LS[BIT12] TO WR[0]
U 17FC, 4756,15 ;16910          BIS LS[BIT11] TO WR[0]
U 17FD, C754,15 ;16911          BIS LS[BIT10] TO WR[0]
U 17FE, C752,15 ;16912          BIS LS[BIT9] TO WR[0]
U 17FF, 4750,15 ;16913          BIS LS[BIT8] TO WR[0]
U 1800, 3EA2,15 ;16914          MOV WR[0] TO LS[DATA.1]
U 1801, E5A4,15 ;16915          CLR LS[DATA.2]
U 1802, 3650,15 ;16916          MOV LS[BIT8] TO WR[0]
U 1803, C752,15 ;16917          BIS LS[BIT9] TO WR[0]
U 1804, BEA6,15 ;16918          MOV WR[0] TO LS[DATA.3]
```

```
U 1805, 887C,6C ;16919          JSR [CHECK.SUB]                 ;Check for error
U 1806, 897D,B4 ;16920          JMP [T26.8]                     ;Loop on error
U 1807, FF82,15 ;16921          INC LS[ERROR.NUMBER]            ;Go onto error 10
U 1808, 8870,3C ;16922          JSR [LO]                        ;Get address of MOV ET0 TO G.MAX
U 1809, 3E1C,15 ;16923          MOV WR[0] TO LS[T14]
U 180A, 8870,3C ;16924          JSR [LO]                        ;Get address of STORE FIRST FLT FTD
U 180B, 3EAE,15 ;16925          MOV WR[0] TO LS[T57]
U 180C, 8870,3C ;16926          JSR [LO]                        ;Get address of STORE SEC FLT FTD
U 180D, 3EB0,15 ;16927          MOV WR[0] TO LS[T58]
U 180E, 8870,3C ;16928          JSR [LO]                        ;Get address of STORE THIRD FLT FTD
U 180F, BEB2,15 ;16929          MOV WR[0] TO LS[T59]
U 1810, 8870,3C ;16930          JSR [LO]                        ;Get address of SHF LEFT EWR[G.MAX] AND
U 1811, 3E1A,15 ;16931          MOV WR[0] TO LS[T13]            ; EQ IN[ONE]
U 1812, 3648,15 ;16932  T26.9:  MOV LS[#10] TO WR[0]            ;Setup error # to 10
U 1813, BE82,15 ;16933          MOV WR[0] TO LS[ERROR.NUMBER]
U 1814, 378C,15 ;16934          MOV LS[TC5] TO WR[0]            ;Setup to CLR ET0
U 1815, B716,95 ;16935          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1816, 90F6,15 ;16936          MISC2 [TRAP.ACC] WR[0]          ;CLR ET0
U 1817, 10F6,95 ;16937          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1818, B790,15 ;16938          MOV LS[TC8] TO WR[0]            ;Setup to MOV ET0 TO EQ
U 1819, 90F6,15 ;16939          MISC2 [TRAP.ACC] WR[0]          ;MOV ET0 TO EQ
U 181A, 10F6,95 ;16940          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 181B, 0878,AC ;16941          JSR [MOV.DATA]                  ;MOV ET0 TO ETD
U 181C, B61A,15 ;16942          MOV LS[T13] TO WR[0]            ;Setup to SHF LEFT EWR[G.MAX] AND EQ
                ;16943                                          ; IN[ONE],
U 181D, 90F6,15 ;16944          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT EWR[G.MAX] AND EQ IN[ONE]
U 181E, 90F6,15 ;16945          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT EWR[G.MAX] AND EQ IN[ONE]
U 181F, 10F6,95 ;16946          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1820, 087A,BC ;16947          JSR [STORE.X.TRIPLE]            ;Store ETE to the CPU
U 1821, 3792,15 ;16948          MOV LS[TC9] TO WR[0]            ;Setup to MOV EQ TO ET0
U 1822, B716,95 ;16949          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1823, 90F6,15 ;16950          MISC2 [TRAP.ACC] WR[0]          ;MOV EQ TO ET0
U 1824, 10F6,95 ;16951          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1825, 65A6,15 ;16952          CLR LS[DATA.3]                  ;Setup data
U 1826, E5A4,15 ;16953          CLR LS[DATA.2]
U 1827, 364E,15 ;16954          MOV LS[BIT7] TO WR[0]
U 1828, 4750,15 ;16955          BIS LS[BIT8] TO WR[0]
U 1829, 3EA2,15 ;16956          MOV WR[0] TO LS[DATA.1]
U 182A, 65AC,15 ;16957          CLR LS[THIRD.FLT]
U 182B, 887C,6C ;16958          JSR [CHECK.SUB]                 ;Check for error
U 182C, 8981,24 ;16959          JMP [T26.9]                     ;Loop on error
U 182D, FF82,15 ;16960          INC LS[ERROR.NUMBER]            ;Go onto error 11
U 182E, 087B,5C ;16961          JSR [STORE.0.TRIPLE]            ;Store ET0 to the CPU
U 182F, 887C,6C ;16962          JSR [CHECK.SUB]                 ;Check for error
U 1830, 8981,24 ;16963          JMP [T26.9]                     ;Loop on error
U 1831, FF82,15 ;16964          INC LS[ERROR.NUMBER]            ;Go onto error 12
U 1832, 8870,3C ;16965          JSR [LO]                        ;Get address of MOV FT0 TO FT4
U 1833, 3E1C,15 ;16966          MOV WR[0] TO LS[T14]
U 1834, 8870,3C ;16967          JSR [LO]                        ;Get address of STORE FIRST FLT FT4
U 1835, 3EAE,15 ;16968          MOV WR[0] TO LS[T57]
U 1836, 8870,3C ;16969          JSR [LO]                        ;Get address of STORE SEC FLT FT4
U 1837, 3EB0,15 ;16970          MOV WR[0] TO LS[T58]
U 1838, 8870,3C ;16971          JSR [LO]                        ;Get address of STORE THIRD FLT FT4
U 1839, BEB2,15 ;16972          MOV WR[0] TO LS[T59]
U 183A, 8870,3C ;16973          JSR [LO]                        ;Get address of SHF LEFT FWR[FT4] IN
```

```
U 183B, 3E1A,15 ;16974          MOV WR[0] TO LS[T13]          ; [LF.ROT], DEC EQ
U 183C, C755,95 ;16975          BIS LS[BIT10] TO WR[3]        ;Setup error mask for left shift in of
                ;16976                                        ; FRAC55 R3 SAVE
U 183D, B69E,15 ;16977  T26.A:  MOV LS[ONES] TO WR[0]         ;Setup data
U 183E, 3EA4,15 ;16978          MOV WR[0] TO LS[DATA.2]
U 183F, BEA6,15 ;16979          MOV WR[0] TO LS[DATA.3]
U 1840, 454C,15 ;16980          BIC LS[BIT6] TO WR[0]
U 1841, 3EA2,15 ;16981          MOV WR[0] TO LS[DATA.1]
U 1842, 0878,FC ;16982          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 1843, 0878,AC ;16983          JSR [MOV.DATA]                ;MOV FT0 TO FTD
U 1844, B61A,15 ;16984          MOV LS[T13] TO WR[0]          ;Setup to SHF LEFT FWR[FT4] IN[LF.ROT]
U 1845, B716,95 ;16985          MOV LS[#300] TO WR[1]         ;Setup to Loop self
U 1846, 90F6,15 ;16986          MISC2 [TRAP.ACC] WR[0]        ;SHF LEFT FWR[FT4] IN[LF.ROT]
                ;16987                                        ; DEC EQ
U 1847, 90F6,15 ;16988          MISC2 [TRAP.ACC] WR[0]        ;SHF LEFT FWR[FT4] IN[LF.ROT]
                ;16989                                        ; DEC EQ
U 1848, 90F6,15 ;16990          MISC2 [TRAP.ACC] WR[0]        ;SHF LEFT FWR[FT4] IN[LF.ROT]
                ;16991                                        ; DEC EQ
U 1849, 10F6,95 ;16992          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 184A, 087A,BC ;16993          JSR [STORE.X.TRIPLE]          ;Store FTD to the CPU
U 184B, B69E,15 ;16994          MOV LS[ONES] TO WR[0]         ;Setup data
U 184C, 3EA4,15 ;16995          MOV WR[0] TO LS[DATA.2]
U 184D, 3EA2,15 ;16996          MOV WR[0] TO LS[DATA.1]
U 184E, C550,15 ;16997          BIC LS[BIT8] TO WR[0]
U 184F, BEA6,15 ;16998          MOV WR[0] TO LS[DATA.3]
U 1850, 887C,6C ;16999          JSR [CHECK.SUB]               ;Check for error
U 1851, 0983,D4 ;17000          JMP [T26.A]                   ;Loop on error
U 1852, FF82,15 ;17001          INC LS[ERROR.NUMBER]          ;Go onto error 13
U 1853, 8870,3C ;17002          JSR [LO]                      ;Get address of SHF LEFT FWR[FT0] AND
U 1854, 3E1A,15 ;17003          MOV WR[0] TO LS[T13]          ; FQ IN[ZERO] and EWR[ET0] AND EQ IN
                ;17004                                        ; [FRAC]
U 1855, 4555,95 ;17005          BIC LS[BIT10] TO WR[3]        ;Reset error mask
U 1856, 3648,15 ;17006  T26.B:  MOV LS[#10] TO WR[0]          ;Set error # to 13
U 1857, 4742,15 ;17007          BIS LS[BIT1] TO WR[0]
U 1858, C740,15 ;17008          BIS LS[BIT0] TO WR[0]
U 1859, BE82,15 ;17009          MOV WR[0] TO LS[ERROR.NUMBER]
U 185A, B69E,15 ;17010          MOV LS[ONES] TO WR[0]         ;Setup data
U 185B, 3EA4,15 ;17011          MOV WR[0] TO LS[DATA.2]
U 185C, BEA6,15 ;17012          MOV WR[0] TO LS[DATA.3]
U 185D, 454C,15 ;17013          BIC LS[BIT6] TO WR[0]
U 185E, 3EA2,15 ;17014          MOV WR[0] TO LS[DATA.1]
U 185F, 0878,FC ;17015          JSR [LOAD.TRIPLE]             ;Load data into FT0 and ET0
U 1860, B790,15 ;17016          MOV LS[TC8] TO WR[0]          ;Setup to MOV FT0 AND ET0 TO FQ AND EQ
U 1861, B716,95 ;17017          MOV LS[#300] TO WR[1]         ;Setup to Loop self
U 1862, 90F6,15 ;17018          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 AND ET0 TO FQ AND EQ
U 1863, 10F6,95 ;17019          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 1864, B61A,15 ;17020          MOV LS[T13] TO WR[0]          ;Setup to SHF LEFT FT0, ET0, FQ, and EQ
                ;17021                                        ; IN[ZERO] AND IN[FRAC]
U 1865, 90F6,15 ;17022          MISC2 [TRAP.ACC] WR[0]        ;SHF LEFT FT0, FQ IN[ZERO] AND ET0, EQ
                ;17023                                        ; IN[FRAC]
U 1866, 90F6,15 ;17024          MISC2 [TRAP.ACC] WR[0]        ;SHF LEFT FT0, FQ IN[ZERO] AND ET0, EQ
                ;17025                                        ; IN[FRAC]
U 1867, 10F6,95 ;17026          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 1868, 087B,5C ;17027          JSR [STORE.0.TRIPLE]          ;Store FT0 and ET0 to CPU
U 1869, B69E,15 ;17028          MOV LS[ONES] TO WR[0]         ;Setup data
```

```
U 186A, 3EA4,15 ;17029          MOV WR[0] TO LS[DATA.2]
U 186B, C54E,15 ;17030          BIC LS[BIT7] TO WR[0]
U 186C, 3EA2,15 ;17031          MOV WR[0] TO LS[DATA.1]
U 186D, 474E,15 ;17032          BIS LS[BIT7] TO WR[0]
U 186E, C550,15 ;17033          BIC LS[BIT8] TO WR[0]
U 186F, 4552,15 ;17034          BIC LS[BIT9] TO WR[0]
U 1870, BEA6,15 ;17035          MOV WR[0] TO LS[DATA.3]
U 1871, 887C,6C ;17036          JSR [CHECK.SUB]                 ;Check for error
U 1872, 8985,64 ;17037          JMP [T26.B]                     ;Loop on error
U 1873, FF82,15 ;17038          INC LS[ERROR.NUMBER]            ;Go onto error 14
U 1874, 3792,15 ;17039          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ TO FT0
U 1875, B716,95 ;17040          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1876, 90F6,15 ;17041          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT0
U 1877, 10F6,95 ;17042          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1878, 087B,5C ;17043          JSR [STORE.0.TRIPLE]            ;Store FT0 and ET0 to the CPU
U 1879, 887C,6C ;17044          JSR [CHECK.SUB]                 ;Check for error
U 187A, 8985,64 ;17045          JMP [T26.B]                     ;Loop on error
U 187B, FF82,15 ;17046          INC LS[ERROR.NUMBER]            ;Go onto error 15
U 187C, 8870,3C ;17047          JSR [LO]                        ;Get address of MOV FT0 TO FT6
U 187D, 3E1C,15 ;17048          MOV WR[0] TO LS[T14]
U 187E, 8870,3C ;17049          JSR [LO]                        ;Get address of STORE FIRST FLT FT6
U 187F, 3EAE,15 ;17050          MOV WR[0] TO LS[T57]
U 1880, 8870,3C ;17051          JSR [LO]                        ;Get address of STORE SEC FLT FT6
U 1881, 3EB0,15 ;17052          MOV WR[0] TO LS[T58]
U 1882, 8870,3C ;17053          JSR [LO]                        ;Get address of STORE THIRD FLT FT6
U 1883, BEB2,15 ;17054          MOV WR[0] TO LS[T59]
U 1884, 8870,3C ;17055          JSR [LO]                        ;Get address of SHF LEFT FT6, ET6
U 1885, 3E1A,15 ;17056          MOV WR[0] TO LS[T13]            ; IN[DIV.SHF] AND FQ, EQ IN[HUGE.FRAC]
U 1886, 3648,15 ;17057  T26.C:  MOV LS[#10] TO WR[0]            ;Set error # to 15
U 1887, 4744,15 ;17058          BIS LS[BIT2] TO WR[0]
U 1888, C740,15 ;17059          BIS LS[BIT0] TO WR[0]
U 1889, BE82,15 ;17060          MOV WR[0] TO LS[ERROR.NUMBER]
U 188A, B69E,15 ;17061          MOV LS[ONES] TO WR[0]           ;Setup data
U 188B, 3EA4,15 ;17062          MOV WR[0] TO LS[DATA.2]
U 188C, BEA6,15 ;17063          MOV WR[0] TO LS[DATA.3]
U 188D, 454C,15 ;17064          BIC LS[BIT6] TO WR[0]
U 188E, 3EA2,15 ;17065          MOV WR[0] TO LS[DATA.1]
U 188F, 0878,FC ;17066          JSR [LOAD.TRIPLE]               ;Load data into FT0 and ET0
U 1890, 0878,AC ;17067          JSR [MOV.DATA]                  ;MOV FT0 TO FT6
U 1891, B790,15 ;17068          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 AND ET0 TO FQ AND EQ
U 1892, B716,95 ;17069          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1893, 90F6,15 ;17070          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 AND ET0 TO FQ AND EQ
U 1894, 10F6,95 ;17071          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1895, B61A,15 ;17072          MOV LS[T13] TO WR[0]            ;Setup to SHF LEFT FT6, ET6, FQ, and EQ
                ;17073                                          ; IN[DIV.SHF] AND IN[HUGE.FRAC]
U 1896, 90F6,15 ;17074          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT FT6, FQ IN[DIV.SHF] AND ET6,
                ;17075                                          ; EQ IN[HUGE.FRAC]
U 1897, 90F6,15 ;17076          MISC2 [TRAP.ACC] WR[0]          ;SHF LEFT FT6, FQ IN[DIV.SHF] AND ET6,
                ;17077                                          ; EQ IN[HUGE.FRAC]
U 1898, 10F6,95 ;17078          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1899, 087A,BC ;17079          JSR [STORE.X.TRIPLE]            ;Store FT6 and ET6 to CPU
U 189A, B69E,15 ;17080          MOV LS[ONES] TO WR[0]           ;Setup data
U 189B, 3EA4,15 ;17081          MOV WR[0] TO LS[DATA.2]
U 189C, C54E,15 ;17082          BIC LS[BIT7] TO WR[0]
U 189D, 3EA2,15 ;17083          MOV WR[0] TO LS[DATA.1]
```

```
U 189E, 474E,15 ;17084          BIS LS[BIT7] TO WR[0]
U 189F, C550,15 ;17085          BIC LS[BIT8] TO WR[0]
U 18A0, BEA6,15 ;17086          MOV WR[0] TO LS[DATA.3]
U 18A1, 887C,6C ;17087          JSR [CHECK.SUB]                 ;Check for error
U 18A2, 0988,64 ;17088          JMP [T26.C]                     ;Loop on error
U 18A3, FF82,15 ;17089          INC LS[ERROR.NUMBER]            ;Go onto error 16
U 18A4, 3792,15 ;17090          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ, EQ TO FTO, ETO
U 18A5, B716,95 ;17091          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 18A6, 90F6,15 ;17092          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ, EQ TO FTO, ETO
U 18A7, 10F6,95 ;17093          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 18A8, 087B,5C ;17094          JSR [STORE.0.TRIPLE]            ;Store FTO ETO to the CPU
U 18A9, 36A6,15 ;17095          MOV LS[DATA.3] TO WR[0]         ;Setup data
U 18AA, 4552,15 ;17096          BIC LS[BIT9] TO WR[0]
U 18AB, BEA6,15 ;17097          MOV WR[0] TO LS[DATA.3]
U 18AC, 887C,6C ;17098          JSR [CHECK.SUB]                 ;Check for error
U 18AD, 0988,64 ;17099          JMP [T26.C]                     ;Loop on error
                ;17100  T26.END:
```

```
;17101  .PAGE
;17102  .TOC    "TEST 27 SIGN CONTROL TEST(M8389 FPA MODULE)"
;17103  ;++
;17104  ;
;17105  ;
;17106  ; Test Description:
;17107  ;
;17108  ;       The purpose of this test isto check the sign control logic. Most of the
;17109  ;       sign control logic is located in the SIGN CTL PAL on page C. The data
;17110  ;       path is as follows: a floating point word will be loaded into the FPA
;17111  ;       and the sign of the word will be clocked into FPAC OP1 SIGN (1) H if
;17112  ;       the clock control field is 111 and the mod field is not EXTEND CLK.
;17113  ;       On the next cycle the sign of first operand will be clocked into  SIGN
;17114  ;       OUT if the clock field is 100 and the mod field is not extend clock and
;17115  ;       the A Address lines are 000. What is clocked into SIGN OUT depends on
;17116  ;       the lower three bits of the A Address lines. These eight possibilities
;17117  ;       will be checked in this test. The basic data path will take 3 cyles.
;17118  ;       One cycle to load the data into OP1 SIGN or OP2 SIGN, a second cycle to
;17119  ;       load the data into SIGN OUT, and a third cycle to load the SIGN OUT
;17120  ;       to bit 15 of the FPA BUS to be read back to the CPU.
;17121  ;
;17122  ; Assumptions:
;17123  ;
;17124  ;       It is assumed that the condition code logic has been checked and works.
;17125  ;
;17126  ; Test Steps:
;17127  ;
;17128  ;       1) Issue a MISC instruction to force a nop with the clock field set to
;17129  ;          4 and the mod field set to nop and 0100 on the A Address lines. This
;17130  ;          should load a zero into SIGN OUT. Issue a MOV instruction to load
;17131  ;          SIGN OUT into bit 15 of the FPA BUS and into the CPU. If bit 15 is
;17132  ;          not zero then issue error 1. Repeat this with the A Address lines
;17133  ;          set to 0110. Issue error 1 if bit 15 is not zero.
;17134  ;       2) Issue a MISC instruction to force a nop with the clock field set to
;17135  ;          4 and the mod field set to nop and the A Address lines set to 0101.
;17136  ;          This should load a one into SIGN OUT. Issue a MOV instruction to
;17137  ;          load SIGN OUT into bit 15 of the FPA BUS and into the CPU. If bit 15
;17138  ;          is not one then, issue error 2. Repeat this with the A Address lines
;17139  ;          set to 0111.
;17140  ;       3) Issue a MISC instrution to load the first float of data with the
;17141  ;          clock field set to 111 and the mod field set to nop. This will load
;17142  ;          OP1 SIGN with whatever is in bit 15 of the FPA BUS. In this case a
;17143  ;          one will be loaded into OP1 SIGN. Issue a MISC instruction to force
;17144  ;          a nop with the clock field set to 4 and the mod field set to nop and
;17145  ;          the A Address lines set to 0000. This should load OP1 SIGN into SIGN
;17146  ;          OUT. Issue a MOV instruction to load SIGN OUT into bit 15 of the
;17147  ;          FPA BUS and into the CPU. If bit 15 is not one then issue error 3.
;17148  ;       4) Repeat step 3 except load OP1 SIGN with zero. If bit 15 is not zero
;17149  ;          then issue error 4.
;17150  ;       5) Issue a MISC instrution to load the first float of data with the
;17151  ;          clock field set to 101 and the mod field set to nop. This will load
;17152  ;          OP2 SIGN with whatever is in bit 15 of the FPA BUS. In this case a
;17153  ;          one will be loaded into OP2 SIGN. Issue a MISC instruction to force
;17154  ;          a nop with the clock field set to 4 and the mod field set to nop and
;17155  ;          the A Address lines set to 0001. This should load OP2 SIGN into
```

```
;17156  ;          SIGN OUT. Issue a MOV instruction to load SIGN OUT into bit 15 of
;17157  ;          the FPA BUS and into the CPU. If bit 15 is not one then issue
;17158  ;          error 5.
;17159  ;       6) Repeat step 5 except load OP2 SIGN with zero. If bit 15 is not zero
;17160  ;          then issue error 6.
;17161  ;       7) Issue a MISC instrution to load the first float of data with the
;17162  ;          clock field set to 111 and the mod field set to nop. This will load
;17163  ;          OP1 SIGN with whatever is in bit 15 of the FPA BUS. In this case a
;17164  ;          one will be loaded into OP1 SIGN. Issue a MISC instrution to load
;17165  ;          the first float of data with the clock field set to 101 and the mod
;17166  ;          field set to nop. This will load OP2 SIGN with whatever is in bit 15
;17167  ;          of the FPA BUS. In this case a one will be loaded into OP2 SIGN.
;17168  ;          Issue a MISC instruction to force a nop with the clock field set to
;17169  ;          4 and the mod field set to nop and the A Address lines set to 0010.
;17170  ;          This will cause the XOR of OP1 SIGN and OP2 SIGN to be loaded into
;17171  ;          SIGN OUT. In this case the XOR of OP1 SIGN and OP2 SIGN is 0.
;17172  ;          Issue a MOV instruction to load SIGN OUT into bit 15 of
;17173  ;          the FPA BUS and into the CPU. If bit 15 is not zero then issue
;17174  ;          error 7.
;17175  ;       8) Repeat step 7 except load OP1 SIGN with 0 and OP2 SIGN with 1. If
;17176  ;          bit 15 is not one then issue error 8.
;17177  ;       9) Repeat step 7 except load OP1 SIGN with 1 and OP2 SIGN with 0. If
;17178  ;          bit 15 is not one then issue error 9.
;17179  ;      10) Repeat step 7 except load OP1 SIGN with 0 and OP2 SIGN with 0. If
;17180  ;          bit 15 is not zero then issue error A.
;17181  ;      11) Issue a MISC instruction to load the first float of data with the
;17182  ;          clock field set to 111 and the mod field set to nop. This will load
;17183  ;          OP1 SIGN with whatever is in bit 15 of the FPA BUS. In this case a
;17184  ;          one will be loaded into OP1 SIGN. Issue a MISC instruction to force
;17185  ;          a nop with the clock field set to 4 and the mod field set to nop and
;17186  ;          the A Address lines set to 0110. This will cause SIGN OUT to be
;17187  ;          loaded with 0.Issue a MISC instruction to force a nop with the clock
;17188  ;          field set to 4 and the mod field set to nop and the A Address lines
;17189  ;          set to 0011. This will cause the XOR of OP1 SIGN and SIGN OUT to be
;17190  ;          loaded into SIGN OUT. In this case the XOR of OP1 SIGN and SIGN OUT
;17191  ;          is 1. Issue a MOV instruction to load SIGN OUT into bit 15 of the
;17192  ;          FPA BUS and into the CPU. If bit 15 is not 1 then issue error B.
;17193  ;      12) Repeat step 11 except load OP1 SIGN with 0. If bit 15 is not 0 then
;17194  ;          issue error C.
;17195  ;      13) Issue a MISC instruction to load the first float of data with the
;17196  ;          clock field set to 111 and the mod field set to nop. This will load
;17197  ;          OP1 SIGN with whatever is in bit 15 of the FPA BUS. In this case a
;17198  ;          one will be loaded into OP1 SIGN. Issue a MISC instruction to force
;17199  ;          a nop with the clock field set to 4 and the mod field set to nop and
;17200  ;          the A Address lines set to 0111. This will cause SIGN OUT to be
;17201  ;          loaded with 1.Issue a MISC instruction to force a nop with the clock
;17202  ;          field set to 4 and the mod field set to nop and the A Address lines
;17203  ;          set to 0011. This will cause the XOR of OP1 SIGN and SIGN OUT to be
;17204  ;          loaded into SIGN OUT. In this case the XOR of OP1 SIGN and SIGN OUT
;17205  ;          is 0. Issue a MOV instruction to load SIGN OUT into bit 15 of the
;17206  ;          FPA BUS and into the CPU. If bit 15 is not 0 then issue error D.
;17207  ;      14) Repeat step 13 except load OP1 SIGN with 0. If bit 15 is not 1 then
;17208  ;          issue error E.
;17209  ;
;17210  ; Error:
```

```
;17211  ;
;17212  ;       Error1 - SIGN OUT failed unasserted when the A Address lines were 0100
;17213  ;                or 0110.
;17214  ;       Error2 - SIGN OUT failed asserted when the A Address lines were 0101 or
;17215  ;                0111.
;17216  ;       Error3 - SIGN OUT failed asserted when the A Address lines were 0000
;17217  ;                and OP1 SIGN was asserted.
;17218  ;       Error4 - SIGN OUT failed unasserted when the A Address lines were 0000
;17219  ;                and OP1 SIGN was unasserted.
;17220  ;       Error5 - SIGN OUT failed asserted when the A Address lines were 0001
;17221  ;                and OP2 SIGN was asserted.
;17222  ;       Error6 - SIGN OUT failed unasserted when the A Address lines were 0001
;17223  ;                and OP2 SIGN was unasserted.
;17224  ;       Error7 - SIGN OUT failed unasserted when the A Address lines were 0010
;17225  ;                and OP1 SIGN and OP2 SIGN were asserted.
;17226  ;       Error8 - SIGN OUT failed asserted when the A Address lines were 0010
;17227  ;                and OP1 SIGN was unasserted and OP2 SIGN was asserted.
;17228  ;       Error9 - SIGN OUT failed asserted when the A Address lines were 0010
;17229  ;                and OP1 SIGN was asserted and OP2 SIGN was unasserted.
;17230  ;       ErrorA - SIGN OUT failed unasserted when the A Address lines were 0010
;17231  ;                and OP1 SIGN and OP2 SIGN were unasserted.
;17232  ;       ErrorB - SIGN OUT failed asserted when the A Address lines were 0110
;17233  ;                and OP1 SIGN was asserted and SIGN OUT was unasserted.
;17234  ;       ErrorC - SIGN OUT failed unasserted when the A Address lines were 0110
;17235  ;                and OP1 SIGN and SIGN OUT were unasserted.
;17236  ;       ErrorD - SIGN OUT failed unasserted when the A Address lines were 0110
;17237  ;                and OP1 SIGN and SIGN OUT were asserted.
;17238  ;       ErrorE - SIGN OUT failed asserted when the A Address lines were 0110
;17239  ;                and OP1 SIGN was unasserted and SIGN OUT was asserted.
;17240  ;
;17241  ; Debug:
;17242  ;
;17243  ;       Error1: If this error occurs then the SIGN CTL PAL should be checked
;17244  ;               for error with respect to SIGN OUT. If the PAL is not the cause
;17245  ;               of error then the A Address lines should be checked to be 0100
;17246  ;               of 0110 and ENB CLK4 should be checked to be asserted. If the
;17247  ;               A Address lines are not the cause of the error then the dirvers
;17248  ;               and the latches they come from on page D should be checked for
;17249  ;               error. If ENB CLK4 is in error then the CLOCK DECODER should be
;17250  ;               checked for error. If the DECODER is not the cause of the error
;17251  ;               then the Clock Ctl signals should be checked for error. If they
;17252  ;               are in error then the latches they come from should be checked
;17253  ;               for error.
;17254  ;       Error2: Same as error one except that the A Address lines should be
;17255  ;               0101 or 0111.
;17256  ;       Error3: There are three cycles that are required to be executed to test
;17257  ;               for this error. There are failures that can occur in two of
;17258  ;               the three cycles that will cause this error. In the first cycle
;17259  ;               OP1 SIGN is loaded from the FPA BUS (bit 15). The errors that
;17260  ;               could occur in this cycle are the failure of the SIGN CTL PAL
;17261  ;               with respect to the loading of OP1 SIGN. If the PAL isn't res-
;17262  ;               ponsible for the error then ENB CLK7 should be checked to be
;17263  ;               asserted. The second cycle is reading OP1 SIGN into SIGN OUT.
;17264  ;               If a failure occurs in this cycle then the SIGN CTL PAL should
;17265  ;               be checked for error. If the PAL is not the cause of error then
```

```
                ;17266  ;               the A Address lines should be checked to be 0000.
                ;17267  ;       Error4: Same as error 3 except that OP1 SIGN is loaded with zero
                ;17268  ;               instead of one.
                ;17269  ;       Error5: There are three cycles that are required to be executed to test
                ;17270  ;               for this error. There are failures that can occur in two of
                ;17271  ;               the three cycles that will cause this error. In the first cycle
                ;17272  ;               OP2 SIGN is loaded from the FPA BUS (bit 15). The errors that
                ;17273  ;               could occur in this cycle are the failure of the SIGN CTL PAL
                ;17274  ;               with respect to the loading of OP2 SIGN. If the PAL isn't res-
                ;17275  ;               ponsible for the error then ENB CLK5 should be checked to be
                ;17276  ;               asserted. The second cycle is reading OP2 SIGN into SIGN OUT.
                ;17277  ;               If a failure occurs in this cycle then the SIGN CTL PAL should
                ;17278  ;               be checked for error. If the PAL is not the cause of error then
                ;17279  ;               the A Address lines should be checked to be 0001.
                ;17280  ;       Error6: Same as error 5 except that OP2 SIGN is loaded with a zero
                ;17281  ;               instead of one.
                ;17282  ;       Error7: It takes four clock cycles to set up for this error. The first
                ;17283  ;               two and the last clock cycle have been tested previously in
                ;17284  ;               this test. If the error does occur is probably the result of
                ;17285  ;               an error in the third clock cycle. At the end of the third
                ;17286  ;               cycle the SIGN OUT should be unasserted. If there is an error
                ;17287  ;               then the SIGN CTL PAL should be checked for error. If the PAL
                ;17288  ;               is not the cause of error then OP1 SIGN and OP2 SIGN should
                ;17289  ;               be checked to be 1. It is unlikely that either of these signals
                ;17290  ;               are in error but if they are then see the debug section for
                ;17291  ;               error 5 and 3.
                ;17292  ;       Error8: Same as error 7 except that OP1 SIGN is unasserted and OP2 SIGN
                ;17293  ;               is asserted.
                ;17294  ;       Error9: Same as error 7 except that OP1 SIGN is asserted and OP2 SIGN
                ;17295  ;               is unasserted.
                ;17296  ;       ErrorA: Same as error 7 except that OP1 SIGN and OP2 SIGN are asserted
                ;17297  ;       ErrorB: It takes four clock cycles to set up for this error. The first
                ;17298  ;               two and the last clock cycle have been tested previously in
                ;17299  ;               this test. If the error does occur is probably the result of
                ;17300  ;               an error in the third clock cycle. At the end of the third
                ;17301  ;               cycle the SIGN OUT should be asserted. If there is an error
                ;17302  ;               then the SIGN CTL PAL should be checked for error. If the PAL
                ;17303  ;               is not the cause of error then OP1 SIGN should be asserted and
                ;17304  ;               SIGN OUT should unasserted. It is unlikely that either of
                ;17305  ;               these signals are in error but if they are then see the debug
                ;17306  ;               section for error 5 and 3.
                ;17307  ;       ErrorC: Same as error 11 except that OP1 SIGN and SIGN OUT should be
                ;17308  ;               unasserted.
                ;17309  ;       ErrorD: Same as error 11 except that OP1 SIGN and SIGN OUT should be
                ;17310  ;               asserted.
                ;17311  ;       ErrorE: Same as error 11 except that OP1 SIGN should be unasserted and
                ;17312  ;               SIGN OUT should be asserted.
                ;17313  ;
                ;17314  ;
                ;17315  ;
                ;17316  ;--
                ;17317  T.27:
U 18AE, B65E,15 ;17318          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;17319                                          ; STATUS WORD
U 18AF, 3E80,15 ;17320          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
```

```
                 ;17321                                  ; BIT 15 INDICATES START OF TEST TO
                 ;17322                                  ; CONSOLE
U 18B0, 10E0,15  ;17323          MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE
                 ;17324  WAIT.T27.0:
U 18B1, 898B,14  ;17325          JMP [WAIT.T27.0]        ;LOOP HERE WAITING FOR CONSOLE TO
                 ;17326                                  ; RESPOND
U 18B2, 087E,6C  ;17327          JSR [SETUP]             ;SET UP ERROR INFO AND CLEAR INSTRUC
                 ;17328                                  ; AND SIZE BITS
U 18B3, B69E,15  ;17329          MOV LS[ONES] TO WR[0]   ;Setup error mask
U 18B4, 455E,15  ;17330          BIC LS[BIT15] TO WR[0]
U 18B5, 3E8A,15  ;17331          MOV WR[0] TO LS[ERROR.MASK]
U 18B6, 37AE,15  ;17332          MOV LS[T27.POINTER] TO WR[0]  ;Initialize pointer for main memory
U 18B7, BE12,15  ;17333          MOV WR[0] TO LS[T9]
U 18B8, 8870,3C  ;17334          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH[0]
U 18B9, 3E10,15  ;17335          MOV WR[0] TO LS[T8]
U 18BA, 8870,3C  ;17336          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH[1]
U 18BB, BE14,15  ;17337          MOV WR[0] TO LS[T10]
U 18BC, 8870,3C  ;17338          JSR [L0]                ;Get address of CLOCK OP1 SIGN
U 18BD, 3E16,15  ;17339          MOV WR[0] TO LS[T11]
U 18BE, 8870,3C  ;17340          JSR [L0]                ;Get address of CLOCK OP2 SIGN
U 18BF, BE18,15  ;17341          MOV WR[0] TO LS[T12]
U 18C0, 8870,3C  ;17342          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH[OP1[
U 18C1, 3E1A,15  ;17343          MOV WR[0] TO LS[T13]
U 18C2, 8870,3C  ;17344          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH[OP2]
U 18C3, 3E1C,15  ;17345          MOV WR[0] TO LS[T14]
U 18C4, 8870,3C  ;17346          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH
U 18C5, BE1E,15  ;17347          MOV WR[0] TO LS[T15]    ; [OP1.XOR.OP2]
U 18C6, 8870,3C  ;17348          JSR [L0]                ;Get address of CLOCK SIGN OUT WITH
U 18C7, BF8C,15  ;17349          MOV WR[0] TO LS[TC5]    ; [SO.XOR.OP1]
U 18C8, B610,15  ;17350  T27.1:  MOV LS[T8] TO WR[0]     ;Setup to CLOCK SIGN OUT WITH [0]
U 18C9, B716,95  ;17351          MOV LS[#300] TO WR[1]   ;Setup to Loop self
U 18CA, 90F6,15  ;17352          MISC2 [TRAP.ACC] WR[0]  ;CLOCK SIGN OUT WITH [0]
U 18CB, 10F6,95  ;17353          MISC2 [TRAP.ACC] WR[1]  ;Loop self
U 18CC, 087B,5C  ;17354          JSR [STORE.0.TRIPLE]    ;Store FT0 and sign bit to the CPU
U 18CD, B6A8,15  ;17355          MOV LS[FIRST.FLT] TO WR[0]  ;Setup received data
U 18CE, 2F85,15  ;17356          CLR WR[2]               ;setup expected data
U 18CF, 0869,3C  ;17357          JSR [CHECK.RESULT]      ;Check for error
U 18D0, 098C,84  ;17358          JMP [T27.1]             ;Loop on error
U 18D1, FF82,15  ;17359          INC LS[ERROR.NUMBER]    ;Go onto error 2
U 18D2, 3614,15  ;17360  T27.2:  MOV LS[T10] TO WR[0]    ;Setup to CLOCK SIGN OUT WITH [1]
U 18D3, B716,95  ;17361          MOV LS[#300] TO WR[1]   ;Setup to Loop self
U 18D4, 90F6,15  ;17362          MISC2 [TRAP.ACC] WR[0]  ;CLOCK SIGN OUT WITH [1]
U 18D5, 10F6,95  ;17363          MISC2 [TRAP.ACC] WR[1]  ;Loop self
U 18D6, 087B,5C  ;17364          JSR [STORE.0.TRIPLE]    ;Store FT0 and sign bit to the CPU
U 18D7, B6A8,15  ;17365          MOV LS[FIRST.FLT] TO WR[0]  ;Setup received data
U 18D8, 365E,95  ;17366          MOV LS[BIT15] TO WR[1]  ;setup expected data
U 18D9, 0869,3C  ;17367          JSR [CHECK.RESULT]      ;Check for error
U 18DA, 898D,24  ;17368          JMP [T27.2]             ;Loop on error
U 18DB, FF82,15  ;17369          INC LS[ERROR.NUMBER]    ;Go onto error 3
U 18DC, B616,15  ;17370  T27.3:  MOV LS[T11] TO WR[0]    ;Setup to CLOCK OP1 SIGN
U 18DD, B776,95  ;17371          MOV LS[#8300] TO WR[1]  ;Setup to Loop self
U 18DE, 90F6,15  ;17372          MISC2 [TRAP.ACC] WR[0]  ;CLOCK OP1 SIGN
U 18DF, 10F6,95  ;17373          MISC2 [TRAP.ACC] WR[1]  ;Enable data onto bus
U 18E0, B61A,15  ;17374          MOV LS[T13] TO WR[0]    ;Setup to CLOCK SIGN OUT WITH [OP1]
U 18E1, B716,95  ;17375          MOV LS[#300] TO WR[1]   ;Setup to Loop self
```

```
U 18E2, 90F6,15 ;17376          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH [OP1]
U 18E3, 10F6,95 ;17377          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 18E4, 087B,5C ;17378          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 18E5, B6A8,15 ;17379          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 18E6, 365E,95 ;17380          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 18E7, 0869,3C ;17381          JSR [CHECK.RESULT]              ;Check for error
U 18E8, 098D,C4 ;17382          JMP [T27.3]                     ;Loop on error
U 18E9, FF82,15 ;17383          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 18EA, B616,15 ;17384  T27.4:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 18EB, B716,95 ;17385          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 18EC, 90F6,15 ;17386          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 18ED, 10F6,95 ;17387          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 18EE, B61A,15 ;17388          MOV LS[T13] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH [OP1]
U 18EF, B716,95 ;17389          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 18F0, 90F6,15 ;17390          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH [OP1]
U 18F1, 10F6,95 ;17391          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 18F2, 087B,5C ;17392          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 18F3, B6A8,15 ;17393          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 18F4, 2F82,95 ;17394          CLR WR[1]                       ;Setup expected data
U 18F5, 0869,3C ;17395          JSR [CHECK.RESULT]              ;Check for error
U 18F6, 098E,A4 ;17396          JMP [T27.4]                     ;Loop on error
U 18F7, FF82,15 ;17397          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 18F8, 3618,15 ;17398  T27.5:  MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
U 18F9, B776,95 ;17399          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 18FA, 90F6,15 ;17400          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 18FB, 10F6,95 ;17401          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 18FC, B61C,15 ;17402          MOV LS[T14] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH [OP2]
U 18FD, B716,95 ;17403          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 18FE, 90F6,15 ;17404          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH [OP2]
U 18FF, 10F6,95 ;17405          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1900, 087B,5C ;17406          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 1901, B6A8,15 ;17407          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1902, 365E,95 ;17408          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 1903, 0869,3C ;17409          JSR [CHECK.RESULT]              ;Check for error
U 1904, 098F,84 ;17410          JMP [T27.5]                     ;Loop on error
U 1905, FF82,15 ;17411          INC LS[ERROR.NUMBER]            ;Go onto error 6
U 1906, 3618,15 ;17412  T27.6:  MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
U 1907, B716,95 ;17413          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1908, 90F6,15 ;17414          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 1909, 10F6,95 ;17415          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 190A, B61C,15 ;17416          MOV LS[T14] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH [OP2]
U 190B, B716,95 ;17417          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 190C, 90F6,15 ;17418          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH [OP2]
U 190D, 10F6,95 ;17419          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 190E, 087B,5C ;17420          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 190F, B6A8,15 ;17421          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1910, 2F82,95 ;17422          CLR WR[1]                       ;Setup expected data
U 1911, 0869,3C ;17423          JSR [CHECK.RESULT]              ;Check for error
U 1912, 0990,64 ;17424          JMP [T27.6]                     ;Loop on error
U 1913, FF82,15 ;17425          INC LS[ERROR.NUMBER]            ;Go onto error 7
U 1914, B616,15 ;17426  T27.7:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 1915, B776,95 ;17427          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 1916, 90F6,15 ;17428          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 1917, 10F6,95 ;17429          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1918, 3618,15 ;17430          MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
```

```
U 1919, B776,95 ;17431          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 191A, 90F6,15 ;17432          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 191B, 10F6,95 ;17433          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 191C, 361E,15 ;17434          MOV LS[T15] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 191D, 90F6,15 ;17435          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 191E, 10F6,95 ;17436          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 191F, 087B,5C ;17437          JSR [STORE.0.TRIPLE]            ;Store FT0 and sign bit to the CPU
U 1920, B6A8,15 ;17438          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1921, 2F82,95 ;17439          CLR WR[1]                       ;Setup expected data
U 1922, 0869,3C ;17440          JSR [CHECK.RESULT]              ;Check for error
U 1923, 0991,44 ;17441          JMP [T27.7]                     ;Loop on error
U 1924, FF82,15 ;17442          INC LS[ERROR.NUMBER]            ;Go onto error 8
U 1925, B616,15 ;17443  T27.8:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 1926, B716,95 ;17444          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1927, 90F6,15 ;17445          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 1928, 10F6,95 ;17446          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1929, 3618,15 ;17447          MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
U 192A, B776,95 ;17448          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 192B, 90F6,15 ;17449          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 192C, 10F6,95 ;17450          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 192D, 361E,15 ;17451          MOV LS[T15] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 192E, 90F6,15 ;17452          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 192F, 10F6,95 ;17453          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1930, 087B,5C ;17454          JSR [STORE.0.TRIPLE]            ;Store FT0 and sign bit to the CPU
U 1931, B6A8,15 ;17455          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1932, 365E,95 ;17456          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 1933, 0869,3C ;17457          JSR [CHECK.RESULT]              ;Check for error
U 1934, 8992,54 ;17458          JMP [T27.8]                     ;Loop on error
U 1935, FF82,15 ;17459          INC LS[ERROR.NUMBER]            ;Go onto error 9
U 1936, B616,15 ;17460  T27.9:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 1937, B776,95 ;17461          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 1938, 90F6,15 ;17462          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 1939, 10F6,95 ;17463          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 193A, 3618,15 ;17464          MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
U 193B, B716,95 ;17465          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 193C, 90F6,15 ;17466          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 193D, 10F6,95 ;17467          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 193E, 361E,15 ;17468          MOV LS[T15] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 193F, 90F6,15 ;17469          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 1940, 10F6,95 ;17470          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1941, 087B,5C ;17471          JSR [STORE.0.TRIPLE]            ;Store FT0 and sign bit to the CPU
U 1942, B6A8,15 ;17472          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1943, 365E,95 ;17473          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 1944, 0869,3C ;17474          JSR [CHECK.RESULT]              ;Check for error
U 1945, 0993,64 ;17475          JMP [T27.9]                     ;Loop on error
U 1946, FF82,15 ;17476          INC LS[ERROR.NUMBER]            ;Go onto error A
U 1947, B616,15 ;17477  T27.A:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 1948, B716,95 ;17478          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1949, 90F6,15 ;17479          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 194A, 10F6,95 ;17480          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 194B, 3618,15 ;17481          MOV LS[T12] TO WR[0]            ;Setup to CLOCK OP2 SIGN
U 194C, B716,95 ;17482          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 194D, 90F6,15 ;17483          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 194E, 10F6,95 ;17484          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 194F, 361E,15 ;17485          MOV LS[T15] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[OP1.XOR.OP2]
```

```
U 1950, 90F6,15 ;17486          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[OP1.XOR.OP2]
U 1951, 10F6,95 ;17487          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1952, 087B,5C ;17488          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 1953, B6A8,15 ;17489          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1954, 2F82,95 ;17490          CLR WR[1]                       ;Setup expected data
U 1955, 0869,3C ;17491          JSR [CHECK.RESULT]              ;Check for error
U 1956, 0994,74 ;17492          JMP [T27.A]                     ;Loop on error
U 1957, FF82,15 ;17493          INC LS[ERROR.NUMBER]            ;Go onto error B
U 1958, B616,15 ;17494  T27.B:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 1959, B776,95 ;17495          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 195A, 90F6,15 ;17496          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 195B, 10F6,95 ;17497          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 195C, B610,15 ;17498          MOV LS[T8] TO WR[0]             ;Setup to CLOCK SIGN OUT WITH[0]
U 195D, B716,95 ;17499          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 195E, 90F6,15 ;17500          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[0]
U 195F, 10F6,95 ;17501          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1960, 378C,15 ;17502          MOV LS[TC5] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[SO.XOR.OP2]
U 1961, 90F6,15 ;17503          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[SO.XOR.OP1]
U 1962, 10F6,95 ;17504          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1963, 087B,5C ;17505          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 1964, B6A8,15 ;17506          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1965, 365E,95 ;17507          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 1966, 0869,3C ;17508          JSR [CHECK.RESULT]              ;Check for error
U 1967, 8995,84 ;17509          JMP [T27.B]                     ;Loop on error
U 1968, FF82,15 ;17510          INC LS[ERROR.NUMBER]            ;Go onto error C
U 1969, B616,15 ;17511  T27.C:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 196A, B716,95 ;17512          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 196B, 90F6,15 ;17513          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 196C, 10F6,95 ;17514          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 196D, B610,15 ;17515          MOV LS[T8] TO WR[0]             ;Setup to CLOCK SIGN OUT WITH[0]
U 196E, B716,95 ;17516          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 196F, 90F6,15 ;17517          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[0]
U 1970, 10F6,95 ;17518          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1971, 378C,15 ;17519          MOV LS[TC5] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[SO.XOR.OP2]
U 1972, 90F6,15 ;17520          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[SO.XOR.OP1]
U 1973, 10F6,95 ;17521          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1974, 087B,5C ;17522          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 1975, B6A8,15 ;17523          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1976, 2F82,95 ;17524          CLR WR[1]                       ;Setup expected data
U 1977, 0869,3C ;17525          JSR [CHECK.RESULT]              ;Check for error
U 1978, 0996,94 ;17526          JMP [T27.C]                     ;Loop on error
U 1979, FF82,15 ;17527          INC LS[ERROR.NUMBER]            ;Go onto error D
U 197A, B616,15 ;17528  T27.D:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 197B, B776,95 ;17529          MOV LS[#8300] TO WR[1]          ;Setup to Loop self
U 197C, 90F6,15 ;17530          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 197D, 10F6,95 ;17531          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 197E, 3614,15 ;17532          MOV LS[T10] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[1]
U 197F, B716,95 ;17533          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1980, 90F6,15 ;17534          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[1]
U 1981, 10F6,95 ;17535          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1982, 378C,15 ;17536          MOV LS[TC5] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[SO.XOR.OP2]
U 1983, 90F6,15 ;17537          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[SO.XOR.OP1]
U 1984, 10F6,95 ;17538          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1985, 087B,5C ;17539          JSR [STORE.0.TRIPLE]            ;Store FTO and sign bit to the CPU
U 1986, B6A8,15 ;17540          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
```

```
U 1987, 2F82,95 ;17541          CLR WR[1]                       ;Setup expected data
U 1988, 0869,3C ;17542          JSR [CHECK.RESULT]              ;Check for error
U 1989, 8997,A4 ;17543          JMP [T27.D]                     ;Loop on error
U 198A, FF82,15 ;17544          INC LS[ERROR.NUMBER]            ;Go onto error E
U 198B, B616,15 ;17545  T27.E:  MOV LS[T11] TO WR[0]            ;Setup to CLOCK OP1 SIGN
U 198C, B716,95 ;17546          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 198D, 90F6,15 ;17547          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 198E, 10F6,95 ;17548          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 198F, 3614,15 ;17549          MOV LS[T10] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[1]
U 1990, B716,95 ;17550          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1991, 90F6,15 ;17551          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[1]
U 1992, 10F6,95 ;17552          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 1993, 378C,15 ;17553          MOV LS[TC5] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[SO.XOR.OP2]
U 1994, 90F6,15 ;17554          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[SO.XOR.OP1]
U 1995, 10F6,95 ;17555          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1996, 087B,5C ;17556          JSR [STORE.0.TRIPLE]            ;Store FT0 and sign bit to the CPU
U 1997, B6A8,15 ;17557          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1998, 365E,95 ;17558          MOV LS[BIT15] TO WR[1]          ;Setup expected data
U 1999, 0869,3C ;17559          JSR [CHECK.RESULT]              ;Check for error
U 199A, 0998,B4 ;17560          JMP [T27.E]                     ;Loop on error
                ;17561  T27.END:
```

```
;17562  .PAGE
;17563  .TOC    "TEST 28 CC CONDITION TEST(M8389 FPA MODULE)"
;17564  ;++
;17565  ;
;17566  ;
;17567  ; Test Description:
;17568  ;
;17569  ;       The purpose of this test is to check the condition code logic
;17570  ;       which is primarily found in the CC CTL PAL. This will be done by
;17571  ;       putting known quantities in D00 - D03, and by setting up known condi-
;17572  ;       tions in the condition codes and reading the FPA to the CPU. The
;17573  ;       first two tests will check the conditions that the condition codes
;17574  ;       won't be enabled onto the FPA BUS. For these tests the condition
;17575  ;       codes will be set up to be one and bits D00 - D03 will be set to zero.
;17576  ;       When the condition codes are enabled onto the FPA BUS the data can be
;17577  ;       integer or floating point. Both of these cases wil be tested for the
;17578  ;       condition codes set and the condition codes clear.
;17579  ;
;17580  ; Assumptions:
;17581  ;
;17582  ;       All the logic of the 2901 should have been tested and be working.
;17583  ;
;17584  ; Test Steps:
;17585  ;
;17586  ;       1) Issue a MISC instruction to load bit 47 of the fraction data path
;17587  ;          with 1 and an integer instruction in the instruction encode bits.
;17588  ;          Issue a MISC instruction to force an instruction with the shift bits
;17589  ;          set to one and the clock bits to 6 and the mod field to nop. This
;17590  ;          will clock the CC bits onto bits 0 - 4 of the FPA BUS. Issue a MOV
;17591  ;          instruction to load the contents of the FPA BUS into the CPU. If
;17592  ;          bit 2 is clear and the rest are set then there is no error, other-
;17593  ;          wise issue error 1.
;17594  ;       2) Issue a MISC instruction with a floating point instruction on the
;17595  ;          instruction bits to load SIGN OUT with 1. Issue a MISC instruction
;17596  ;          with the shift bits set to 10 and the clock bits set to 6 and the
;17597  ;          mod field set to nop. This should clock the CC bits onto the FPA BUS
;17598  ;          A MOV instruction will load the FPA BUS into the CPU where the CC
;17599  ;          bits can be checked to be 1010. If they are not then issue error 2.
;17600  ;       3) Issue a MISC instruction with an integer instruction on the instruc-
;17601  ;          tion bits to load zeros into bits 16 - 47 of the fraction data path.
;17602  ;          Issue a MISC isntruction with shift bits set to 01 and the clock
;17603  ;          field set to 6 and the mod field set to nop. This will clock the CC
;17604  ;          bits onto the FPA BUS. A MOV instruction will load the FPA BUS into
;17605  ;          the CPU where the CC bits can be checked to be 0101. If they are not
;17606  ;          then issue error 3.
;17607  ;       4) Issue a MISC instruction with the instruction encode bits set to an
;17608  ;          floating point instruction to load the exponent data path with zeros
;17609  ;          Issue a second MISC instruction with the shift bits set to 00 and
;17610  ;          clock field set to 6 and the mod field set to nop. This will clock
;17611  ;          the CC bits onto the FPA BUS. A MOV instruction load the FPA BUS
;17612  ;          into the CPU where the CC bits can be checked to be 0100. If there's
;17613  ;          an error then issue error 4.
;17614  ;       5) Set the instruction bits to an integer instruction. Load the frac-
;17615  ;          tion data path with the sign bit (bit 47) clear and the LSB set.
;17616  ;          Force CLOCK CC. Then store the CC's to the CPU. All the cc's should
```

```
;17617  ;          be clear, if they aren't issue error 5.
;17618  ;       6) Repeat step 5 except set the instruction and size bits to 0. If all
;17619  ;          the cc's aren't clear then issue error 6.
;17620  ;
;17621  ; Error:
;17622  ;
;17623  ;       Error1 - Condition Codes failed when they should've been 1011.
;17624  ;       Error2 - Condition Codes failed when they should've been 1010.
;17625  ;       Error3 - Condition Codes failed when they should've been 0101.
;17626  ;       Error4 - Condition Codes failed when they should've been 0100.
;17627  ;       Error5 - Condition Codes failed when they should've been 0000 and the
;17628  ;                data type was integer.
;17629  ;       Error6 - Condition Codes failed when they should've been 0000 and the
;17630  ;                data type was floating.
;17631  ;
;17632  ; Debug:
;17633  ;
;17634  ;       Error1: If this error occurs then it's likely that the error occured
;17635  ;               in the second of three clock cycles. This is the cycle when the
;17636  ;               CC's are enabled onto the FPA BUS. If a failure occurs here it
;17637  ;               is likely to be a result the CC CTL PAL. If the PAL ia not the
;17638  ;               cause of the error then FRAC 47 F3 SAVE, FPAL ALTER INT, FPAC
;17639  ;               ENB CLK6, FPAE SHF0, FPAE SHF1, and FPAL ENB CC should be
;17640  ;               asserted. FPAD EXT CLK should be unasserted. If FRAC 47 F3 SAVE
;17641  ;               is in error then the STATUS REG 2 LATCH should be checked for
;17642  ;               error. IF FPAL ALTER INT is in error then the STORE CTL GATE
;17643  ;               should be checked for error. If the gate is not the cause of
;17644  ;               error then the INSTRUCTION PAL should be checked for error. If
;17645  ;               FPAC ENB CLK6 is in error then the clock decoder should be
;17646  ;               checked for error. If FPAL ENB CC is in error then the STORE
;17647  ;               CTL PAL should be checked for error. If EXT CLK is in error
;17648  ;               then the latch it came from should be checked for error. If the
;17649  ;               latch is not the cause of error then the SIGN CTL PAL should be
;17650  ;               checked for error.
;17651  ;       Error2: If this error occurs and the previous error didn't occur then
;17652  ;               it's likely that the CC CTL PAL is the cause of the error. If
;17653  ;               it isn't the cause of the error then SIGN OUT should be checked
;17654  ;               to be asserted and FPAL ALTER INT should be checked to be un-
;17655  ;               asserted. If SIGN OUT is in error then the etch between the
;17656  ;               CC CTL PAL and the SIGN CTL PAL should be checked for error. If
;17657  ;               FPAL ALTER INT is in error then the STORE CTL GATE should be
;17658  ;               checked for error. If the gate isn't the cause of error then
;17659  ;               the INSTRUCTION PAL should be checked for error.
;17660  ;       Error3: If this error occurs and the previous errors haven't occured
;17661  ;               then it's likekly that the CC CTL PAL is the cause of error. If
;17662  ;               it's not the cause of error then FRAC 47=0 and FRAC 31=0 should
;17663  ;               be checked to be asserted. If they aren't in error then check
;17664  ;               the etch between the CC TL PAL and the 2901's.
;17665  ;       Error4: If this error occurs and the previous errors haven't occured
;17666  ;               then it's likely that the CC CTL PAL is the cause of the error.
;17667  ;               If the PAL is not in error then FPAB EXP EQ 0 should be checked
;17668  ;               to be asserted. If FPAB EXP EQ 0 is in error then the BRANCH 3
;17669  ;               PAL should be checked for error. If the PAL is not the cause of
;17670  ;               error then the etch between the BRANCH 3 PAL and the least sig-
;17671  ;               nificant 2901 in the exponent data path should be checked for
```

```
                ;17672  ;               error.
                ;17673  ;       Error5: See errors 1 - 4.
                ;17674  ;       Error6: See errors 1 - 4.
                ;17675  ;
                ;17676  ;
                ;17677  ;--
                ;17678  T.28:
U 199B, B65E,15 ;17679          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;17680                                          ; STATUS WORD
U 199C, 3E80,15 ;17681          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;17682                                          ; BIT 15 INDICATES START OF TEST TO
                ;17683                                          ; CONSOLE
U 199D, 10E0,15 ;17684          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;17685  WAIT.T28.0:
U 199E, 8999,E4 ;17686          JMP [WAIT.T28.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;17687                                          ; RESPOND
U 199F, 087E,6C ;17688          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;17689                                          ; AND SIZE BITS
U 19A0, B69E,15 ;17690          MOV LS[ONES] TO WR[0]           ;Setup error mask
U 19A1, 4540,15 ;17691          BIC LS[BIT0] TO WR[0]
U 19A2, C542,15 ;17692          BIC LS[BIT1] TO WR[0]
U 19A3, C544,15 ;17693          BIC LS[BIT2] TO WR[0]
U 19A4, 4546,15 ;17694          BIC LS[BIT3] TO WR[0]
U 19A5, 3E8A,15 ;17695          MOV WR[0] TO LS[ERROR.MASK]
U 19A6, 37B0,15 ;17696          MOV LS[T28.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 19A7, BE12,15 ;17697          MOV WR[0] TO LS[T9]
U 19A8, 8870,3C ;17698          JSR [L0]                        ;Get address of CLOCK CC SET[V.C]
U 19A9, 3E10,15 ;17699          MOV WR[0] TO LS[T8]
U 19AA, 8870,3C ;17700          JSR [L0]                        ;Get address of CLOCK CC SET[C]
U 19AB, BE14,15 ;17701          MOV WR[0] TO LS[T10]
U 19AC, 8870,3C ;17702          JSR [L0]                        ;Get address of CLOCK CC SET[V]
U 19AD, 3E16,15 ;17703          MOV WR[0] TO LS[T11]
U 19AE, 8870,3C ;17704          JSR [L0]                        ;Get address of CLOCK CC
U 19AF, BE18,15 ;17705          MOV WR[0] TO LS[T12]
U 19B0, 8870,3C ;17706          JSR [L0]                        ;Get address of MOV FT0 TO FQ, ET0 TO EQ
U 19B1, 3E1A,15 ;17707          MOV WR[0] TO LS[T13]
U 19B2, 8870,3C ;17708          JSR [L0]                        ;Get address of STORE CC
U 19B3, 3E1C,15 ;17709          MOV WR[0] TO LS[T14]
U 19B4, 8870,3C ;17710          JSR [L0]                        ;Get address of CLR FT0, ET0
U 19B5, 3EAE,15 ;17711          MOV WR[0] TO LS[T57]
U 19B6, 8870,3C ;17712          JSR [L0]                        ;Get address of CLOCK SIGN OUT WITH[0]
U 19B7, 3EB0,15 ;17713          MOV WR[0] TO LS[T58]
U 19B8, 8870,3C ;17714          JSR [L0]                        ;Get address of CLOCK SIGN OUT WITH[1]
U 19B9, BEB2,15 ;17715          MOV WR[0] TO LS[T59]
U 19BA, 367E,15 ;17716  T28.1:  MOV LS[BIT31] TO WR[0]          ;Setup data
U 19BB, 3EA2,15 ;17717          MOV WR[0] TO LS[DATA.1]
U 19BC, E5A4,15 ;17718          CLR LS[DATA.2]
U 19BD, 65A6,15 ;17719          CLR LS[DATA.3]
U 19BE, 0878,FC ;17720          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 19BF, 3716,15 ;17721          MOV LS[#300] TO WR[0]           ;Set the instruction bits to MULL
U 19C0, C764,15 ;17722          BIS LS[BIT18] TO WR[0]          ; and the size bit to zero
U 19C1, 475C,15 ;17723          BIS LS[BIT14] TO WR[0]
U 19C2, 475A,15 ;17724          BIS LS[BIT13] TO WR[0]
U 19C3, 90F6,15 ;17725          MISC2 [TRAP.ACC] WR[0]
U 19C4, B61A,15 ;17726          MOV LS[T13] TO WR[0]            ;Setup to MOV FT0 TO FT0
```

```
U 19C5, 3610,95 ;17727          MOV LS[T8] TO WR[1]          ;Setup to CLOCK CC SET[V.C]
U 19C6, 90F6,15 ;17728          MISC2 [TRAP.ACC] WR[0]       ;MOV FTO TO FTO
U 19C7, 10F6,95 ;17729          MISC2 [TRAP.ACC] WR[1]       ;CLOCK CC SET[V.C]
U 19C8, B716,95 ;17730          MOV LS[#300] TO WR[1]        ;Setup to Loop self
U 19C9, 10F6,95 ;17731          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19CA, 4764,95 ;17732          BIS LS[BIT18] TO WR[1]       ;Setup to reset the instruction and
U 19CB, 10F6,95 ;17733          MISC2 [TRAP.ACC] WR[1]       ; size bits
U 19CC, B61C,15 ;17734          MOV LS[T14] TO WR[0]         ;Setup to STORE CC
U 19CD, 90F6,15 ;17735          MISC2 [TRAP.ACC] WR[0]       ;STORE CC
U 19CE, 3A1E,15 ;17736          MOV FPA TO LS[T15]           ;Enable CC into CPU
U 19CF, 10F6,95 ;17737          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19D0, 361E,15 ;17738          MOV LS[T15] TO WR[0]         ;Setup received data
U 19D1, 3646,95 ;17739          MOV LS[BIT3] TO WR[1]        ;Setup expected data
U 19D2, C742,95 ;17740          BIS LS[BIT1] TO WR[1]
U 19D3, 4740,95 ;17741          BIS LS[BIT0] TO WR[1]
U 19D4, 0869,3C ;17742          JSR [CHECK.RESULT]           ;Check for error
U 19D5, 899B,A4 ;17743          JMP [T28.1]                  ;Loop on error
U 19D6, FF82,15 ;17744          INC LS[ERROR.NUMBER]         ;Go onto error 2
U 19D7, B6AE,15 ;17745  T28.2:  MOV LS[T57] TO WR[0]         ;Setup to CLR FTO
U 19D8, B716,95 ;17746          MOV LS[#300] TO WR[1]        ;Setup to Loop Self
U 19D9, 90F6,15 ;17747          MISC2 [TRAP.ACC] WR[0]       ;CLR FTO
U 19DA, 10F6,95 ;17748          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19DB, 36B2,15 ;17749          MOV LS[T59] TO WR[0]         ;Setup to CLOCK SIGN OUT WITH[1]
U 19DC, 90F6,15 ;17750          MISC2 [TRAP.ACC] WR[0]       ;CLOCK SIGN OUT WITH[1]
U 19DD, 10F6,95 ;17751          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19DE, B61A,15 ;17752          MOV LS[T13] TO WR[0]         ;Setup to MOV FTO TO FTO
U 19DF, 3616,95 ;17753          MOV LS[T11] TO WR[1]         ;Setup to CLOCK CC SET[V]
U 19E0, 90F6,15 ;17754          MISC2 [TRAP.ACC] WR[0]       ;MOV FTO TO FTO
U 19E1, 10F6,95 ;17755          MISC2 [TRAP.ACC] WR[1]       ;CLOCK CC SET[V]
U 19E2, B716,95 ;17756          MOV LS[#300] TO WR[1]        ;Setup to Loop self
U 19E3, 10F6,95 ;17757          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19E4, B61C,15 ;17758          MOV LS[T14] TO WR[0]         ;Setup to STORE CC
U 19E5, 90F6,15 ;17759          MISC2 [TRAP.ACC] WR[0]       ;STORE CC
U 19E6, 3A1E,15 ;17760          MOV FPA TO LS[T15]           ;Enable CC into CPU
U 19E7, 10F6,95 ;17761          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19E8, 361E,15 ;17762          MOV LS[T15] TO WR[0]         ;Setup received data
U 19E9, 3646,95 ;17763          MOV LS[BIT3] TO WR[1]        ;Setup expected data
U 19EA, C744,95 ;17764          BIS LS[BIT2] TO WR[1]
U 19EB, C742,95 ;17765          BIS LS[BIT1] TO WR[1]
U 19EC, 0869,3C ;17766          JSR [CHECK.RESULT]           ;Check for error
U 19ED, 099D,74 ;17767          JMP [T28.2]                  ;Loop on error
U 19EE, 65A6,15 ;17768  T28.2A: CLR LS[DATA.3]               ;Setup data
U 19EF, E5A4,15 ;17769          CLR LS[DATA.2]
U 19F0, 364E,15 ;17770          MOV LS[BIT7] TO WR[0]
U 19F1, 3EA2,15 ;17771          MOV WR[0] TO LS[DATA.1]
U 19F2, 0878,FC ;17772          JSR [LOAD.TRIPLE]            ;Load data into FTO
U 19F3, 36B2,15 ;17773          MOV LS[T59] TO WR[0]         ;Setup to CLOCK SIGN OUT WITH[1]
U 19F4, B716,95 ;17774          MOV LS[#300] TO WR[1]        ;Setup to Loop self
U 19F5, 90F6,15 ;17775          MISC2 [TRAP.ACC] WR[0]       ;CLOCK SIGN OUT WITH[1]
U 19F6, 10F6,95 ;17776          MISC2 [TRAP.ACC] WR[1]       ;Loop self
U 19F7, B61A,15 ;17777          MOV LS[T13] TO WR[0]         ;Setup to MOV FTO TO FTO
U 19F8, 3616,95 ;17778          MOV LS[T11] TO WR[1]         ;Setup to CLOCK CC SET[V]
U 19F9, 90F6,15 ;17779          MISC2 [TRAP.ACC] WR[0]       ;MOV FTO TO FTO
U 19FA, 10F6,95 ;17780          MISC2 [TRAP.ACC] WR[1]       ;CLOCK CC SET[V]
U 19FB, B716,95 ;17781          MOV LS[#300] TO WR[1]        ;Setup to Loop self
```

```
U 19FC, 10F6,95 ;17782          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 19FD, B61C,15 ;17783          MOV LS[T14] TO WR[0]            ;Setup to STORE CC
U 19FE, 90F6,15 ;17784          MISC2 [TRAP.ACC] WR[0]          ;STORE CC
U 19FF, 3A1E,15 ;17785          MOV FPA TO LS[T15]              ;Enable CC into CPU
U 1A00, 10F6,95 ;17786          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A01, 361E,15 ;17787          MOV LS[T15] TO WR[0]            ;Setup received data
U 1A02, 3646,95 ;17788          MOV LS[BIT3] TO WR[1]           ;Setup expected data
U 1A03, C742,95 ;17789          BIS LS[BIT1] TO WR[1]
U 1A04, 0869,3C ;17790          JSR [CHECK.RESULT]              ;Check for error
U 1A05, 099D,74 ;17791          JMP [T28.2]                     ;Loop on error
U 1A06, FF82,15 ;17792          INC LS[ERROR.NUMBER]            ;Go onto error 3
U 1A07, 087C,1C ;17793  T28.3:  JSR [CLR.FTO]                   ;LOAD ZEROS INTO FTO
U 1A08, 3716,15 ;17794          MOV LS[#300] TO WR[0]           ;Set the instruction bits to MULL
U 1A09, C764,15 ;17795          BIS LS[BIT18] TO WR[0]          ; and the size bit to zero
U 1A0A, 475C,15 ;17796          BIS LS[BIT14] TO WR[0]
U 1A0B, 475A,15 ;17797          BIS LS[BIT13] TO WR[0]
U 1A0C, 90F6,15 ;17798          MISC2 [TRAP.ACC] WR[0]
U 1A0D, B61A,15 ;17799          MOV LS[T13] TO WR[0]            ;Setup to MOV FTO TO FTO
U 1A0E, B614,95 ;17800          MOV LS[T10] TO WR[1]            ;Setup to CLOCK CC SET[C]
U 1A0F, 90F6,15 ;17801          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FTO
U 1A10, 10F6,95 ;17802          MISC2 [TRAP.ACC] WR[1]          ;CLOCK CC SET[C]
U 1A11, B716,95 ;17803          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1A12, 10F6,95 ;17804          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A13, 4764,95 ;17805          BIS LS[BIT18] TO WR[1]          ;Setup to reset the instruction and
U 1A14, 10F6,95 ;17806          MISC2 [TRAP.ACC] WR[1]          ; size bits
U 1A15, B61C,15 ;17807          MOV LS[T14] TO WR[0]            ;Setup to STORE CC
U 1A16, 90F6,15 ;17808          MISC2 [TRAP.ACC] WR[0]          ;STORE CC
U 1A17, 3A1E,15 ;17809          MOV FPA TO LS[T15]              ;Enable CC into CPU
U 1A18, 10F6,95 ;17810          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A19, 361E,15 ;17811          MOV LS[T15] TO WR[0]            ;Setup received data
U 1A1A, B644,95 ;17812          MOV LS[BIT2] TO WR[1]           ;Setup expected data
U 1A1B, 4740,95 ;17813          BIS LS[BIT0] TO WR[1]
U 1A1C, 0869,3C ;17814          JSR [CHECK.RESULT]              ;Check for error
U 1A1D, 89A0,74 ;17815          JMP [T28.3]                     ;Loop on error
U 1A1E, FF82,15 ;17816          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 1A1F, B6AE,15 ;17817  T28.4:  MOV LS[T57] TO WR[0]            ;Setup to CLR FTO
U 1A20, B716,95 ;17818          MOV LS[#300] TO WR[1]           ;Setup to Loop Self
U 1A21, 90F6,15 ;17819          MISC2 [TRAP.ACC] WR[0]          ;CLR FTO
U 1A22, 10F6,95 ;17820          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A23, B6B0,15 ;17821          MOV LS[T58] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[0]
U 1A24, 90F6,15 ;17822          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[0]
U 1A25, 10F6,95 ;17823          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A26, B61A,15 ;17824          MOV LS[T13] TO WR[0]            ;Setup to MOV FTO TO FTO
U 1A27, B618,95 ;17825          MOV LS[T12] TO WR[1]            ;Setup to CLOCK CC
U 1A28, 90F6,15 ;17826          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FTO
U 1A29, 10F6,95 ;17827          MISC2 [TRAP.ACC] WR[1]          ;CLOCK CC
U 1A2A, B716,95 ;17828          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1A2B, 10F6,95 ;17829          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A2C, B61C,15 ;17830          MOV LS[T14] TO WR[0]            ;Setup to STORE CC
U 1A2D, 90F6,15 ;17831          MISC2 [TRAP.ACC] WR[0]          ;STORE CC
U 1A2E, 3A1E,15 ;17832          MOV FPA TO LS[T15]              ;Enable CC into CPU
U 1A2F, 10F6,95 ;17833          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A30, 361E,15 ;17834          MOV LS[T15] TO WR[0]            ;Setup received data
U 1A31, B644,95 ;17835          MOV LS[BIT2] TO WR[1]           ;Setup expected data
U 1A32, 0869,3C ;17836          JSR [CHECK.RESULT]              ;Check for error
```

```
U 1A33, 89A1,F4 ;17837          JMP [T28.4]                     ;Loop on error
U 1A34, FF82,15 ;17838          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 1A35, E5A4,15 ;17839  T28.5:  CLR LS[DATA.2]                  ;Setup data
U 1A36, 3660,15 ;17840          MOV LS[BIT16] TO WR[0]
U 1A37, 3EA2,15 ;17841          MOV WR[0] TO LS[DATA.1]
U 1A38, 65A6,15 ;17842          CLR LS[DATA.3]
U 1A39, 0878,FC ;17843          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 1A3A, 3716,15 ;17844          MOV LS[#300] TO WR[0]           ;Set the instruction bits to MULL
U 1A3B, C764,15 ;17845          BIS LS[BIT18] TO WR[0]          ; and the size bit to zero
U 1A3C, 475C,15 ;17846          BIS LS[BIT14] TO WR[0]
U 1A3D, 475A,15 ;17847          BIS LS[BIT13] TO WR[0]
U 1A3E, 90F6,15 ;17848          MISC2 [TRAP.ACC] WR[0]
U 1A3F, B61A,15 ;17849          MOV LS[T13] TO WR[0]            ;Setup to MOV FTO TO FTO
U 1A40, B618,95 ;17850          MOV LS[T12] TO WR[1]            ;Setup to CLOCK CC
U 1A41, 90F6,15 ;17851          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FTO
U 1A42, 10F6,95 ;17852          MISC2 [TRAP.ACC] WR[1]          ;CLOCK CC
U 1A43, B716,95 ;17853          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1A44, 10F6,95 ;17854          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A45, 4764,95 ;17855          BIS LS[BIT18] TO WR[1]          ;Setup to reset the instruction and
U 1A46, 10F6,95 ;17856          MISC2 [TRAP.ACC] WR[1]          ; size bits
U 1A47, B61C,15 ;17857          MOV LS[T14] TO WR[0]            ;Setup to STORE CC
U 1A48, 90F6,15 ;17858          MISC2 [TRAP.ACC] WR[0]          ;STORE CC
U 1A49, 3A1E,15 ;17859          MOV FPA TO LS[T15]              ;Enable CC into CPU
U 1A4A, 10F6,95 ;17860          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A4B, 361E,15 ;17861          MOV LS[T15] TO WR[0]            ;Setup received data
U 1A4C, 2F82,95 ;17862          CLR WR[1]                       ;Setup expected data
U 1A4D, 0869,3C ;17863          JSR [CHECK.RESULT]              ;Check for error
U 1A4E, 09A3,54 ;17864          JMP [T28.5]                     ;Loop on error
U 1A4F, FF82,15 ;17865          INC LS[ERROR.NUMBER]            ;Go onto error 6
U 1A50, 65A6,15 ;17866  T28.6:  CLR LS[DATA.3]                  ;Setup data
U 1A51, E5A4,15 ;17867          CLR LS[DATA.2]
U 1A52, 364E,15 ;17868          MOV LS[BIT7] TO WR[0]
U 1A53, 3EA2,15 ;17869          MOV WR[0] TO LS[DATA.1]
U 1A54, 0878,FC ;17870          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 1A55, B6B0,15 ;17871          MOV LS[T58] TO WR[0]            ;Setup to CLOCK SIGN OUT WITH[0]
U 1A56, 90F6,15 ;17872          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH[0]
U 1A57, 10F6,95 ;17873          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A58, B61A,15 ;17874          MOV LS[T13] TO WR[0]            ;Setup to MOV FTO TO FTO
U 1A59, B618,95 ;17875          MOV LS[T12] TO WR[1]            ;Setup to CLOCK CC
U 1A5A, 90F6,15 ;17876          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FTO
U 1A5B, 10F6,95 ;17877          MISC2 [TRAP.ACC] WR[1]          ;CLOCK CC
U 1A5C, B716,95 ;17878          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1A5D, 10F6,95 ;17879          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A5E, B61C,15 ;17880          MOV LS[T14] TO WR[0]            ;Setup to STORE CC
U 1A5F, 90F6,15 ;17881          MISC2 [TRAP.ACC] WR[0]          ;STORE CC
U 1A60, 3A1E,15 ;17882          MOV FPA TO LS[T15]              ;Enable CC into CPU
U 1A61, 10F6,95 ;17883          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1A62, 361E,15 ;17884          MOV LS[T15] TO WR[0]            ;Setup received data
U 1A63, 2F82,95 ;17885          CLR WR[1]                       ;Setup expected data
U 1A64, 0869,3C ;17886          JSR [CHECK.RESULT]              ;Check for error
U 1A65, 09A5,04 ;17887          JMP [T28.6]                     ;Loop on error
                ;17888  T28.END:
```

```
;17889  .PAGE
;17890  .TOC    "TEST 29 ENABLE LITERAL TEST(M8389 FPA MODULE)"
;17891  ;++
;17892  ;
;17893  ;
;17894  ; Test Description:
;17895  ;
;17896  ;       The purpose of this test is to check the ENABLE LITERAL clock condition
;17897  ;       If the mod field is set to extend clock and the clock field is set to
;17898  ;       ENABLE LITERAL then the lower seven bits of the micropointer field will
;17899  ;       be enabled onto bits 7 - 14 of the FPA BUS. To determine that the bits
;17900  ;       were actually enabled onto the FPA BUS an OR instruction in both the
;17901  ;       exponent and fraction data paths with 0 can be executed. This should
;17902  ;       cause the FPA BUS bits 7 - 14 to appear in bits 0 - 6 of the extension
;17903  ;       data path, bits 23 - 30 of the fraction data path, and bits 0 - 7 of
;17904  ;       the extension data path. The other thing that will occur if ENABLE LIT-
;17905  ;       ERAL is asserted is the microsequencer will take the next PC from the
;17906  ;       microprogram counter.
;17907  ;
;17908  ; Assumptions:
;17909  ;
;17910  ;       It is assumed that the branch logic has been tested previously and that
;17911  ;       it works.
;17912  ;
;17913  ; Test Steps:
;17914  ;
;17915  ;       1) Load a WR with 0's. Issue a nop followed by a nop with the micro-
;17916  ;          pointer field bits all set to 1. Issue a LITERAL[] TO EWR[]. The
;17917  ;          literal field should contain 01. Store the contents of the EWR to
;17918  ;          the CPU and check for error. IF there is an error then issue error 1
;17919  ;       2) Repeat step one except load the WR with ones and the micropointer
;17920  ;          field with FF. If there is an error then issue error 2.
;17921  ;       3) Repeat step one except the literal field should contain 80. If there
;17922  ;          is anr error then issue error 3.
;17923  ;       4) Repeat step one except the literal field should contain 0A and the
;17924  ;          it should be EQ instead of EWR. If there is an error then issue
;17925  ;          error 4.
;17926  ;       5) Issue a nop followed by a nop with the micropointer field all zeros
;17927  ;          and the clock field set to 010 and the mod field set to extend
;17928  ;          clock. Issue a read of the micro address lines and then a move back
;17929  ;          to the CPU. If the address lines are not equal to what's in the
;17930  ;          micropionter field of the word with the literal in it then issue
;17931  ;          error 5.
;17932  ;
;17933  ; Assumptions:
;17934  ;
;17935  ;       It is assumed that the branch logic has been tested previously and
;17936  ;       works.
;17937  ;
;17938  ; Error:
;17939  ;
;17940  ;       Error1 - ENABLE LITERAL failed to enable 01 into EWR.
;17941  ;       Error2 - ENABLE LITERAL failed to enable FF into EWR.
;17942  ;       Error3 - ENABLE LITERAL failed to enable 80 into EWR.
;17943  ;       Error4 - ENABLE LITERAL failed to enable 0A into EQ.
```

```
                ;17944  ;       Error5 - ENABLE LITERAL failed to cause the UPC to select the next
                ;17945  ;                microword for the next address.
                ;17946  ;
                ;17947  ; Debug:
                ;17948  ;
                ;17949  ;       Error1: If this error occurs then there are two likely causes of the
                ;17950  ;               error, they are the failure of the enable signal or the failure
                ;17951  ;               of the line drivers. If ENABLE LITERAL L failed then the CC CTL
                ;17952  ;               PAL should be checked for error. If the PAL is not the cause of
                ;17953  ;               error then EXT CLK and ENB CLK2 should be checked to be
                ;17954  ;               asserted. If the enable isn't the cause of the error then the
                ;17955  ;               LITERAL FIELD LINE DRIVERS should be checked for error.
                ;17956  ;       Error2: Same as error 1.
                ;17957  ;       Error3: Same as error 1.
                ;17958  ;       Error4: Same as error 1, except the destination was EQ.
                ;17959  ;       Error5: Same as error 1 except that the 2909's or the gates preceeding
                ;17960  ;               2909's below then EXT BRANCH PAL could also be the cause of the
                ;17961  ;               error and should be suspected if the first two errors don't
                ;17962  ;               occur.
                ;17963  ;
                ;17964  ;
                ;17965  ;--
                ;17966  T.29:
U 1A66, B65E,15 ;17967          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;17968                                          ; STATUS WORD
U 1A67, 3E80,15 ;17969          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;17970                                          ; BIT 15 INDICATES START OF TEST TO
                ;17971                                          ; CONSOLE
U 1A68, 10E0,15 ;17972          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;17973  WAIT.T29.0:
U 1A69, 09A6,94 ;17974          JMP [WAIT.T29.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;17975                                          ; RESPOND
U 1A6A, 087E,6C ;17976          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;17977                                          ; AND SIZE BITS
U 1A6B, B65E,15 ;17978          MOV LS[BIT15] TO WR[0]          ;Setup error mask
U 1A6C, 3E8A,15 ;17979          MOV WR[0] TO LS[ERROR.MASK]
U 1A6D, B7B2,15 ;17980          MOV LS[T29.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1A6E, BE12,15 ;17981          MOV WR[0] TO LS[T9]
U 1A6F, 8870,3C ;17982          JSR [LO]                        ;Get address of LITERAL[2] TO EQ
U 1A70, 3E10,15 ;17983          MOV WR[0] TO LS[T8]
U 1A71, 8870,3C ;17984          JSR [LO]                        ;Get address of MOV FT0 TO FT9 and ET0
U 1A72, BE14,15 ;17985          MOV WR[0] TO LS[T10]             ; TO ET9
U 1A73, 8870,3C ;17986          JSR [LO]                        ;Get address of STORE FT9 to the CPU
U 1A74, 3E16,15 ;17987          MOV WR[0] TO LS[T11]
U 1A75, 8870,3C ;17988          JSR [LO]                        ;Get address of LITERAL[FF] TO EWR[ETC]
U 1A76, BE18,15 ;17989          MOV WR[0] TO LS[T12]
U 1A77, 8870,3C ;17990          JSR [LO]                        ;Get address of MOV FT0 TO FTC and ET0
U 1A78, BF8C,15 ;17991          MOV WR[0] TO LS[TC5]             ; TO ETC
U 1A79, 8870,3C ;17992          JSR [LO]                        ;Get address of STORE FTC to the CPU
U 1A7A, 3E1C,15 ;17993          MOV WR[0] TO LS[T14]
U 1A7B, 8870,3C ;17994          JSR [LO]                        ;Get address of LITERAL[80] TO EWR[ET6]
U 1A7C, BE1E,15 ;17995          MOV WR[0] TO LS[T15]
U 1A7D, 8870,3C ;17996          JSR [LO]                        ;Get address of MOV FT0 TO FT6 and ET0
U 1A7E, 3EAE,15 ;17997          MOV WR[0] TO LS[T57]             ; TO ET6
U 1A7F, 8870,3C ;17998          JSR [LO]                        ;Get address of STORE FT6 to the CPU
```

```
U 1A80, 3EB0,15 ;17999          MOV WR[0] TO LS[T58]
U 1A81, 8870,3C ;18000          JSR [L0]                    ;Get address of LITERAL[0A] TO EQ
U 1A82, BEB2,15 ;18001          MOV WR[0] TO LS[T59]
U 1A83, 8870,3C ;18002          JSR [L0]                    ;Get addresss of MOV FT0 TO FQ and ET0
U 1A84, 3F90,15 ;18003          MOV WR[0] TO LS[TC8]        ; TO EQ
U 1A85, 8870,3C ;18004          JSR [L0]                    ;Get address of MOV FQ TO FT0 and EQ TO
U 1A86, BF92,15 ;18005          MOV WR[0] TO LS[TC9]
U 1A87, B69E,15 ;18006  T29.1:  MOV LS[ONES] TO WR[0]       ;Setup data
U 1A88, 3EA2,15 ;18007          MOV WR[0] TO LS[DATA.1]
U 1A89, 0878,FC ;18008          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 1A8A, B790,15 ;18009          MOV LS[TC8] TO WR[0]        ;Setup to MOV FT0 TO FQ and ET0 TO EQ
U 1A8B, 90F6,15 ;18010          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ and ET0 TO EQ
U 1A8C, 10F6,95 ;18011          MISC2 [TRAP.ACC] WR[1]      ;Loop self
U 1A8D, B610,15 ;18012          MOV LS[T8] TO WR[0]         ;Setup TO LITERAL[1] TO EWR[ET9]
U 1A8E, 90F6,15 ;18013          MISC2 [TRAP.ACC] WR[0]      ;Force the word before the literal
U 1A8F, DB00,15 ;18014          NOP                         ;LITERAL[2] TO EQ
U 1A90, DB00,15 ;18015          NOP
U 1A91, 10F6,95 ;18016          MISC2 [TRAP.ACC] WR[1]      ;Loop self
U 1A92, 3792,15 ;18017          MOV LS[TC9] TO WR[0]        ;Setup to MOV FQ TO FT0 AND EQ TO ET0
U 1A93, 90F6,15 ;18018          MISC2 [TRAP.ACC] WR[0]      ;MOV FQ TO FT0 AND EQ TO ET0
U 1A94, 10F6,95 ;18019          MISC2 [TRAP.ACC] WR[1]      ;Loop self
U 1A95, 087B,5C ;18020          JSR [STORE.0.TRIPLE]        ;Store date
U 1A96, B6A8,15 ;18021          MOV LS[FIRST.FLT] TO WR[0]  ;Setup received data
U 1A97, B628,95 ;18022          MOV LS[#FFFF] TO WR[1]      ;Setup expected data
U 1A98, A0C2,95 ;18023          COM WR[1]
U 1A99, 4726,95 ;18024          BIS LS[#FF] TO WR[1]
U 1A9A, 0869,3C ;18025          JSR [CHECK.RESULT]          ;Check for error
U 1A9B, 09A8,74 ;18026          JMP [T29.1]                 ;Loop on error
U 1A9C, FF82,15 ;18027          INC LS[ERROR.NUMBER]        ;Go onto error 2
U 1A9D, B61C,15 ;18028          MOV LS[T14] TO WR[0]        ;Setup for the STORE
U 1A9E, 3E16,15 ;18029          MOV WR[0] TO LS[T11]
U 1A9F, E5A2,15 ;18030  T29.2:  CLR LS[DATA.1]              ;Setup data
U 1AA0, 0878,FC ;18031          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 1AA1, 378C,15 ;18032          MOV LS[TC5] TO WR[0]        ;Setup to MOV FT0 TO FTC and ET0 TO ETC
U 1AA2, 90F6,15 ;18033          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FTC and ET0 TO ETC
U 1AA3, 10F6,95 ;18034          MISC2 [TRAP.ACC] WR[1]      ;Loop self
U 1AA4, 3618,15 ;18035          MOV LS[T12] TO WR[0]        ;Setup TO LITERAL[0FF] TO EWR[ETC]
U 1AA5, 90F6,15 ;18036          MISC2 [TRAP.ACC] WR[0]      ;Force the word before the literal
U 1AA6, DB00,15 ;18037          NOP                         ;LITERAL[0FF] TO EWR[ETC]
U 1AA7, DB00,15 ;18038          NOP
U 1AA8, 10F6,95 ;18039          MISC2 [TRAP.ACC] WR[1]      ;Loop self
U 1AA9, 8878,2C ;18040          JSR [STORE.DATA.1]          ;Store FF of FTC, ETC to the CPU
U 1AAA, B61A,15 ;18041          MOV LS[T13] TO WR[0]        ;Setup received data
U 1AAB, B628,95 ;18042          MOV LS[#FFFF] TO WR[1]
U 1AAC, 422C,95 ;18043          AND LS[#FFFFFF00] TO WR[1]
U 1AAD, C74E,95 ;18044          BIS LS[BIT7] TO WR[1]
U 1AAE, 0869,3C ;18045          JSR [CHECK.RESULT]          ;Check for error
U 1AAF, 09A9,F4 ;18046          JMP [T29.2]                 ;Loop on error
U 1AB0, FF82,15 ;18047          INC LS[ERROR.NUMBER]        ;Go onto error 3
U 1AB1, B6B0,15 ;18048          MOV LS[T58] TO WR[0]        ;Setup for the STORE
U 1AB2, 3E16,15 ;18049          MOV WR[0] TO LS[T11]
U 1AB3, B69E,15 ;18050  T29.3:  MOV LS[ONES] TO WR[0]       ;Setup data
U 1AB4, 3EA2,15 ;18051          MOV WR[0] TO LS[DATA.1]
U 1AB5, 0878,FC ;18052          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 1AB6, B6AE,15 ;18053          MOV LS[T57] TO WR[0]        ;Setup to MOV FT0 TO FT6 and ET0 TO ET6
```

```
U 1AB7, 90F6,15 ;18054          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT6 and ET0 TO ET6
U 1AB8, 10F6,95 ;18055          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1AB9, 361E,15 ;18056          MOV LS[T15] TO WR[0]            ;Setup TO LITERAL[80] TO EWR[ET6]
U 1ABA, 90F6,15 ;18057          MISC2 [TRAP.ACC] WR[0]          ;Force the word before the literal
U 1ABB, DB00,15 ;18058          NOP                             ;LITERAL[80] TO EWR[ET6]
U 1ABC, DB00,15 ;18059          NOP
U 1ABD, 10F6,95 ;18060          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1ABE, 8878,2C ;18061          JSR [STORE.DATA.1]              ;Store FF of FT6, ET6 to the CPU
U 1ABF, B61A,15 ;18062          MOV LS[T13] TO WR[0]            ;Setup received data
U 1AC0, B628,95 ;18063          MOV LS[#FFFF] TO WR[1]          ;Setup expected data
U 1AC1, A0C2,95 ;18064          COM WR[1]
U 1AC2, 4726,95 ;18065          BIS LS[#FF] TO WR[1]
U 1AC3, C75C,95 ;18066          BIS LS[BIT14] TO WR[1]
U 1AC4, 454E,95 ;18067          BIC LS[BIT7] TO WR[1]
U 1AC5, 0869,3C ;18068          JSR [CHECK.RESULT]              ;Check for error
U 1AC6, 89AB,34 ;18069          JMP [T29.3]                     ;Loop on error
U 1AC7, FF82,15 ;18070          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 1AC8, B69E,15 ;18071  T29.4:  MOV LS[ONES] TO WR[0]           ;Setup data
U 1AC9, 3EA2,15 ;18072          MOV WR[0] TO LS[DATA.1]
U 1ACA, 0878,FC ;18073          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 1ACB, B790,15 ;18074          MOV LS[TC8] TO WR[0]            ;Setup to MOV FT0 TO FQ and ET0 TO EQ
U 1ACC, 90F6,15 ;18075          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ and ET0 TO EQ
U 1ACD, 10F6,95 ;18076          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1ACE, 36B2,15 ;18077          MOV LS[T59] TO WR[0]            ;Setup TO LITERAL[0A] TO EQ
U 1ACF, 90F6,15 ;18078          MISC2 [TRAP.ACC] WR[0]          ;Force the word before the literal
U 1AD0, DB00,15 ;18079          NOP
U 1AD1, DB00,15 ;18080          NOP                             ;LITERAL[0A] TO EQ
U 1AD2, 10F6,95 ;18081          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1AD3, 3792,15 ;18082          MOV LS[TC9] TO WR[0]            ;Setup to MOV FQ, EQ TO FT0 ET0
U 1AD4, 90F6,15 ;18083          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ,EQ TO FT0, ET0
U 1AD5, 10F6,95 ;18084          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1AD6, 087B,5C ;18085          JSR [STORE.0.TRIPLE]            ;Store FT0 to the CPU
U 1AD7, B6A8,15 ;18086          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 1AD8, B628,95 ;18087          MOV LS[#FFFF] TO WR[1]          ;Setup expected data
U 1AD9, A0C2,95 ;18088          COM WR[1]
U 1ADA, 4726,95 ;18089          BIS LS[#FF] TO WR[1]
U 1ADB, 4754,95 ;18090          BIS LS[BIT10] TO WR[1]
U 1ADC, 0869,3C ;18091          JSR [CHECK.RESULT]              ;Check for error
U 1ADD, 89AC,84 ;18092          JMP [T29.4]                     ;Loop on error
U 1ADE, FF82,15 ;18093          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 1ADF, B700,15 ;18094          MOV LS[FFFFFC00.MASK] TO WR[0]  ;Change error mask for NUA
U 1AE0, 3E8A,15 ;18095          MOV WR[0] TO LS[ERROR.MASK]
U 1AE1, B610,15 ;18096  T29.5:  MOV LS[T8] TO WR[0]             ;Setup to LITERAL[2] TO EQ
U 1AE2, B716,95 ;18097          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 1AE3, 90F6,15 ;18098          MISC2 [TRAP.ACC] WR[0]          ;Force the word before the literal
U 1AE4, DB00,15 ;18099          NOP                             ;Wait for literal
U 1AE5, DB00,15 ;18100          NOP
U 1AE6, 10F8,15 ;18101          MISC2 [READ.ACC.UPC]            ;Enable the NUA onto the FPA
U 1AE7, BA1A,15 ;18102          MOV FPA TO LS[T13]              ;Enable the FPA onto the CPU
U 1AE8, 10F6,95 ;18103          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 1AE9, B61A,15 ;18104          MOV LS[T13] TO WR[0]            ;Setup received data
U 1AEA, 364A,95 ;18105          MOV LS[BIT5] TO WR[1]           ;Setup expected data
U 1AEB, 474C,95 ;18106          BIS LS[BIT6] TO WR[1]
U 1AEC, 4740,95 ;18107          BIS LS[BIT0] TO WR[1]
U 1AED, 0869,3C ;18108          JSR [CHECK.RESULT]              ;Check for error
```

```
U 1AEE, 09AE,14 ;18109          JMP [T29.5]                    ;Loop on error
                ;18110  T29.END:
```

```
;18111  .page
;18112  .TOC    "TEST 2A EXP COUT AND GRAND BRANCH TEST(M8389 FPA MODULE)"
;18113  ;++
;18114  ;
;18115  ;
;18116  ; Test Description:
;18117  ;
;18118  ;       The purpose of this test will be to test the EXP COUT and GRAND
;18119  ;       branch condition. The branch control bits will be set to 00000.
;18120  ;       If EXP COUT is asserted and GRAND is asserted then the microcode
;18121  ;       will branch to 4 plus the contents of the next micro address
;18122  ;       field provided that the two least significant bits of the NUA are
;18123  ;       zero. The NUA will be 3 + the NUA field if only EXP COUT is asserted.
;18124  ;       The NUA will be 2 + the NUA field if only GRAND is asserted. If neither
;18125  ;       of the signals are asserted then the NUA will be whatever's in the NUA
;18126  ;       field. The data path that will be used is as follows: the branch condi-
;18127  ;       tions will be asserted then a branch instruction will be issued and
;18128  ;       regardless of whether there is a branch or not the next instruction
;18129  ;       will be jump to self. Then the NUA will be read to see if the correct
;18130  ;       branch was taken. This same procedure will be followed to check that
;18131  ;       all four of the possible jumps can be made.
;18132  ;
;18133  ; Assumptions:
;18134  ;
;18135  ;       All of the 2901 logic, the condition code and sign logic, and the
;18136  ;       majority of the clock field logic should have been tested and should
;18137  ;       be working.
;18138  ;
;18139  ; Test Steps:
;18140  ;
;18141  ;       1) Load the exponent data path with ones in all 16 bits by loading
;18142  ;          eight bits and shifting and then loading eight more. Issue an add
;18143  ;          B PLUS A, inorder to set EXP COUT. Previous to this instruction the
;18144  ;          IB BUS should have been enabled into the FPA instruction ROM to load
;18145  ;          the SIZE bits with 10. Having set both the branch conditions a nop
;18146  ;          instruction will be issued with a branch on these conditions. If
;18147  ;          the branch doesn't branch to the NUA field + 3 then issue error1.
;18148  ;       2) Repeat step 1 except delete the steps required to set the EXP COUT
;18149  ;          branch condition. If the branch doesn't branch to the NUA field + 1
;18150  ;          then issue error2.
;18151  ;       3) Repeat step 1 except delete the steps required to set GRAND. If the
;18152  ;          branch doesn't branch to the NUA field + 2 then issue error3.
;18153  ;       4) Repeat step 1 expect delete the steps required to set either of the
;18154  ;          branch conditions. If the branch does branch at all then issue
;18155  ;          error4.
;18156  ;
;18157  ; Error:
;18158  ;
;18159  ;       Error1 - Branch failed when both EXP COUT and GRAND were asserted.
;18160  ;       Error2 - Branch failed when GRAND was asserted.
;18161  ;       Error3 - Branch failed when EXP COUT was asserted.
;18162  ;       Error4 - Branch occured when none of the branch condtions were asserted
;18163  ;
;18164  ; Debug:
;18165  ;
```

```
                ;18166  ;       Error1: If this error occurs there are several possibilities that could
                ;18167  ;               cause the error, they are the control logic to the microse-
                ;18168  ;               quencer, the microsequencer, or the logic behind the branch
                ;18169  ;               condition. The select lines should be 01. If the select lines
                ;18170  ;               are in error then the logic next to the microsequencer should
                ;18171  ;               be checked for error. If the logic is not the cause of error
                ;18172  ;               then FPAA IRD STATE L should be checked to be asserted, FPAD
                ;18173  ;               UBCTL4 (1) H, FPAD UBCTL3 (1) H, FPAD UBCTL2 (1) H, FPAD LIT
                ;18174  ;               CLK2 L, FPAD EXTEND CLK (1) H should all be checked to be
                ;18175  ;               unasserted. If any of the UBCTL signals are in error then the
                ;18176  ;               the CS latch they came from should be checked for error. If
                ;18177  ;               FPAA IARD STATE L is in error then the gate above the
                ;18178  ;               EXT BRANCH PAL should be checked for error. If FPAD EXTEND CLK
                ;18179  ;               (1) H is in error then the etch the latch it comes from and the
                ;18180  ;               gate it goes to should be checked for error. If FPAD LIT CLK2 L
                ;18181  ;               is in error then the latch it came form should be checked for
                ;18182  ;               error, if the latch isn't in error then the logic before the
                ;18183  ;               latch should be checked for error. If the select logic isn't
                ;18184  ;               the cause of error the branch logic should be checked for
                ;18185  ;               error. For this error both branch 1 and branch 0 should be
                ;18186  ;               asserted. If branch 0 is in error then the BRANCH 1 PAL should
                ;18187  ;               be checked for error. If the PAL isn't the cause of error then
                ;18188  ;               all the UBCTL signals should be checked to be unasserted, and
                ;18189  ;               EXP COUT should be checked to be asserted. If the UBCTL signals
                ;18190  ;               are in error then the CONTROL STORE LATCHES they came from
                ;18191  ;               should be checked for error. If EXP COUT is in error then the
                ;18192  ;               latch it came from should be checked for error. If the
                ;18193  ;               latch isn't the cause of error then the carry out of the 2901
                ;18194  ;               should be checked to be asserted. If the carry out is in error
                ;18195  ;               then suspect the 2901 of error. If branch 0 is in error then
                ;18196  ;               the BRANCH 0 PAL should be checked for error. If the PAL isn't
                ;18197  ;               the cause of error then the UBCTL signals should be all checked
                ;18198  ;               to be uasserted. The SIZE bits should be 10. If the size bits
                ;18199  ;               are in error then check the etch from the MUX to the PAL.
                ;18200  ;       Error2: Same as error1 except that EXP COUT is not asserted.
                ;18201  ;       Error3: Same as error1 except that GRAND is not asserted.
                ;18202  ;       Error4: Same as error1 except that both GRAND and EXP COUT aren't
                ;18203  ;               asserted.
                ;18204  ;
                ;18205  ;
                ;18206  ;
                ;18207  ;       T10 - Load EWR[ET0]
                ;18208  ;       T11 - Shift left EWR[ET0]
                ;18209  ;       T14 - MOV EWR[ET0] to EWR[ET2]
                ;18210  ;       T15 - ADD EWR[ET0] TO EWR[ET2]
                ;18211  ;       T13 - BRANCH on EXP COUT and GRAND
                ;18212  ;
                ;18213  ;--
                ;18214  T.2A:
U 1AEF, B65E,15 ;18215          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;18216                                          ; STATUS WORD
U 1AF0, 3E80,15 ;18217          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18218                                          ; BIT 15 INDICATES START OF TEST TO
                ;18219                                          ; CONSOLE
U 1AF1, 10E0,15 ;18220          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
```

```
                ;18221  WAIT.T2A.0:
U 1AF2, 89AF,24 ;18222          JMP [WAIT.T2A.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18223                                          ; RESPOND
U 1AF3, 887E,EC ;18224          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18225                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1AF4, B73C,15 ;18226          MOV LS[T2A.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1AF5, BE12,15 ;18227          MOV WR[0] TO LS[T9]              ; references.
U 1AF6, 8870,3C ;18228          JSR [L0]
U 1AF7, BE14,15 ;18229          MOV WR[0] TO LS[T10]             ;Load ET0
U 1AF8, 8870,3C ;18230          JSR [L0]
U 1AF9, 3E16,15 ;18231          MOV WR[0] TO LS[T11]             ;shift ET0
U 1AFA, 8870,3C ;18232          JSR [L0]
U 1AFB, 3E1C,15 ;18233          MOV WR[0] TO LS[T14]             ;MOV ET0 to ET2
U 1AFC, 8870,3C ;18234          JSR [L0]
U 1AFD, BE1E,15 ;18235          MOV WR[0] TO LS[T15]             ;Add ET0 to ET2
U 1AFE, 8870,3C ;18236          JSR [L0]
U 1AFF, 3E1A,15 ;18237          MOV WR[0] TO LS[T13]             ;Branch on EXP COUT and GRAND
U 1B00, B744,15 ;18238          MOV LS[#60300] TO WR[0]          ;Set size bits to grand
U 1B01, 90F6,15 ;18239          MISC2 [TRAP.ACC]  WR[0]
U 1B02, 365C,15 ;18240          MOV LS[BIT14] TO WR[0]           ;Set up th MSB of the exponent
U 1B03, 3EA2,15 ;18241          MOV WR[0] TO LS[DATA.1]          ; data path
U 1B04, 8877,DC ;18242  LD.EXP: JSR [LOAD.DATA]                  ;Load data into lower exponent
U 1B05, 0876,2C ;18243          JSR [SHIFT.LEFT.8]               ;Shift the data into upper exponent
U 1B06, 0878,AC ;18244          JSR [MOV.DATA]                   ;MOV data from ET0 to ET2
U 1B07, 361E,15 ;18245          MOV LS[T15] TO WR[0]             ;Set up for force
U 1B08, 361A,95 ;18246          MOV LS[T13] TO WR[1]             ;Set up for force
U 1B09, 90F6,15 ;18247          MISC2 [TRAP.ACC] WR[0]           ;Add ET0 to ET2
U 1B0A, 10F6,95 ;18248          MISC2 [TRAP.ACC] WR[1]           ;Branch on EXP COUT and GRAND
U 1B0B, 10F8,15 ;18249          MISC2 [READ.ACC.UPC]             ;Enable NUA BUS onto FPA BUS
U 1B0C, 3A18,15 ;18250          MOV FPA TO LS[T12]               ;Enable FPA BUS onto Y BUS
U 1B0D, 3618,15 ;18251          MOV LS[T12] TO WR[0]             ;Set up expected data
U 1B0E, 09B2,4C ;18252          JSR [SET.UP.EXP]                 ;Set up received data
U 1B0F, 0869,3C ;18253          JSR [CHECK.RESULT]               ;Report error if there is one
U 1B10, 09B0,44 ;18254          JMP [LD.EXP]                     ;Loop on error
U 1B11, FF82,15 ;18255          INC LS[ERROR.NUMBER]             ;go onto next error
U 1B12, 3682,15 ;18256          MOV LS[ERROR.NUMBER] TO WR[0]    ;Put error number in WR[0] for compare
                ;18257          CMP WR[0] WITH LS[#2],           ;If not error 2
U 1B13, 4F42,35 ;18258           DT(LONG)&SET.ALU.CC
U 1B14, 89B1,81 ;18259          JMP.IF[NEQ] TO [EXP.COUT]        ; then jump to EXP COUT
U 1B15, B69C,95 ;18260          MOV LS[#0] TO WR[1]              ; else clear EXP COUT to zero
U 1B16, BEA2,95 ;18261          MOV WR[1] TO LS[DATA.1]
U 1B17, 09B0,44 ;18262          JMP [LD.EXP]                     ; and repeat branch sequence
                ;18263  EXP.COUT:
                ;18264          CMP WR[0] WITH LS[#3(H)],        ;If not error 3
U 1B18, 4FC0,35 ;18265           DT(LONG)&SET.ALU.CC
U 1B19, 09B1,F1 ;18266          JMP.IF[NEQ] TO [NEITHER]         ; then jump to NEITHER
U 1B1A, B65C,95 ;18267          MOV LS[BIT14] TO WR[1]           ; else set EXP COUT
U 1B1B, BEA2,95 ;18268          MOV WR[1] TO LS[DATA.1]
U 1B1C, 3718,95 ;18269          MOV LS[#00040300] TO WR[1]       ; and the SIZE BITS to SINGLE
U 1B1D, 10F6,95 ;18270          MISC2 [TRAP.ACC] WR[1]
U 1B1E, 09B0,44 ;18271          JMP [LD.EXP]                     ; and repeat branch sequence
                ;18272  NEITHER:
                ;18273          CMP WR[0] WITH LS[#4],           ;If not error 4
U 1B1F, 4F44,35 ;18274           DT(LONG)&SET.ALU.CC
U 1B20, 89B3,31 ;18275          JMP.IF[NEQ] TO [T2A.END]         ; then jump to the end of the test
```

```
U 1B21, B69C,95 ;18276          MOV LS[#0] TO WR[1]              ; else clear EXP COUT
U 1B22, BEA2,95 ;18277          MOV WR[1] TO LS[DATA.1]
U 1B23, 09B0,44 ;18278          JMP [LD.EXP]                     ; and repeat branch sequence
                ;18279
                ;18280
                ;18281  ; SET UP EXPECTED DATA FOR EXP COUT AND GRAND BRANCH TEST
                ;18282  ;
                ;18283  ;       This subroutine loads WR[1] with the expected data according to the
                ;18284  ;       the error number.
                ;18285
                ;18286  SET.UP.EXP:
U 1B24, B683,15 ;18287          MOV LS[ERROR.NUMBER] TO WR[2]    ;Move the error # to WR 2
                ;18288          CMP WR[2] WITH LS[#1],           ;Does the error # go with this
U 1B25, 4F41,35 ;18289           DT(LONG)&SET.ALU.CC             ;pattern?
U 1B26, 09B2,91 ;18290          JMP.IF[NEQ] TO [ERR.2]           ;If it doesn't go onto next error
U 1B27, 3742,95 ;18291          MOV LS[#303] TO WR[1]            ; else move expected data to WR 1
U 1B28, 5B00,14 ;18292          RETURN                           ;Return to calling routine
                ;18293  ERR.2:  CMP WR[2] WITH LS[#2],           ;Does the error # go with this
U 1B29, CF43,35 ;18294           DT(LONG)&SET.ALU.CC             ; pattern?
U 1B2A, 89B2,D1 ;18295          JMP.IF[NEQ] TO [ERR.3]           ;If it doesn't then go onto next error
U 1B2B, B73E,95 ;18296          MOV LS[#301] TO WR[1]            ; else move expected data to WR 1
U 1B2C, 5B00,14 ;18297          RETURN                           ;Return to calling routine
                ;18298  ERR.3:  CMP WR[2] WITH LS[#3(H)],        ;Does the error # go with this
U 1B2D, CFC1,35 ;18299           DT(LONG)&SET.ALU.CC             ; pattern?
U 1B2E, 09B3,11 ;18300          JMP.IF[NEQ] TO [ERR.4]           ;If it doesn't then go onto next error
U 1B2F, B740,95 ;18301          MOV LS[#302] TO WR[1]            ; else move expected data to WR 1
U 1B30, 5B00,14 ;18302          RETURN                           ;Return to calling routine
U 1B31, B716,95 ;18303  ERR.4:  MOV LS[#300] TO WR[1]            ;Move expected data to WR 1
U 1B32, 5B00,14 ;18304          RETURN                           ;Return to calling routine
                ;18305
                ;18306  T2A.END:
```

```
;18307  .page
;18308  .TOC    "TEST 2B SIGN OUT AND HUGE BRANCH TEST(M8389 FPA MODULE)"
;18309  ;++
;18310  ;
;18311  ;
;18312  ; Test Description:
;18313  ;
;18314  ;       The purpose of this test is to check the branch conditions, SIGN OUT
;18315  ;       and HUGE. In order to test these branch conditions the branch field
;18316  ;       must contain 00001. Since the control logic required to branch has
;18317  ;       been tested under previous branch conditions the logic that is being
;18318  ;       tested is the logic required to asserted the branch conditions. The
;18319  ;       data path will be as follows: SIGN OUT will be asserted by enabling
;18320  ;       1 into sign out by setting the A address lines to 1 in the 3 LSB's.
;18321  ;       HUGE will be asserted by issuing a force with bit 18 set from the CPU
;18322  ;       to set the SIZE bits to 11. Once the branch conditions have been set a
;18323  ;       branch instruction will be issued and regardless of whether there is a
;18324  ;       branch or not the next instruction will be a jump to self. This whole
;18325  ;       procedure will be repeated for the case where both branch conditions
;18326  ;       are not asserted.
;18327  ;
;18328  ; Assumptions:
;18329  ;
;18330  ;       It is assumed that the EXP COUT and GRAND branch condition test has
;18331  ;       been run and there are no errors.
;18332  ;
;18333  ; Test Steps:
;18334  ;
;18335  ;       1) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18336  ;          and the size bits. Set the SIGN OUT bit by enabling 1 into OP1 SIGN
;18337  ;          and enabling OP1 SIGN into SIGN OUT. Issue a nop with a branch
;18338  ;          instruction. Read the microaddress back to the CPU. If no branch or
;18339  ;          branch to the wrong place then issue error1.
;18340  ;       2) Repeat step1 except that the size bits should be loaded with 00 and
;18341  ;          SIGN OUT should be clear.
;18342  ;
;18343  ; Error:
;18344  ;
;18345  ;       Error1 - Either SIGN OUT or HUGE failed asserted.
;18346  ;       Error2 - Either SIGN OUT or HUGE failed unasserted.
;18347  ;
;18348  ; Debug:
;18349  ;
;18350  ;       Error1: Since the control logic has already been tested in a previous
;18351  ;               test the most likely cause of error is the branch conditions
;18352  ;               themselves. If the branch failed and branched ahead 3 instead
;18353  ;               4 then BRANCH 0 probably failed. If the branch failed and
;18354  ;               branched ahead 2 instead of 4 then BRANCH 1 probably failed.
;18355  ;               If BRANCH 0 failed then the BRANCH 0 PAL should be checked for
;18356  ;               error. If the PAL isn't the source of error then the UBCTL
;18357  ;               signals should be checked to be 00001 and the SIZE bits should
;18358  ;               be checked to be 11. If the UBCTL bits are in error then the
;18359  ;               latch at the top of page D should be checked for error. If the
;18360  ;               SIZE bits are in error then the etch between the MUX and the
;18361  ;               PAL should be checked for error.
```

```
                ;18362  ;       Error2: Same as error1 except that SIGN OUT is 0 and the SIZE bits are
                ;18363  ;               both 1.
                ;18364  ;
                ;18365  ;
                ;18366  ;--
                ;18367  T.2B:
U 1B33, B65E,15 ;18368          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND'
                ;18369                                          ; STATUS WORD
U 1B34, 3E80,15 ;18370          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18371                                          ; BIT 15 INDICATES START OF TEST TO
                ;18372                                          ; CONSOLE
U 1B35, 10E0,15 ;18373          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;18374  WAIT.T2B.0:
U 1B36, 89B3,64 ;18375          JMP [WAIT.T2B.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18376                                          ; RESPOND
U 1B37, 887E,EC ;18377          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18378                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1B38, 3746,15 ;18379          MOV LS[T2B.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1B39, BE12,15 ;18380          MOV WR[0] TO LS[T9]              ; references.
U 1B3A, B742,15 ;18381          MOV LS[#303] TO WR[0]           ;MOV expected data for error 1 to
U 1B3B, 3EA2,15 ;18382          MOV WR[0] TO LS[DATA.1]         ; data.1
U 1B3C, B748,15 ;18383          MOV LS[#70300] TO WR[0]         ;Set size bits to huge
U 1B3D, 90F6,15 ;18384          MISC2 [TRAP.ACC] WR[0]
U 1B3E, 8870,3C ;18385  T2B.2:  JSR [L0]                        ;Get SIGN OUT address to force
U 1B3F, 0870,8C ;18386          JSR [L1]                        ;Get branch address to force
U 1B40, BE14,15 ;18387          MOV WR[0] TO LS[T10]             ;Save registers fo loop on error
U 1B41, BE16,95 ;18388          MOV WR[1] TO LS[T11]
U 1B42, 3614,15 ;18389  T2B.1:  MOV LS[T10] TO WR[0]            ;Restore registers for loop on error
U 1B43, 3616,95 ;18390          MOV LS[T11] TO WR[1]
U 1B44, 90F6,15 ;18391          MISC2 [TRAP.ACC] WR[0]          ;Set SIGN OUT to 1 or 0
U 1B45, 10F6,95 ;18392          MISC2 [TRAP.ACC] WR[1]          ;Branch on SIGN OUT and GRAND
U 1B46, 10F8,15 ;18393          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 1B47, 3A18,15 ;18394          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 1B48, 3618,15 ;18395          MOV LS[T12] TO WR[0]            ;Set up received data
U 1B49, 36A2,95 ;18396          MOV LS[DATA.1] TO WR[1]         ;Set up expected data
U 1B4A, 0869,3C ;18397          JSR [CHECK.RESULT]              ;Report error if there is one
U 1B4B, 89B4,24 ;18398          JMP [T2B.1]                     ;Loop on error
U 1B4C, 3682,15 ;18399          MOV LS[ERROR.NUMBER] TO WR[0]   ;If error 2 has been checked for then
                ;18400          CMP LS[#2] WITH WR[0],          ; go on to test F
U 1B4D, CE42,35 ;18401           DT(LONG)&SET.ALU.CC
U 1B4E, 09B5,59 ;18402          JMP.IF[EQL] TO [T2B.END]
U 1B4F, FF82,15 ;18403          INC LS[ERROR.NUMBER]            ;Go on to next error
U 1B50, B718,15 ;18404          MOV LS[#00040300] TO WR[0]      ;Set size bits back to single
U 1B51, 90F6,15 ;18405          MISC2 [TRAP.ACC] WR[0]
U 1B52, 3716,15 ;18406          MOV LS[#300] TO WR[0]           ;Put expected data for error 2 in
U 1B53, 3EA2,15 ;18407          MOV WR[0] TO LS[DATA.1]         ; DATA.1
U 1B54, 09B3,E4 ;18408          JMP [T2B.2]                     ;Loop back for error 2.
                ;18409  T2B.END:
```

```
;18410  .page
;18411  .TOC    "TEST 2C CPU DATA AVAIL AND SINGLE BRANCH TEST(M8389 FPA MODULE)"
;18412  ;++
;18413  ;
;18414  ;
;18415  ; Test Description:
;18416  ;
;18417  ;       The purpose of this test is to check the CPU DATA AVAIL and SINGLE
;18418  ;       branch conditions. This can be done by setting the branch control
;18419  ;       field to 00010 and by asserting the two branch conditons. The data
;18420  ;       path will be to assert the two branch conditions, execute the branch
;18421  ;       and read the NUA back to the CPU to see if the branch worked. Then
;18422  ;       repeat the whole procedure with the two branch conditions unasserted.
;18423  ;
;18424  ; Assumptions:
;18425  ;
;18426  ;       It is assumed that the control of the branch logic has been tested
;18427  ;       previous to this test and is working.
;18428  ;
;18429  ; Test Steps:
;18430  ;
;18431  ;       1) Issue a FORCE to set bit 18 set to set the instruction encode bits
;18432  ;          and the size bits. Issue a nop FPA instruction with the branch on
;18433  ;          CPU DATA AVAIL. Issue a MISC instruction with bit 18 clear and
;18434  ;          the MISC FUNC field set to ASSERT DATA AVAIL and PASS WR TO ACC/
;18435  ;          PORT. If there is no branch or a branch to the wrong place then
;18436  ;          issue error1.
;18437  ;       2) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18438  ;          and the size bits. Issue a nop with the branch field set to 00010.
;18439  ;          If there is no branch or a branch to the wrong place then issue
;18440  ;          error2.
;18441  ;       3) Repeat step 2 except set the size bits to 11. CPU DATA AVAIL won't
;18442  ;          be set because the MISC instruction won't have been issued to enable
;18443  ;          the CPU DATA AVAIL. If there is any branch then issue error3.
;18444  ;
;18445  ; Error:
;18446  ;
;18447  ;       Error1 - Branch failed on CPU DATA AVAIL asserted.
;18448  ;       Error2 - Branch failed on SINGLE asserted.
;18449  ;       Error3 - Branch failed on SINGLE and CPU DATA AVAIL unasserted.
;18450  ;
;18451  ; Debug:
;18452  ;
;18453  ;       Error1: If the branch failed on CPU DATA AVAIL then it'd likely that
;18454  ;               the error is a result of the branch condition itself not being
;18455  ;               asserted. If that is the case then the BRANCH 1 PAL should be
;18456  ;               checked for error. If the PAL isn't the source of error then
;18457  ;               the UBCTL bits should be checked to be 00010, and CPU DATA
;18458  ;               AVAIL should be checked to be asserted. If the UBCTL bits are
;18459  ;               in error then the CS LATCH they came from should be checked
;18460  ;               for error. If CPU DATA AVAIL is in error then the line between
;18461  ;               the PAL on the FPA board and the DECODER on the CPU board
;18462  ;               should be checked for error.
;18463  ;       Error2: If the branch failed on SINGLE then it's likely that the error
;18464  ;               is a result of the branch condiiton itself not being asserted.
```

```
                ;18465  ;               If this is the case then the BRANCH 0 PAL should be checked for
                ;18466  ;               error. If the PAL isn't the cause of error then the UBCTL bit
                ;18467  ;               should be checked to be 00010 and the size bits should both be
                ;18468  ;               0. If the UBCTL bits are in error then the latch at the top of
                ;18469  ;               page D should be checked for error. If the size bits are in
                ;18470  ;               error then the etch between the PAL and the MUX should be
                ;18471  ;               checked for error.
                ;18472  ;       Error3: Same as both error1 and error2 except that both CPU DATA AVAIL
                ;18473  ;               and SINGLE are not asserted.
                ;18474  ;
                ;18475  ;
                ;18476  ;--
                ;18477  T.2C:
U 1B55, B65E,15 ;18478          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;18479                                          ; STATUS WORD
U 1B56, 3E80,15 ;18480          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18481                                          ; BIT 15 INDICATES START OF TEST TO
                ;18482                                          ; CONSOLE
U 1B57, 10E0,15 ;18483          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;18484  WAIT.T2C.0:
U 1B58, 09B5,84 ;18485          JMP [WAIT.T2C.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18486                                          ; RESPOND
U 1B59, 887E,EC ;18487          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18488                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1B5A, 374A,15 ;18489          MOV LS[T2C.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1B5B, BE12,15 ;18490          MOV WR[0] TO LS[T9]              ; references.
U 1B5C, 8870,3C ;18491          JSR [L0]                        ;Get branch address
U 1B5D, BE14,15 ;18492          MOV WR[0] TO LS[T10]             ;Save address for loop on error
U 1B5E, 3614,15 ;18493  T2C.1:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1B5F, 90F6,15 ;18494          MISC2 [TRAP.ACC] WR[0]          ;Force the branch on CPU DATA AVAIL and
                ;18495                                          ; SINGLE
U 1B60, 10F4,15 ;18496          MISC2 [ASSERT.DA.AND.PASS.WR]   ;Assert CPU DATA AVAIL
U 1B61, 10F8,15 ;18497          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1B62, 3A18,15 ;18498          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1B63, 3618,15 ;18499          MOV LS[T12] TO WR[0]            ;Set up received data
U 1B64, 3742,95 ;18500          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1B65, 0869,3C ;18501          JSR [CHECK.RESULT]              ;Report error if there is one
U 1B66, 09B5,E4 ;18502          JMP [T2C.1]                     ;Loop on error
U 1B67, FF82,15 ;18503          INC LS[ERROR.NUMBER]            ;Go on to next error
U 1B68, 3614,15 ;13504  T2C.2:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1B69, 90F6,15 ;18505          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and SINGLE
U 1B6A, 10F8,15 ;18506          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1B6B, 3A18,15 ;18507          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1B6C, 3618,15 ;18508          MOV LS[T12] TO WR[0]            ;Set up received data
U 1B6D, B73E,95 ;18509          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1B6E, 0869,3C ;18510          JSR [CHECK.RESULT]              ;Report error if there is one
U 1B6F, 09B6,84 ;18511          JMP [T2C.2]                     ;Loop on error
U 1B70, FF82,15 ;18512          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1B71, B748,15 ;18513          MOV LS[#70300] TO WR[0]         ;Set the size bits to huge
U 1B72, 90F6,15 ;18514          MISC2 [TRAP.ACC] WR[0]
U 1B73, 3614,15 ;18515  T2C.3:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1B74, 90F6,15 ;18516          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and SINGLE
U 1B75, 10F8,15 ;18517          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1B76, 3A18,15 ;18518          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1B77, 3618,15 ;18519          MOV LS[T12] TO WR[0]            ;Set up received data
```

```
U 1B78, B716,95 ;18520          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1B79, 0869,3C ;18521          JSR [CHECK.RESULT]             ;Report error if there is one
U 1B7A, 09B7,34 ;18522          JMP [T2C.3]                    ;Loop on error
U 1B7B, B718,15 ;18523          MOV LS[#00040300] TO WR[0]     ;Restore the size bits to their
U 1B7C, 90F6,15 ;18524          MISC2 [TRAP.ACC] WR[0]         ; original state
                ;18525  T2C.END:
```

```
;18526  .page
;18527  .TOC    "TEST 2D OPTION SYNC TEST(M8389 FPA MODULE, M8390 CPU MODULE)"
;18528  ;++
;18529  ;
;18530  ;
;18531  ; Test Description:
;18532  ;
;18533  ;       The purpose of this test is to check the OPTION SYNC signal. OPTION
;18534  ;       SYNC is a CPU - FPA interface control signal. OPTION SYNC will be
;18535  ;       asserted if the branch control bits are 2(H), 3(H), or 16(H). This
;18536  ;       test will check each of these three cases. Once OPTION SYNC is asserted
;18537  ;       the signal will go the CPU. Once in the CPU a nop with 1F in the skip
;18538  ;       control field will be issued. If there's no skip then an error will
;18539  ;       be printed else you will fall through. Then the loop if no interrupt
;18540  ;       and no acc sync will be checked by asserting the loop then falling out
;18541  ;       of the loop by issuing a CONS HALT interrupt. This will be repeated
;18542  ;       again for the case when ACC SYNC is asserted.
;18543  ;
;18544  ; Assumptions:
;18545  ;
;18546  ;       It is assumed that all the 2901 logic has been tested previously and
;18547  ;       has no errors. Since this is testing logic that hasn't been tested on
;18548  ;       either the CPU or FPA the error can occur on either of two boards.
;18549  ;
;18550  ; Test Steps:
;18551  ;
;18552  ;       1) Issue a MISC instruction that will force a nop with the branch field
;18553  ;          set to 00010. Issue a nop in the CPU with the skip field set to 1F.
;18554  ;          If there is no skip then issue error 1.
;18555  ;       2) Issue a MISC instruction that will force a nop with the branch field
;18556  ;          set to 00011. Issue a nop in the CPU with the skip field set to 1F.
;18557  ;          If there is no skip then issue error 2.
;18558  ;       3) Issue a MISC instruction that will force a nop with the branch field
;18559  ;          set to 10110. Issue a nop in the CPU with the skip field set to 1F.
;18560  ;          If there is no skip then issue error 3.
;18561  ;       4) Issue a JSR to the location following the next location. At the
;18562  ;          location jumped to issue a MISC isntruction to force an instruction
;18563  ;          to assert ACC SYNC. Issue a nop in the CPU with the skip field set
;18564  ;          to 17(H). This should cause no loop and a skip.The possible errors
;18565  ;          that could result from this are loop and no skip or no loop and no
;18566  ;          skip.
;18567  ;       5) Repeat instruction 4 with SYNC unasserted this should cause a loop
;18568  ;          but no skip. There are two ways this could fail no loop and no
;18569  ;          no skip or no loop and a skip.
;18570  ;
;18571  ; Error:
;18572  ;
;18573  ;       Error1 - OPTION SYNC failed to be asserted when the branch field was
;18574  ;                00010.
;18575  ;       Error2 - OPTION SYNC failed to be asserted when the branch faild was
;18576  ;                00011.
;18577  ;       Error3 - OPTION SYNC failed to be asserted when the branch field was
;18578  ;                10110.
;18579  ;       Error4 - No loop and no skip occured when there should have been loop
;18580  ;                and no skip.
```

```
                ;18581  ;       Error5 - No Loop and skip occured when there should have been a loop
                ;18582  ;                and no skip.
                ;18583  ;       Error6 - Loop and no skip occured when there should have been a no loop
                ;18584  ;                and skip.
                ;18585  ;       Error7 - No loop and no skip occured when there should have been no
                ;18586  ;                loop and no skip.
                ;18587  ;
                ;18588  ; Debug:
                ;18589  ;
                ;18590  ;       Error1: If this error occurs then the failure can occur in either the
                ;18591  ;               FPA or the CPU. If the error occurs in the FPA then the BRANCH
                ;18592  ;               2 PAL should be checked for error. If the PAL is not the cause
                ;18593  ;               of the error then the latches that the branch control bits come
                ;18594  ;               from should be checked for error. If the CPU is the cause of
                ;18595  ;               error then the USEQ. CTL PAL should be checked for error. If
                ;18596  ;               the PAL is no the cause of error then the NOR gate from the PAL
                ;18597  ;               should be checked for error.
                ;18598  ;       Error2: Same as error 1 except that the branch field in the FPA was
                ;18599  ;               00011.
                ;18600  ;       Error3: Same as error 1 except that the branch field in the FPA was
                ;18601  ;               10110.
                ;18602  ;       Error4: If this error occurs and the previous three have not occured
                ;18603  ;               then it's likely that the error is in the CPU. This being the
                ;18604  ;               case the USEQ. CTL PAL should be checked for error. If the PAL
                ;18605  ;               is not the cause of error then the AND and NOT gates below the
                ;18606  ;               PAL should be checked for error. If these are not the cause of
                ;18607  ;               error then the USTACK POINTER PAL and the disable to the STACK
                ;18608  ;               RAM should be checked for error. The STACK ADDER and MUX could
                ;18609  ;               also be sources of error. If this error occurs then Errors 5, 6
                ;18610  ;               and 7 will not be tested for.
                ;18611  ;       Error5: Same as error 4. If this error occurs then Errors 6 and 7 will
                ;18612  ;               not be tested for.
                ;18613  ;       Error6: Same as error 4.
                ;18614  ;       Error7: Same as error4.
                ;18615  ;
                ;18616  ;
                ;18617  ;--
                ;18618  T.2D:
U 1B7D, 3752,15 ;18619          MOV LS[BIT16.BEGIN.TEST] TO WR[0];SET BIT 15 AND BIT 16 IN WR[0] FOR
                ;18620                                          ;CONTROL AND STATUS WORD
U 1B7E, 3E80,15 ;18621          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18622                                          ; BIT 15 INDICATES START OF TEST TO
                ;18623                                          ; CONSOLE
U 1B7F, 10E0,15 ;18624          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;18625  WAIT.T2D.0:
U 1B80, 09B8,04 ;18626          JMP [WAIT.T2D.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18627                                          ; RESPOND
U 1B81, 087E,6C ;18628          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18629                                          ; AND SIZE BITS
U 1B82, B750,15 ;18630          MOV LS[#1F000] TO WR[0]         ;DISABLE REMAINING INTERRUPTS
U 1B83, BFFE,15 ;18631          MOV WR[0] TO LS[PSL.HW]
U 1B84, B646,15 ;18632          MOV LS[FPA] TO WR[0]            ;Store module code in LS to indicate
U 1B85, 4742,15 ;18633          BIS LS[CPU] TO WR[0]            ; FPA and CPU boards.
U 1B86, 3E8C,15 ;18634          MOV WR[0] TO LS[MODULE.NUM]
U 1B87, B66E,15 ;18635          MOV LS[BIT23] TO WR[0]          ;Set expected and recieved to N/A
```

```
U 1B88, 3E80,15 ;18636          MOV WR[0] TO LS[CONTROL.STATUS]
U 1B89, B74E,15 ;18637          MOV LS[T2D.POINTER] TO WR[0]     ;Set up memory pointer
U 1B8A, BE12,15 ;18638          MOV WR[0] TO LS[T9]
U 1B8B, 8870,3C ;18639  T2D.2:  JSR [L0]                         ;Get branch instruction
U 1B8C, BE14,15 ;18640          MOV WR[0] TO LS[T10]              ;Save address for loop on error
U 1B8D, 3614,15 ;18641  T2D.1:  MOV LS[T10] TO WR[0]              ;Restore address for loop on error
U 1B8E, 90F6,15 ;18642          MISC2 [TRAP.ACC] WR[0]           ;Force the branch that will assert
                ;18643                                           ; OPTION SYNC
U 1B8F, DB00,1F ;18644          SKIP.IF[SYNC]                    ;Branch in the CPU on SYNC asserted
U 1B90, 09B9,24 ;18645          JMP [T2D.ERROR]                  ;If there's an error then go to error
U 1B91, 09B9,74 ;18646          JMP [NXT.ERROR]                  ;If there's no error go on to NXT.ERROR
                ;18647  T2D.ERROR:
U 1B92, 2F80,15 ;18648          CLR WR[0]                        ;Set up phony data
U 1B93, 2F82,95 ;18649          CLR WR[1]
U 1B94, 2042,95 ;18650          INC WR[1]
U 1B95, 0869,3C ;18651          JSR [CHECK.RESULT]               ;Report error
U 1B96, 89B8,D4 ;18652          JMP [T2D.1]                      ;Loop on error
                ;18653  NXT.ERROR:
U 1B97, 36C0,15 ;18654          MOV LS[#3(H)] TO WR[0]           ;If all branches tested go on to loop
                ;18655          CMP WR[0] WITH LS[ERROR.NUMBER],; test, else get next branch
U 1B98, 4F82,35 ;18656           DT(LONG)&SET.ALU.CC
U 1B99, 09B9,C9 ;18657          JMP.IF[EQL] TO [LOOP.TEST.0]
U 1B9A, FF82,15 ;18658          INC LS[ERROR.NUMBER]             ;Increase the error number by one
U 1B9B, 89B8,B4 ;18659          JMP [T2D.2]                      ;Get next branch
                ;18660  LOOP.TEST.0:
U 1B9C, 8870,3C ;18661          JSR [L0]
U 1B9D, BE14,15 ;18662          MOV WR[0] TO LS[T10]              ;Set up branch to keep SYNC asserted
                ;18663  LOOP.TEST:
U 1B9E, 89BA,7C ;18664          JSR [T2D.5]                      ;Set up the stack
U 1B9F, DB00,1F ;18665          SKIP.IF[SYNC]                    ;If SYNC is asserted then skip
U 1BA0, 5B00,1E ;18666          SKIP                             ;Skip the next instruction
U 1BA1, 09BB,04 ;18667          JMP [T2D.6]                      ;Report error 2
U 1BA2, 3614,15 ;18668          MOV LS[T10] TO WR[0]             ;Assert SYNC
U 1BA3, 90F6,15 ;18669          MISC2 [TRAP.ACC] WR[0]
U 1BA4, 5B00,17 ;18670          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 1BA5, 89BB,74 ;18671          JMP [SKIP.SYNC]                  ;If there is no skip and there should
                ;18672                                           ; be then report SKIP.SYNC error
U 1BA6, 09BC,44 ;18673          JMP [T2D.END]                    ; else jump to end of test
U 1BA7, 5B00,17 ;18674  T2D.5:  LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 1BA8, 89BB,D4 ;18675          JMP [NO.LOOP.NO.SKIP]            ;If the loop failed and there's no
                ;18676                                           ; skip then go the NO.LOOP.NO.SKIP
U 1BA9, B712,15 ;18677          MOV LS[#5] TO WR[0]              ;Set up error number
U 1BAA, BE82,15 ;18678          MOV WR[0] TO LS[ERROR.NUMBER]
U 1BAB, 2F80,15 ;18679          CLR WR[0]                        ;Set up phony data
U 1BAC, 3640,95 ;18680          MOV LS[#1] TO WR[1]
U 1BAD, 0869,3C ;18681          JSR [CHECK.RESULT]
U 1BAE, 09B9,E4 ;18682          JMP [LOOP.TEST]                  ;Loop on error
U 1BAF, 09BC,44 ;18683          JMP [T2D.END]
U 1BB0, B714,15 ;18684  T2D.6:  MOV LS[#6] TO WR[0]              ;Set up error number
U 1BB1, BE82,15 ;18685          MOV WR[0] TO LS[ERROR.NUMBER]
U 1BB2, 2F80,15 ;18686          CLR WR[0]                        ;Set up phony data
U 1BB3, 3640,95 ;18687          MOV LS[#1] TO WR[1]
U 1BB4, 0869,3C ;18688          JSR [CHECK.RESULT]
U 1BB5, 09B9,E4 ;18689          JMP [LOOP.TEST]                  ;Loop on error
U 1BB6, 09BC,44 ;18690          JMP [T2D.END]
```

```
                ;18691  SKIP.SYNC:
U 1BB7, 3720,15 ;18692          MOV LS[#7] TO WR[0]              ;Set up error number
U 1BB8, BE82,15 ;18693          MOV WR[0] TO LS[ERROR.NUMBER]
U 1BB9, 2F80,15 ;18694          CLR WR[0]                        ;Set up phony data
U 1BBA, 3640,95 ;18695          MOV LS[#1] TO WR[1]
U 1BBB, 0869,3C ;18696          JSR [CHECK.RESULT]
U 1BBC, 09B9,E4 ;18697          JMP [LOOP.TEST]                  ;Loop on error
                ;18698  NO.LOOP.NO.SKIP:
U 1BBD, 3644,15 ;18699          MOV LS[#4] TO WR[0]              ;Set up error number
U 1BBE, BE82,15 ;18700          MOV WR[0] TO LS[ERROR.NUMBER]
U 1BBF, 2F80,15 ;18701          CLR WR[0]                        ;Set up phony data
U 1BC0, 3640,95 ;18702          MOV LS[#1] TO WR[1]
U 1BC1, 0869,3C ;18703          JSR [CHECK.RESULT]
U 1BC2, 09B9,E4 ;18704          JMP [LOOP.TEST]                  ;Loop on error
U 1BC3, 09BC,44 ;18705          JMP [T2D.END]
                ;18706  T2D.END:
```

```
;18707  .page
;18708  .TOC    "TEST 2E CPU DATA AVAIL AND ADD + SUB BRANCH TEST(M8389 FPA MODULE)"
;18709  ;++
;18710  ;
;18711  ;
;18712  ; Test Description:
;18713  ;
;18714  ;       The purpose of this test is to check the branch conditions, CPU DATA
;18715  ;       AVAIL and ADD + SUB when 00011 is in the branch control field. Since
;18716  ;       CPU DATA AVAIL has already been tested this test will serve only to
;18717  ;       show that the branch condition works when the branch field is different
;18718  ;       The second branch condition that is tested is the ADD + SUB which is a
;18719  ;       branch on any of the of the first seven instructions defined by the
;18720  ;       instruction encode bits. The seven instructions will be loaded onto the
;18721  ;       instruction encode bits by enabling the IB BUS into the FPA one at a
;18722  ;       time and be tested for ability to assert the branch condition. Once the
;18723  ;       branch conditions have been tested asserted then they will be tested
;18724  ;       unasserted. CPU DATA AVAIL should also cause ACC SYNC to be asserted
;18725  ;       which is a skip condition in the CPU. This will be tested by asserting
;18726  ;       ACC SYNC and issuing a nop from the CPU with the skip field set to
;18727  ;       skip on ACC SYNC.
;18728  ;
;18729  ; Assumptions:
;18730  ;
;18731  ;       It is assumed that the control logic required for branch instructions
;18732  ;       has been tested previously and works.
;18733  ;
;18734  ; Test Steps:
;18735  ;
;18736  ;       1) Issue a MISC instruction with bit 18 clear and the MISC FUNC field
;18737  ;          set to ASSERT DATA AVAIL and PASS WR TO ACC/PORT. Then force a nop
;18738  ;          instruction with branch on CPU DATA AVAIL and ADD + SUB. The in-
;18739  ;          struction encode bits should be a number greater than 00111. There
;18740  ;          should be a branch to three instructions ahead. If there is an error
;18741  ;          in the branch then issue error1.
;18742  ;       2) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18743  ;          and the size bits. Issue a nop instruction with a branch on ADD +
;18744  ;          SUB. If the branch fails then issue error2. The instruction encode
;18745  ;          bits should be 00000.
;18746  ;       3) Repeat step 2 except the instruction encode bits should be 00001.
;18747  ;       4) Repeat step 2 except the instruction encode bits should be 00010.
;18748  ;       5) Repeat step 2 except the instruction encode bits should be 00011.
;18749  ;       6) Repeat step 2 except the instruction encode bits should be 00100.
;18750  ;       7) Repeat step 2 except the instruction encode bits should be 01000.
;18751  ;
;18752  ; Error:
;18753  ;
;18754  ;       Error1 - Branch failed on CPU DATA AVAIL asserted and ADD + SUB unas-
;18755  ;                serted.
;18756  ;       Error2 - Branch failed on ADD + SUB asserted.
;18757  ;       Error3 - Branch failed on ADD + SUB asserted.
;18758  ;       Error4 - Branch failed on ADD + SUB asserted.
;18759  ;       Error5 - Branch failed on ADD + SUB asserted.
;18760  ;       Error6 - Branch failed on ADD + SUB unasserted.
;18761  ;       Error7 - Branch failed on ADD + SUB unasserted.
```

```
                ;18762  ;
                ;18763  ; Debug:
                ;18764  ;
                ;18765  ;       Error1: If this error occurs then it's likely that either the BRANCH 1
                ;18766  ;               PAL is in error or the UBCTL bits are in error. If the PAL is
                ;18767  ;               not in error then the UBCTL bits are probably the source of the
                ;18768  ;               because the CPA DATA AVAIL bit has been tested previous to this
                ;18769  ;       Error2: If this error occurs the BRANCH 0 PAL should be be checked for
                ;18770  ;               error. If the PAL isn't the cause of the error then it's likely
                ;18771  ;               that the instruction encode bits are the cause of error if
                ;18772  ;               error 1 didn't occur because the UBCTL bits have been tested in
                ;18773  ;               error1. The instruction encode bits should be 0's. If the
                ;18774  ;               instruction encode bits are in error then the IRD MUX's
                ;18775  ;               should be checked for error. If the MUX's aren't the cause of
                ;18776  ;               error then the IRD ROM should be checked for error.
                ;18777  ;       Error3: Same as error2 except that the instruction encode bits should
                ;18778  ;               be 00001.
                ;18779  ;       Error4: Same as error2 except that the instruction encode bits should
                ;18780  ;               be 00010.
                ;18781  ;       Error5: Same as error2 except that the instruction encode bits should
                ;18782  ;               be 00011.
                ;18783  ;       Error6: Same as error2 except that the instruction encode bits should
                ;18784  ;               be 00100.
                ;18785  ;       Error7: Same as error2 except that the instruction encode bits should
                ;18786  ;               be 01000.
                ;18787  ;
                ;18788  ;
                ;18789  ;--
                ;18790  T.2E:
U 1BC4, B65E,15 ;18791          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;18792                                          ; STATUS WORD
U 1BC5, 3E80,15 ;18793          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18794                                          ; BIT 15 INDICATES START OF TEST TO
                ;18795                                          ; CONSOLE
U 1BC6, 10E0,15 ;18796          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;18797  WAIT.T2E.0:
U 1BC7, 09BC,74 ;18798          JMP [WAIT.T2E.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18799                                          ; RESPOND
U 1BC8, 887E,EC ;18800          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18801                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1BC9, 375E,15 ;18802          MOV LS[T2E.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1BCA, BE12,15 ;18803          MOV WR[0] TO LS[T9]              ; references.
U 1BCB, 8870,3C ;18804          JSR [L0]                        ;Get branch address
U 1BCC, BE14,15 ;18805          MOV WR[0] TO LS[T10]             ;Save address for loop on error
U 1BCD, 3754,15 ;18806          MOV LS[#44300] TO WR[0]         ;Make instruction bits not ADD+SUB
U 1BCE, 90F6,15 ;18807          MISC2 [TRAP.ACC] WR[0]
U 1BCF, 3614,15 ;18808  T2E.1:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1BD0, 90F6,15 ;18809          MISC2 [TRAP.ACC] WR[0]          ;Force the branch on CPU DATA AVAIL and
                ;18810                                          ; ADD+SUB
U 1BD1, 10F4,15 ;18811          MISC2 [ASSERT.DA.AND.PASS.WR]   ;Assert CPU DATA AVAIL
U 1BD2, 10F8,15 ;18812          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1BD3, 3A18,15 ;18813          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1BD4, 3618,15 ;18814          MOV LS[T12] TO WR[0]            ;Set up received data
U 1BD5, B740,95 ;18815          MOV LS[#302] TO WR[1]           ;Set up expected data
U 1BD6, 0869,3C ;18816          JSR [CHECK.RESULT]              ;Report error if there is one
```

```
U 1BD7, 89BC,F4 ;18817          JMP [T2E.1]                     ;Loop on error
U 1BD8, FF82,15 ;18818          INC LS[ERROR.NUMBER]            ;Go on to next error
U 1BD9, B718,15 ;18819          MOV LS[#00040300] TO WR[0]      ;Set instruction bits to ADD+SUB
U 1BDA, 90F6,15 ;18820          MISC2 [TRAP.ACC] WR[0]
U 1BDB, 3614,15 ;18821  T2E.2:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1BDC, 90F6,15 ;18822          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1BDD, 10F8,15 ;18823          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1BDE, 3A18,15 ;18824          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1BDF, 3618,15 ;18825          MOV LS[T12] TO WR[0]            ;Set up received data
U 1BE0, B73E,95 ;18826          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1BE1, 0869,3C ;18827          JSR [CHECK.RESULT]              ;Report error if there is one
U 1BE2, 89BD,B4 ;18828          JMP [T2E.2]                     ;Loop on error
U 1BE3, FF82,15 ;18829          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1BE4, B756,15 ;18830          MOV LS[#40700] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1BE5, 90F6,15 ;18831          MISC2 [TRAP.ACC] WR[0]
U 1BE6, 3614,15 ;18832  T2E.3:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1BE7, 90F6,15 ;18833          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1BE8, 10F8,15 ;18834          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1BE9, 3A18,15 ;18835          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1BEA, 3618,15 ;18836          MOV LS[T12] TO WR[0]            ;Set up received data
U 1BEB, B73E,95 ;18837          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1BEC, 0869,3C ;18838          JSR [CHECK.RESULT]              ;Report error if there is one
U 1BED, 09BE,64 ;18839          JMP [T2E.3]                     ;Loop on error
U 1BEE, FF82,15 ;18840          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1BEF, 3758,15 ;18841          MOV LS[#40B00] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1BF0, 90F6,15 ;18842          MISC2 [TRAP.ACC] WR[0]
U 1BF1, 3614,15 ;18843  T2E.4:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1BF2, 90F6,15 ;18844          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1BF3, 10F8,15 ;18845          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1BF4, 3A18,15 ;18846          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1BF5, 3618,15 ;18847          MOV LS[T12] TO WR[0]            ;Set up received data
U 1BF6, B73E,95 ;18848          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1BF7, 0869,3C ;18849          JSR [CHECK.RESULT]              ;Report error if there is one
U 1BF8, 09BF,14 ;18850          JMP [T2E.4]                     ;Loop on error
U 1BF9, FF82,15 ;18851          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1BFA, B75A,15 ;18852          MOV LS[#40F00] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1BFB, 90F6,15 ;18853          MISC2 [TRAP.ACC] WR[0]
U 1BFC, 3614,15 ;18854  T2E.5:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1BFD, 90F6,15 ;18855          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1BFE, 10F8,15 ;18856          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1BFF, 3A18,15 ;18857          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1C00, 3618,15 ;18858          MOV LS[T12] TO WR[0]            ;Set up received data
U 1C01, B73E,95 ;18859          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1C02, 0869,3C ;18860          JSR [CHECK.RESULT]              ;Report error if there is one
U 1C03, 89BF,C4 ;18861          JMP [T2E.5]                     ;Loop on error
U 1C04, FF82,15 ;18862          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1C05, B75C,15 ;18863          MOV LS[#41300] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1C06, 90F6,15 ;18864          MISC2 [TRAP.ACC] WR[0]
U 1C07, 3614,15 ;18865  T2E.6:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1C08, 90F6,15 ;18866          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1C09, 10F8,15 ;18867          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1C0A, 3A18,15 ;18868          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1C0B, 3618,15 ;18869          MOV LS[T12] TO WR[0]            ;Set up received data
U 1C0C, B716,95 ;18870          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1C0D, 0869,3C ;18871          JSR [CHECK.RESULT]              ;Report error if there is one
```

```
U 1C0E, 89C0,74 ;18872          JMP [T2E.6]                     ;Loop on error
U 1C0F, FF82,15 ;18873          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1C10, 3762,15 ;18874          MOV LS[#42300] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1C11, 90F6,15 ;18875          MISC2 [TRAP.ACC] WR[0]
U 1C12, 3614,15 ;18876  T2E.7:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1C13, 90F6,15 ;18877          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1C14, 10F8,15 ;18878          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1C15, 3A18,15 ;18879          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1C16, 3618,15 ;18880          MOV LS[T12] TO WR[0]            ;Set up received data
U 1C17, B716,95 ;18881          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1C18, 0869,3C ;18882          JSR [CHECK.RESULT]              ;Report error if there is one
U 1C19, 09C1,24 ;18883          JMP [T2E.7]                     ;Loop on error
                ;18884  T2E.END:
```

```
;18885  .page
;18886  .TOC    "TEST 2F FRAC COUT AND EXT FUNC BRANCH TEST(M8389 FPA MODULE)"
;18887  ;++
;18888  ;
;18889  ;
;18890  ; Test Description:
;18891  ;
;18892  ;       The purpose of this test is to check the FRAC COUT and EXT FUNC branch
;18893  ;       conditions. This will be done by setting the two branch conditions then
;18894  ;       exacuting a nop with 00011 in the branch field. FRAC COUT can be set by
;18895  ;       adding two WR together that have 1's in the bit 55 of the fraction data
;18896  ;       path. EXT FUNC can be set if the size bits are either grand or huge.
;18897  ;       Once the branch conditions have been tested asserted they will be
;18898  ;       tested unasserted.
;18899  ;
;18900  ; Assumptions:
;18901  ;
;18902  ;       It is assumed that the branch control logic has been tested and does
;18903  ;       work.
;18904  ;
;18905  ; Test Steps:
;18906  ;
;18907  ;       1) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18908  ;          and the size bits. Load two WR with 1's in bit 55 of the fraction
;18909  ;          data path. Issue an add of the two WR to set FRAC COUT with the
;18910  ;          branch field set to 00100. If there is no branch or a branch to the
;18911  ;          wrong place then issue error1.
;18912  ;       2) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18913  ;          and the size bits. Issue a nop instruction with 00100 in the branch
;18914  ;          field. If there is no branch or a branch to the wrong place then
;18915  ;          issue error2.
;18916  ;       3) Issue a FORCE to with bit 18 set to set the instruction encode bits
;18917  ;          and the size bits. Load two WR's with 0's and issue an OR to clear
;18918  ;          FRAC COUT, the branch field of this instruction should be 00100. If
;18919  ;          there is a branch then issue error3.
;18920  ;
;18921  ; Error:
;18922  ;
;18923  ;       Error1 - Branch failed when FRAC COUT and EXT FUNC asserted with 10 in
;18924  ;                the size bits.
;18925  ;       Error2 - Branch failed when FRAC COUT and EXT FUNC asserted with 11 in
;18926  ;                the size bits.
;18927  ;       Error3 - Branch failed when FRAC COUT and EXT FUNC unasserted.
;18928  ;
;18929  ; Debug:
;18930  ;
;18931  ;       Error1: If this error occurs because FRAC COUT was not asserted then
;18932  ;               the BRANCH 1 PAL should be checked for error. If the PAL is
;18933  ;               not the cause of the error then the STATUS REG 1 LATCH
;18934  ;               should be checked for error. If the latch isn't the cause of
;18935  ;               error then the carry out of the 2901 should be checked for
;18936  ;               error. If the error occurs because EXT FUNC is in error then
;18937  ;               the BRANCH 0 PAL should be checked for error. If the PAL isn't
;18938  ;               the cause of the error then the size bits should be checked to
;18939  ;               be 10. If the size bits are in error then the etch between the
```

```
                ;18940  ;               MUX and the PAL should be checked for error. In the case of
                ;18941  ;               both PALs the UBCTL bits should be checked to be 00100. If the
                ;18942  ;               UBCTL bits are in error then the CS latch they came from should
                ;18943  ;               be checked for error.
                ;18944  ;       Error2: Same as error1 except that the size bits should be 11 and FRAC
                ;18945  ;               COUT should be unasserted.
                ;18946  ;       Error3: Same as error1 except that the size bits should be 00 and FRAC
                ;18947  ;               COUT should be unasserted.
                ;18948  ;
                ;18949  ;
                ;18950  ;       T10 - Load FWR[0]
                ;18951  ;       T14 - MOV FWR[0] TO FWR[2]
                ;18952  ;       T15 - ADD FWR[0] TO FWR[2]
                ;18953  ;       T13 - BRANCH ON FCOUT AND EXT FUNC
                ;18954  ;       T12 - CLR FWR[0]
                ;18955  ;
                ;18956  ;--
                ;18957  T.2F:
U 1C1A, B65E,15 ;18958          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;18959                                          ; STATUS WORD
U 1C1B, 3E80,15 ;18960          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;18961                                          ; BIT 15 INDICATES START OF TEST TO
                ;18962                                          ; CONSOLE
U 1C1C, 10E0,15 ;18963          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;18964  WAIT.T2F.0:
U 1C1D, 09C1,D4 ;18965          JMP [WAIT.T2F.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;18966                                          ; RESPOND
U 1C1E, 887E,EC ;18967          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;18968                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1C1F, B760,15 ;18969          MOV LS[T2F.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1C20, BE12,15 ;18970          MOV WR[0] TO LS[T9]              ; references.
U 1C21, B72E,15 ;18971          MOV LS[#2A7] TO WR[0]           ;Set up for load
U 1C22, BE14,15 ;18972          MOV WR[0] TO LS[T10]
U 1C23, 3732,15 ;18973          MOV LS[#368] TO WR[0]           ;Set up for MOV
U 1C24, C152,15 ;18974          SUB LS[#200] FROM WR[0]
U 1C25, 3E1C,15 ;18975          MOV WR[0] TO LS[T14]
U 1C26, 8870,3C ;18976          JSR [L0]                        ;Set up for ADD
U 1C27, BE1E,15 ;18977          MOV WR[0] TO LS[T15]
U 1C28, 8870,3C ;18978          JSR [L0]                        ;Set up for branch
U 1C29, 3E1A,15 ;18979          MOV WR[0] TO LS[T13]
U 1C2A, 8870,3C ;18980          JSR [L0]                        ;Set up for CLR
U 1C2B, BE18,15 ;18981          MOV WR[0] TO LS[T12]
U 1C2C, B744,15 ;18982          MOV LS[#60300] TO WR[0]         ;Set size bits to grand
U 1C2D, 90F6,15 ;18983          MISC2 [TRAP.ACC] WR[0]
U 1C2E, B69E,15 ;18984  T2F.1:  MOV LS[#FFFFFFFF] TO WR[0]      ;Set up data to be passed
U 1C2F, 3EA2,15 ;18985          MOV WR[0] TO LS[DATA.1]
U 1C30, 8877,DC ;18986          JSR [LOAD.DATA]                 ;Set bit 55 in FWR[0]
U 1C31, 0878,AC ;18987          JSR [MOV.DATA]                  ;Set bit 55 in FWR[2]
U 1C32, 361E,15 ;18988          MOV LS[T15] TO WR[0]            ;Add FWR[0] to FWR[2]
U 1C33, 361A,95 ;18989          MOV LS[T13] TO WR[1]            ;And branch on the results
U 1C34, 90F6,15 ;18990          MISC2 [TRAP.ACC] WR[0]
U 1C35, 10F6,95 ;18991          MISC2 [TRAP.ACC] WR[1]
U 1C36, 10F8,15 ;18992          MISC2 [READ.ACC.UPC]            ;Enable the NUA lines in the FPA BUS
U 1C37, BA16,15 ;18993          MOV FPA TO LS[T11]              ;Enable the FPA BUS onto the Y BUS
U 1C38, B616,15 ;18994          MOV LS[T11] TO WR[0]            ;Set up received data
```

```
U 1C39, 3742,95 ;18995          MOV LS[#303] TO WR[1]          ;Set up expected data
U 1C3A, 0869,3C ;18996          JSR [CHECK.RESULT]             ;Check for error
U 1C3B, 09C2,E4 ;18997          JMP [T2F.1]                    ;Loop on error
U 1C3C, FF82,15 ;18998          INC LS[ERROR.NUMBER]           ;Go onto next error
U 1C3D, B748,15 ;18999          MOV LS[#70300] TO WR[0]        ;Set size bits to huge
U 1C3E, 90F6,15 ;19000          MISC2 [TRAP.ACC] WR[0]
U 1C3F, B69E,15 ;19001  T2F.2:  MOV LS[#FFFFFFFF] TO WR[0]     ;Set up data to be passed
U 1C40, 3EA2,15 ;19002          MOV WR[0] TO LS[DATA.1]
U 1C41, 8877,DC ;19003          JSR [LOAD.DATA]                ;Set bit 55 in FWR[0]
U 1C42, 0878,AC ;19004          JSR [MOV.DATA]                 ;Set bit 55 in FWR[2]
U 1C43, 361E,15 ;19005          MOV LS[T15] TO WR[0]           ;Add FWR[0] to FWR[2]
U 1C44, 361A,95 ;19006          MOV LS[T13] TO WR[1]           ;And branch on the results
U 1C45, 90F6,15 ;19007          MISC2 [TRAP.ACC] WR[0]
U 1C46, 10F6,95 ;19008          MISC2 [TRAP.ACC] WR[1]
U 1C47, 10F8,15 ;19009          MISC2 [READ.ACC.UPC]           ;Enable nua lines onto the FPA BUS
U 1C48, BA16,15 ;19010          MOV FPA TO LS[T11]             ;Enable the FPA BUS onto the Y BUS
U 1C49, B616,15 ;19011          MOV LS[T11] TO WR[0]           ;Set up received data
U 1C4A, 3742,95 ;19012          MOV LS[#303] TO WR[1]          ;Set up expected data
U 1C4B, 0869,3C ;19013          JSR [CHECK.RESULT]             ;Check for error
U 1C4C, 09C3,F4 ;19014          JMP [T2F.2]                    ;Loop on error
U 1C4D, FF82,15 ;19015          INC LS[ERROR.NUMBER]           ;Go on to next error
U 1C4E, B718,15 ;19016          MOV LS[#00040300] TO WR[0]     ;Set the size bits to single
U 1C4F, 90F6,15 ;19017          MISC2 [TRAP.ACC] WR[0]
U 1C50, 3618,15 ;19018  T2F.3:  MOV LS[T12] TO WR[0]           ;CLR FWR[0]
U 1C51, B716,95 ;19019          MOV LS[#300] TO WR[1]
U 1C52, 90F6,15 ;19020          MISC2 [TRAP.ACC] WR[0]
U 1C53, 10F6,95 ;19021          MISC2 [TRAP.ACC] WR[1]
U 1C54, B61C,15 ;19022          MOV LS[T14] TO WR[0]           ;MOV FWR[0] TO FWR[2]
U 1C55, 90F6,15 ;19023          MISC2 [TRAP.ACC] WR[0]
U 1C56, 10F6,95 ;19024          MISC2 [TRAP.ACC] WR[1]
U 1C57, 361E,15 ;19025          MOV LS[T15] TO WR[0]           ;Add FWR[0] TO FWR[2]
U 1C58, 90F6,15 ;19026          MISC2 [TRAP.ACC] WR[0]
U 1C59, 10F6,95 ;19027          MISC2 [TRAP.ACC] WR[1]
U 1C5A, B61A,15 ;19028          MOV LS[T13] TO WR[0]           ;Branch on FCOUT AND EXT FUNC
U 1C5B, 90F6,15 ;19029          MISC2 [TRAP.ACC] WR[0]
U 1C5C, 10F8,15 ;19030          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS on to the FPA BUS
U 1C5D, BA16,15 ;19031          MOV FPA TO LS[T11]             ;Enable the FPA BUS onto the Y BUS
U 1C5E, B616,15 ;19032          MOV LS[T11] TO WR[0]           ;Set up received data
U 1C5F, B716,95 ;19033          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1C60, 0869,3C ;19034          JSR [CHECK.RESULT]             ;Check for error
U 1C61, 09C5,04 ;19035          JMP [T2F.3]                    ;Loop on error
                ;19036  T2F.END:
```

```
;19037  .page
;19038  .TOC    "TEST 30 OP1 SIGN AND EMOD BRANCH TEST(M8389 FPA MODULE)"
;19039  ;++
;19040  ;
;19041  ;
;19042  ; Test Description:
;19043  ;
;19044  ;       The purpose of this test will be to test the branch conditions OP1 SIGN
;19045  ;       and EMOD. The data path will be as follows: the branch conditions will
;19046  ;       be asserted then a nop will be issued with the branch field set to
;19047  ;       00101. Then the branch condition will be tested unasserted. OP1 SIGN
;19048  ;       will be asserted by enabling bit 15 of the FPA BUS into OP1 SIGN. EMOD
;19049  ;       will be set if the instruction encode bits are 00111.
;19050  ;
;19051  ; Assumptions:
;19052  ;
;19053  ;       It is assumed that the branch control logic has been tested and works.
;19054  ;
;19055  ; Test Steps:
;19056  ;
;19057  ;       1) Issue a FORCE instruction with bit 18 set to set the instruction
;19058  ;          encode bits and the size bits. Enable 1 into OP1 SIGN by loading the
;19059  ;          the first float with the sign bit set and the clock field set to 111
;19060  ;          the mod field set to nop. Issue a nop instruction with 00101 in the
;19061  ;          branch field. If there is no branch or an incorrect branch then
;19062  ;          issue error1.
;19063  ;       2) Issue a FORCE instruction with bit 18 set to set the instruction
;19064  ;          encode bits and the size bits. Enable 0 into OP1 SIGN by loading the
;19065  ;          the first float with the sign bit clear and the clock field set to
;19066  ;          111 the mod field set to nop. Issue a nop instruction with 00101 in
;19067  ;          the branch field. If there is a branch then issue error2.
;19068  ;       3) Set the instruction encode bits to 10111 with a MISC instruction
;19069  ;          then branch on them.
;19070  ;       4) Repeat step 3 with the instruction bits set to 01111.
;19071  ;       5) Repeat step 3 with the instruction bits set to 00011.
;19072  ;       6) Repeat step 3 with the instruction bits set to 00101.
;19073  ;       7) Repeat step 3 with the instruction bits set to 00110.
;19074  ;
;19075  ; Error:
;19076  ;
;19077  ;       Error1 - OP1 SIGN and EMOD failed asserted.
;19078  ;       Error2 - OP1 SIGN and EMOD failed unasserted.
;19079  ;       Error3 - EMOD failed asserted.
;19080  ;       Error4 - EMOD failed asserted.
;19081  ;       Error5 - EMOD failed asserted.
;19082  ;       Error6 - EMOD failed asserted.
;19083  ;       Error7 - EMOD failed asserted.
;19084  ;
;19085  ; Debug:
;19086  ;
;19087  ;       Error1: If this error occurs because OP1 SIGN failed then the BRANCH 1
;19088  ;               PAL should be checked for error. If the PAL isn't the cause of
;19089  ;               the error then the etch between the BRANCH 1 PAL and the SIGN
;19090  ;               CTL PAL should be checked for error. The UBCTL bits going into
;19091  ;               the BRANCH 1 PAL should be 00101. If the UBCTL bits are in
```

```
                ;19092  ;               error then the CS latch thet came from should be checked for
                ;19093  ;               error. If this error occured because EMOD failed then the
                ;19094  ;               BRANCH 0 PAL should be checked for error. If the PAL isn't the
                ;19095  ;               cause of error then the instruction encode bits should be
                ;19096  ;               checked to be 11000. If the instruction encode bits are in
                ;19097  ;               error then the etch between the PAL and the MUX should be
                ;19098  ;               checked for error.
                ;19099  ;       Error2: Same as error1 except that OP1 SIGN is unasserted and the
                ;19100  ;               instruction encode bits are 00000.
                ;19101  ;       Error3: Same as error 1 except that OP1 SIGN in unasserted and the
                ;19102  ;               instruction bits are 10111.
                ;19103  ;       Error4: Same as error 1 except that OP1 SIGN in unasserted and the
                ;19104  ;               instruction bits are 01111.
                ;19105  ;       Error5: Same as error 1 except thatOP1 SIGN in unasserted and the
                ;19106  ;               instruction bits are 00011.
                ;19107  ;       Error6: Same as error 1 except that OP1 SIGN in unasserted and the
                ;19108  ;               instruction bits are 00101.
                ;19109  ;       Error7: Same as error 1 except that OP1 SIGN in unasserted and the
                ;19110  ;               instruction bits are 10110.
                ;19111  ;
                ;19112  ;
                ;19113  ;--
                ;19114  T.30:
U 1C62, B65E,15 ;19115          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;19116                                          ; STATUS WORD
U 1C63, 3E80,15 ;19117          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19118                                          ; BIT 15 INDICATES START OF TEST TO
                ;19119                                          ; CONSOLE
U 1C64, 10E0,15 ;19120          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19121  WAIT.T30.0:
U 1C65, 09C6,54 ;19122          JMP [WAIT.T30.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19123                                          ; RESPOND
U 1C66, 887E,EC ;19124          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19125                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1C67, 3768,15 ;19126          MOV LS[T30.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1C68, BE12,15 ;19127          MOV WR[0] TO LS[T9]             ; references.
U 1C69, 8870,3C ;19128          JSR [L0]                        ;Get branch address
U 1C6A, BE14,15 ;19129          MOV WR[0] TO LS[T10]
U 1C6B, 8870,3C ;19130          JSR [L0]                        ;Get Address to set OP1 SIGN
U 1C6C, 3E16,15 ;19131          MOV WR[0] TO LS[T11]
U 1C6D, 376E,15 ;19132          MOV LS[#41F00] TO WR[0]         ;Set the instruction bits to EMOD
U 1C6E, 90F6,15 ;19133          MISC2 [TRAP.ACC] WR[0]
U 1C6F, B616,15 ;19134  T30.1:  MOV LS[T11] TO WR[0]            ;Get address to CLOCK OP1 SIGN
U 1C70, B776,95 ;19135          MOV LS[#8300] TO WR[1]          ;Get address to force to enable the
                ;19136                                          ; the BIT 15 onto the FPA BUS
U 1C71, 90F6,15 ;19137          MISC2 [TRAP.ACC] WR[0]          ;Force CLOCK OP1 SIGN
U 1C72, 10F6,95 ;19138          MISC2 [TRAP.ACC] WR[1]          ;Force the data
U 1C73, 3614,15 ;19139          MOV LS[T10] TO WR[0]            ;Get branch address
U 1C74, 90F6,15 ;19140          MISC2 [TRAP.ACC] WR[0]          ;Force branch
U 1C75, 10F8,15 ;19141          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1C76, 3A18,15 ;19142          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1C77, 3618,15 ;19143          MOV LS[T12] TO WR[0]            ;Set up received data
U 1C78, 3742,95 ;19144          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1C79, 0869,3C ;19145          JSR [CHECK.RESULT]              ;Check for error
U 1C7A, 09C6,F4 ;19146          JMP [T30.1]                     ;Loop on error
```

```
U 1C7B, FF82,15 ;19147          INC LS[ERROR.NUMBER]              ;Go on to next error
U 1C7C, B718,15 ;19148          MOV LS[#00040300] TO WR[0]        ;Set instruction bits to zero
U 1C7D, 90F6,15 ;19149          MISC2 [TRAP.ACC] WR[0]
U 1C7E, B616,15 ;19150  T30.2:  MOV LS[T11] TO WR[0]              ;Get CLOCK OP1 SIGN address
U 1C7F, B716,95 ;19151          MOV LS[#300] TO WR[1]             ;Get Address to enable tranceivers with
                ;19152                                            ; with bit 15 clear
U 1C80, 90F6,15 ;19153          MISC2 [TRAP.ACC] WR[0]            ;Force CLOCK OP1 SIGN
U 1C81, 10F6,95 ;19154          MISC2 [TRAP.ACC] WR[1]            ;Force enable of tranceivers onto the
                ;19155                                            ; FPA BUS
U 1C82, B614,95 ;19156          MOV LS[T10] TO WR[1]              ;Get branch adress to force
U 1C83, 10F6,95 ;19157          MISC2 [TRAP.ACC] WR[1]
U 1C84, 10F8,15 ;19158          MISC2 [READ.ACC.UPC]              ;Enable the NUA BUS onto the FPA BUS
U 1C85, 3A18,15 ;19159          MOV FPA TO LS[T12]                ;Enable the FPA BUS onto the Y BUS
U 1C86, 3618,15 ;19160          MOV LS[T12] TO WR[0]              ;Set up received data
U 1C87, B716,95 ;19161          MOV LS[#300] TO WR[1]             ;Set up expected data
U 1C88, 0869,3C ;19162          JSR [CHECK.RESULT]                ;Check for error
U 1C89, 09C7,E4 ;19163          JMP [T30.2]                       ;Loop on error
U 1C8A, FF82,15 ;19164          INC LS[ERROR.NUMBER]              ;Go onto next error
U 1C8B, B76A,15 ;19165          MOV LS[#45F00] TO WR[0]           ;Set instruction bits to not EMOD
U 1C8C, 90F6,15 ;19166          MISC2 [TRAP.ACC] WR[0]
U 1C8D, 3614,15 ;19167  T30.3:  MOV LS[T10] TO WR[0]              ;Get branch address
U 1C8E, 90F6,15 ;19168          MISC2 [TRAP.ACC] WR[0]
U 1C8F, 10F8,15 ;19169          MISC2 [READ.ACC.UPC]              ;Enable the NUA BUS onto the FPA BUS
U 1C90, 3A18,15 ;19170          MOV FPA TO LS[T12]                ;Enable the FPA BUS onto the Y BUS
U 1C91, 3618,15 ;19171          MOV LS[T12] TO WR[0]              ;Set up received data
U 1C92, B716,95 ;19172          MOV LS[#300] TO WR[1]             ;Set up expected data
U 1C93, 0869,3C ;19173          JSR [CHECK.RESULT]                ;Check for error
U 1C94, 09C8,D4 ;19174          JMP [T30.3]                       ;Loop on error
U 1C95, FF82,15 ;19175          INC LS[ERROR.NUMBER]              ;Go onto next error
U 1C96, B76C,15 ;19176          MOV LS[#43F00] TO WR[0]           ;Set instruction bits to not EMOD
U 1C97, 90F6,15 ;19177          MISC2 [TRAP.ACC] WR[0]
U 1C98, 3614,15 ;19178  T30.4:  MOV LS[T10] TO WR[0]              ;Get branch address
U 1C99, 90F6,15 ;19179          MISC2 [TRAP.ACC] WR[0]
U 1C9A, 10F8,15 ;19180          MISC2 [READ.ACC.UPC]              ;Enable the NUA BUS onto the FPA BUS
U 1C9B, 3A18,15 ;19181          MOV FPA TO LS[T12]                ;Enable the FPA BUS onto the Y BUS
U 1C9C, 3618,15 ;19182          MOV LS[T12] TO WR[0]              ;Set up received data
U 1C9D, B716,95 ;19183          MOV LS[#300] TO WR[1]             ;Set up expected data
U 1C9E, 0869,3C ;19184          JSR [CHECK.RESULT]                ;Check for error
U 1C9F, 89C9,84 ;19185          JMP [T30.4]                       ;Loop on error
U 1CA0, FF82,15 ;19186          INC LS[ERROR.NUMBER]              ;Go onto next error
U 1CA1, B75A,15 ;19187          MOV LS[#40F00] TO WR[0]           ;Set instruction bits to not EMOD
U 1CA2, 90F6,15 ;19188          MISC2 [TRAP.ACC] WR[0]
U 1CA3, 3614,15 ;19189  T30.5:  MOV LS[T10] TO WR[0]              ;Get branch address
U 1CA4, 90F6,15 ;19190          MISC2 [TRAP.ACC] WR[0]
U 1CA5, 10F8,15 ;19191          MISC2 [READ.ACC.UPC]              ;Enable the NUA BUS onto the FPA BUS
U 1CA6, 3A18,15 ;19192          MOV FPA TO LS[T12]                ;Enable the FPA BUS onto the Y BUS
U 1CA7, 3618,15 ;19193          MOV LS[T12] TO WR[0]              ;Set up received data
U 1CA8, B716,95 ;19194          MOV LS[#300] TO WR[1]             ;Set up expected data
U 1CA9, 0869,3C ;19195          JSR [CHECK.RESULT]                ;Check for error
U 1CAA, 09CA,34 ;19196          JMP [T30.5]                       ;Loop on error
U 1CAB, FF82,15 ;19197          INC LS[ERROR.NUMBER]              ;Go onto next error
U 1CAC, 3764,15 ;19198          MOV LS[#41700] TO WR[0]           ;Set instruction bits to not EMOD
U 1CAD, 90F6,15 ;19199          MISC2 [TRAP.ACC] WR[0]
U 1CAE, 3614,15 ;19200  T30.6:  MOV LS[T10] TO WR[0]              ;Get branch address
U 1CAF, 90F6,15 ;19201          MISC2 [TRAP.ACC] WR[0]
```

```
U 1CB0, 10F8,15 ;19202          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1CB1, 3A18,15 ;19203          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1CB2, 3618,15 ;19204          MOV LS[T12] TO WR[0]            ;Set up received data
U 1CB3, B716,95 ;19205          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1CB4, 0869,3C ;19206          JSR [CHECK.RESULT]              ;Check for error
U 1CB5, 89CA,E4 ;19207          JMP [T30.6]                     ;Loop on error
U 1CB6, FF82,15 ;19208          INC LS[ERROR.NUMBER]            ;Go onto next error
U 1CB7, B766,15 ;19209          MOV LS[#41B00] TO WR[0]         ;Set instruction bits to not EMOD
U 1CB8, 90F6,15 ;19210          MISC2 [TRAP.ACC] WR[0]
U 1CB9, 3614,15 ;19211  T30.7:  MOV LS[T10] TO WR[0]            ;Get branch address
U 1CBA, 90F6,15 ;19212          MISC2 [TRAP.ACC] WR[0]
U 1CBB, 10F8,15 ;19213          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1CBC, 3A18,15 ;19214          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1CBD, 3618,15 ;19215          MOV LS[T12] TO WR[0]            ;Set up received data
U 1CBE, B716,95 ;19216          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1CBF, 0869,3C ;19217          JSR [CHECK.RESULT]              ;Check for error
U 1CC0, 89CB,94 ;19218          JMP [T30.7]                     ;Loop on error
                ;19219  T30.END:
```

```
;19220  .page
;19221  .TOC    "TEST 31 FRAC55 F3 and SINGLE BRANCH TEST(M8389 FPA MODULE)"
;19222  ;++
;19223  ;
;19224  ;
;19225  ; Test Description:
;19226  ;
;19227  ;       The purpose of this test is to check the branch conditions, FRAC55 F3
;19228  ;       and SINGLE. The data path will be as follows: The branch conditions
;19229  ;       will be set then a nop instruction will be issued with the branch field
;19230  ;       set to 00110 and an error statement will be issued if there is an error
;19231  ;       The same procedure will be repeated with the branch conditions unas-
;19232  ;       serted. FRAC55 F3 will be asserted if the sign bit of the 2901 on the
;19233  ;       left hand side of page L is set. This will be set by moving the
;19234  ;       contents of a WR with the MSB set to itself. Single can be set by
;19235  ;       loading the size bits with 00.
;19236  ;
;19237  ; Assumptions:
;19238  ;
;19239  ;       It is assumed that the branch control logic has been tested previous to
;19240  ;       this test and works.
;19241  ;
;19242  ; Test Steps:
;19243  ;
;19244  ;       1) Issue a FORCE instruction with bit 18 set to set the instruction
;19245  ;          encode bits and the size bits. Load a WR with a 1 in the MSB of
;19246  ;          one and 0's in the other. Move the contents of the WR to itself to
;19247  ;          set the sign bit, with 00110 in the branch field. If there is no
;19248  ;          branch or an incorrect branch then issue error1.
;19249  ;       2) Issue a FORCE instruction with bit 18 set to set the instruction
;19250  ;          encode bits and the size bits. Load two WR's with 0's and OR them
;19251  ;          together with 00110 in the branch field. If there is a branch then
;19252  ;          issue error2.
;19253  ;
;19254  ; Error:
;19255  ;
;19256  ;       Error1 - FRAC55 F3 and SINGLE failed asserted.
;19257  ;       Error2 - FRAC55 F3 and SINGLE failed unasserted.
;19258  ;
;19259  ; Debug:
;19260  ;
;19261  ;       Error1: If this error occurs it's likely that it's either the UBCTL
;19262  ;               bits or the FRAC55 F3 branch condition. If the branch condi-
;19263  ;               tion is in error then the BRANCH 1 PAL should be checked for
;19264  ;               error. If the PAL isn't the cause of the error then the STATUS
;19265  ;               REG 1 LATCH should be checked for error. If the latch isn't the
;19266  ;               cause of the error then the sign output of the 2901 should be
;19267  ;               checked for error. If the UBCTL bits are not 00110 then the
;19268  ;               etch between the MUX and the PAL should be checked for error.
;19269  ;       Error2: Same as error1 except that the size bits are 10 and FRAC55 F3
;19270  ;               isn't asserted.
;19271  ;
;19272  ;
;19273  ;--
;19274  T.31:
```

```
U 1CC1, B65E,15 ;19275          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;19276                                          ; STATUS WORD
U 1CC2, 3E80,15 ;19277          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19278                                          ; BIT 15 INDICATES START OF TEST TO
                ;19279                                          ; CONSOLE
U 1CC3, 10E0,15 ;19280          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19281  WAIT.T31.0:
U 1CC4, 89CC,44 ;19282          JMP [WAIT.T31.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19283                                          ; RESPOND
U 1CC5, 887E,EC ;19284          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19285                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1CC6, 3770,15 ;19286          MOV LS[T31.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1CC7, BE12,15 ;19287          MOV WR[0] TO LS[T9]              ; references.
U 1CC8, B72E,15 ;19288          MOV LS[#2A7] TO WR[0]           ;Get address of Load routine
U 1CC9, BE14,15 ;19289          MOV WR[0] TO LS[T10]
U 1CCA, 8870,3C ;19290          JSR [L0]                        ;Get branch address
U 1CCB, 3E1A,15 ;19291          MOV WR[0] TO LS[T13]
U 1CCC, 8870,3C ;19292          JSR [L0]                        ;Get address of CLR FWR[FT0]
U 1CCD, 3E16,15 ;19293          MOV WR[0] TO LS[T11]
U 1CCE, 8870,3C ;19294          JSR [L0]                        ;Get Address of MOV FT0 TO FT0
U 1CCF, 3E1C,15 ;19295          MOV WR[0] TO LS[T14]
U 1CD0, B718,15 ;19296          MOV LS[#00040300] TO WR[0]      ;Set size bits to single
U 1CD1, 90F6,15 ;19297          MISC2 [TRAP.ACC] WR[0]
U 1CD2, E5A2,15 ;19298  T31.1:  CLR LS[DATA.1]                  ;Put zeros in tne data to be loaded
U 1CD3, 8877,DC ;19299          JSR [LOAD.DATA]                 ;Load the data into FWR[FT0]
U 1CD4, B61C,15 ;19300          MOV LS[T14] TO WR[0]            ;Force move to set up F55 F3
U 1CD5, 361A,95 ;19301          MOV LS[T13] TO WR[1]            ;Force branch
U 1CD6, 90F6,15 ;19302          MISC2 [TRAP.ACC] WR[0]
U 1CD7, 10F6,95 ;19303          MISC2 [TRAP.ACC] WR[1]
U 1CD8, 10F8,15 ;19304          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1CD9, 3A18,15 ;19305          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1CDA, 3618,15 ;19306          MOV LS[T12] TO WR[0]            ;Set up received data
U 1CDB, 3742,95 ;19307          MOV LS[#303] TO WR[1]           ;Set expected data
U 1CDC, 0869,3C ;19308          JSR [CHECK.RESULT]              ;Check for error
U 1CDD, 09CD,24 ;19309          JMP [T31.1]                     ;Loop on error
U 1CDE, FF82,15 ;19310          INC LS[ERROR.NUMBER]            ;Go on to next error
U 1CDF, B748,15 ;19311          MOV LS[#70300] TO WR[0]         ;Set size bit to other than single
U 1CE0, 90F6,15 ;19312          MISC2 [TRAP.ACC] WR[0]
U 1CE1, B616,15 ;19313  T31.2:  MOV LS[T11] TO WR[0]            ;Get address to CLR FT0
U 1CE2, 361A,95 ;19314          MOV LS[T13] TO WR[1]            ;Get branch address
U 1CE3, 90F6,15 ;19315          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 1CE4, 10F6,95 ;19316          MISC2 [TRAP.ACC] WR[1]          ;Branch on F55 F3 and single
U 1CE5, 10F8,15 ;19317          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1CE6, 3A18,15 ;19318          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1CE7, 3618,15 ;19319          MOV LS[T12] TO WR[0]            ;Set up received data
U 1CE8, B716,95 ;19320          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1CE9, 0869,3C ;19321          JSR [CHECK.RESULT]              ;Check for error
U 1CEA, 09CE,14 ;19322          JMP [T31.2]                     ;Loop on error
U 1CEB, B718,15 ;19323          MOV LS[#00040300] TO WR[0]      ;Set size bits back to single
U 1CEC, 90F6,15 ;19324          MISC2 [TRAP.ACC] WR[0]
                ;19325  T31.END:
```

```
;19326  .page
;19327  .TOC    "TEST 32 OP2 SIGN AND ADD + SUB BRANCH TEST(M8389 FPA MODULE)"
;19328  ;++
;19329  ;
;19330  ;
;19331  ; Test Description:
;19332  ;
;19333  ;       The purpose of this test is to check the OP2 SIGN and ADD + SUB branch
;19334  ;       conditions. The data path is as follows: the branch conditions are set
;19335  ;       with the branch field set to 00111. Then an error statement is issued
;19336  ;       if there is an error. This procedure is repeated for the bracnch condi-
;19337  ;       tions unasserted. Both OP2 SIGN  and ADD + SUB have been tested before.
;19338  ;       OP2 SIGN can be asserted by enabling 1 into OP2 SIGN by setting the A
;19339  ;       address lines to the appropriate value and the clock field to the right
;19340  ;       value. ADD + SUB can be asserted by loading the instruction encode with
;19341  ;       00000.
;19342  ;
;19343  ; Assumptions:
;19344  ;
;19345  ;       It is assumed that the branch control logic has been tested previously
;19346  ;       and is working.
;19347  ;
;19348  ; Test Steps:
;19349  ;
;19350  ;       1) Issue a FORCE instruction with bit 18 set to set the instruction
;19351  ;          encode bits and the size bits. Issue an FPA instruction to set the
;19352  ;          OP2 SIGN bit with the branch field set to 00110. If there is no
;19353  ;          branch or an incorrect branch then issue error 1.
;19354  ;       2) Issue a FORCE instruction with bit 18 set to set the instruction
;19355  ;          encode bits and the size bits. Issue a nop with the branch field
;19356  ;          set to 00110. If there is any branch then issue error 2.
;19357  ;       3) Set the instruction encode bits to 00010 and branch on OP2 SIGN
;19358  ;          and ADD+SUB.
;19359  ;       4) Repeat step 3 with the instruction bits set to 00001.
;19360  ;       5) Repeat step 3 with the instruction bits set to 00011.
;19361  ;       6) Repeat step 3 with the instruction bits set to 01000.
;19362  ;       7) Repeat step 3 with the instruction bits set to 10000.
;19363  ;
;19364  ; Error:
;19365  ;
;19366  ;       Error1 - OP2 SIGN and ADD + SUB failed asserted.
;19367  ;       Error2 - OP2 SIGN and ADD + SUB failed unasserted.
;19368  ;       Error3 - ADD + SUB failed asserted.
;19369  ;       Error4 - ADD + SUB failed asserted.
;19370  ;       Error5 - ADD + SUB failed asserted.
;19371  ;       Error6 - ADD + SUB failed unasserted.
;19372  ;       Error7 - ADD + SUB failed unasserted.
;19373  ;
;19374  ; Debug:
;19375  ;
;19376  ;       Error1: If this error occurs it's likely to be an error in the BRANCH 1
;19377  ;               PAL or the UBCTL  bits because everything else has been tested
;19378  ;               before. If OP2 SIGN is in error the BRANCH 1 PAL should be
;19379  ;               checked for error. If the UBCTL bits are in error then the
;19380  ;               BRANCH 1 PAL and the BRANCH 0 PAL should be checked for error.
```

```
                ;19381  ;               If the PAL's are not the source of error then the UPF latch
                ;19382  ;               should be checked for error.
                ;19383  ;       Error2: Same as error1 except that OP2 SIGN and ADD + SUB are both
                ;19384  ;               unasserted.
                ;19385  ;       Error3: Same as error 1 except OP2 SIGN is unasserted.
                ;19386  ;       Error4: Same as error 1 except OP2 SIGN is unasserted.
                ;19387  ;       Error5: Same as error 1 except OP2 SIGN is unasserted.
                ;19388  ;       Error6: Same as error 1 except ADD + SUB and OP2 SIGN is unasserted.
                ;19389  ;       Error7: Same as error 1 except ADD + SUB and OP2 SIGN is unasserted.
                ;19390  ;
                ;19391  ;
                ;19392  ;--
                ;19393  T.32:
U 1CED, B65E,15 ;19394          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;19395                                          ; STATUS WORD
U 1CEE, 3E80,15 ;19396          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19397                                          ; BIT 15 INDICATES START OF TEST TO
                ;19398                                          ; CONSOLE
U 1CEF, 10E0,15 ;19399          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19400  WAIT.T32.0:
U 1CF0, 09CF,04 ;19401          JMP [WAIT.T32.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19402                                          ; RESPOND
U 1CF1, 887E,EC ;19403          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19404                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1CF2, B772,15 ;19405          MOV LS[T32.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1CF3, BE12,15 ;19406          MOV WR[0] TO LS[T9]              ; references.
U 1CF4, 8870,3C ;19407          JSR [L0]                        ;Get branch address
U 1CF5, BE14,15 ;19408          MOV WR[0] TO LS[T10]
U 1CF6, 8870,3C ;19409          JSR [L0]                        ;Get Address to set OP2 SIGN
U 1CF7, 3E16,15 ;19410          MOV WR[0] TO LS[T11]
U 1CF8, B718,15 ;19411          MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to ADD + SUB
U 1CF9, 90F6,15 ;19412          MISC2 [TRAP.ACC] WR[0]
U 1CFA, B616,15 ;19413  T32.1:  MOV LS[T11] TO WR[0]            ;Get address to CLOCK OP2 SIGN
U 1CFB, B776,95 ;19414          MOV LS[#8300] TO WR[1]          ;Get address to force to enable the
                ;19415                                          ; the tranceivers onto the FPA BUS
U 1CFC, 90F6,15 ;19416          MISC2 [TRAP.ACC] WR[0]          ;Force CLOCK OP2 SIGN
U 1CFD, 10F6,95 ;19417          MISC2 [TRAP.ACC] WR[1]          ;Enable BIT15 onto the FPA BUS
U 1CFE, B614,95 ;19418          MOV LS[T10] TO WR[1]            ;Get branch adress to force
U 1CFF, 10F6,95 ;19419          MISC2 [TRAP.ACC] WR[1]
U 1D00, 10F8,15 ;19420          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D01, 3A18,15 ;19421          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D02, 3618,15 ;19422          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D03, 3742,95 ;19423          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1D04, 0869,3C ;19424          JSR [CHECK.RESULT]              ;Check for error
U 1D05, 09CF,A4 ;19425          JMP [T32.1]                     ;Loop on error
U 1D06, FF82,15 ;19426          INC LS[ERROR.NUMBER]            ;Go on to next error
U 1D07, 3754,15 ;19427          MOV LS[#44300] TO WR[0]         ;Set instruction bits to zero
U 1D08, 90F6,15 ;19428          MISC2 [TRAP.ACC] WR[0]
U 1D09, B616,15 ;19429  T32.2:  MOV LS[T11] TO WR[0]            ;Get address to CLOCK OP2 SIGN
U 1D0A, B716,95 ;19430          MOV LS[#300] TO WR[1]           ;Get address to force to enable the
                ;19431                                          ; the tranceivers onto the FPA BUS
U 1D0B, 90F6,15 ;19432          MISC2 [TRAP.ACC] WR[0]          ;Force CLOCK OP2 SIGN
U 1D0C, 10F6,95 ;19433          MISC2 [TRAP.ACC] WR[1]          ;Enable BIT15 onto the FPA BUS
U 1D0D, B614,95 ;19434          MOV LS[T10] TO WR[1]            ;Get branch adress to force
U 1D0E, 10F6,95 ;19435          MISC2 [TRAP.ACC] WR[1]
```

```
U 1D0F, 10F8,15 ;19436          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D10, 3A18,15 ;19437          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D11, 3618,15 ;19438          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D12, B716,95 ;19439          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1D13, 0869,3C ;19440          JSR [CHECK.RESULT]              ;Check for error
U 1D14, 89D0,94 ;19441          JMP [T32.2]                     ;Loop on error
U 1D15, FF82,15 ;19442          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1D16, 3758,15 ;19443          MOV LS[#40B00] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1D17, 90F6,15 ;19444          MISC2 [TRAP.ACC] WR[0]
U 1D18, 3614,15 ;19445  T32.3:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1D19, 90F6,15 ;19446          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1D1A, 10F8,15 ;19447          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D1B, 3A18,15 ;19448          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D1C, 3618,15 ;19449          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D1D, B73E,95 ;19450          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1D1E, 0869,3C ;19451          JSR [CHECK.RESULT]              ;Report error if there is one
U 1D1F, 89D1,84 ;19452          JMP [T32.3]                     ;Loop on error
U 1D20, FF82,15 ;19453          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1D21, B756,15 ;19454          MOV LS[#40700] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1D22, 90F6,15 ;19455          MISC2 [TRAP.ACC] WR[0]
U 1D23, 3614,15 ;19456  T32.4:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1D24, 90F6,15 ;19457          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1D25, 10F8,15 ;19458          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D26, 3A18,15 ;19459          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D27, 3618,15 ;19460          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D28, B73E,95 ;19461          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1D29, 0869,3C ;19462          JSR [CHECK.RESULT]              ;Report error if there is one
U 1D2A, 09D2,34 ;19463          JMP [T32.4]                     ;Loop on error
U 1D2B, FF82,15 ;19464          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1D2C, B75A,15 ;19465          MOV LS[#40F00] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1D2D, 90F6,15 ;19466          MISC2 [TRAP.ACC] WR[0]
U 1D2E, 3614,15 ;19467  T32.5:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1D2F, 90F6,15 ;19468          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1D30, 10F8,15 ;19469          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D31, 3A18,15 ;19470          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D32, 3618,15 ;19471          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D33, B73E,95 ;19472          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1D34, 0869,3C ;19473          JSR [CHECK.RESULT]              ;Report error if there is one
U 1D35, 89D2,E4 ;19474          JMP [T32.5]                     ;Loop on error
U 1D36, FF82,15 ;19475          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1D37, 3762,15 ;19476          MOV LS[#42300] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1D38, 90F6,15 ;19477          MISC2 [TRAP.ACC] WR[0]
U 1D39, 3614,15 ;19478  T32.6:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1D3A, 90F6,15 ;19479          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
U 1D3B, 10F8,15 ;19480          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D3C, 3A18,15 ;19481          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D3D, 3618,15 ;19482          MOV LS[T12] TO WR[0]            ;Set up received data
U 1D3E, B716,95 ;19483          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1D3F, 0869,3C ;19484          JSR [CHECK.RESULT]              ;Report error if there is one
U 1D40, 89D3,94 ;19485          JMP [T32.6]                     ;loop on error
U 1D41, FF82,15 ;19486          INC LS[ERROR.NUMBER]            ;Go on to the next error
U 1D42, B75C,15 ;19487          MOV LS[#41300] TO WR[0]         ;Set instruction bits to ADD+SUB
U 1D43, 90F6,15 ;19488          MISC2 [TRAP.ACC] WR[0]
U 1D44, 3614,15 ;19489  T32.7:  MOV LS[T10] TO WR[0]            ;Restore address for loop on error
U 1D45, 90F6,15 ;19490          MISC2 [TRAP.ACC] WR[0]          ;Branch on CPU DATA AVAIL and ADD+SUB
```

```
U 1D46, 10F8,15 ;19491          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1D47, 3A18,15 ;19492          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1D48, 3618,15 ;19493          MOV LS[T12] TO WR[0]          ;Set up received data
U 1D49, B716,95 ;19494          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1D4A, 0869,3C ;19495          JSR [CHECK.RESULT]            ;Report error if there is one
U 1D4B, 89D4,44 ;19496          JMP [T32.7]                   ;Loop on error
                ;19497  T32.END:
```

```
                ;19498  .page
                ;19499  .TOC    "TEST 33 EXP COUT AND EXP15 F3 BRANCH TEST(M8389 FPA MODULE)"
                ;19500  ;++
                ;19501  ;
                ;19502  ;
                ;19503  ; Test Description:
                ;19504  ;
                ;19505  ;       The purpose of this test is to check the EXP COUT and EXP15 F3 branch
                ;19506  ;       conditions. The data path will be as follows: the branch conditions
                ;19507  ;       will be asserted and branched on by setting the branch field to 01000.
                ;19508  ;       If there is an error then an error statement is issued. Otherwise the
                ;19509  ;       branch conditions are branched on when they are unasserted. EXP COUT
                ;19510  ;       has been tested previously and can be asserted by asserting the carry
                ;19511  ;       out of the exponent data path. EXP15 F3 can be asserted if the overflow
                ;19512  ;       bit of the most significant 2901 in the exponent data path is asserted.
                ;19513  ;
                ;19514  ; Assumptions:
                ;19515  ;
                ;19516  ;       It is assumed that the branch control logic has been tested previously
                ;19517  ;       and is working.
                ;19518  ;
                ;19519  ; Test Steps:
                ;19520  ;
                ;19521  ;       1) Load one WR of the exponent data path with zeros and the other
                ;19522  ;          WR with one. Subtract one from zero with 01000 in the branch field.
                ;19523  ;          If there is no branch or a branch to the wrong place then issue
                ;19524  ;          error1.
                ;19525  ;       2) Load two WR of the exponent data path with a one in the MSB and
                ;19526  ;          issue an add instruction with the branch field set to 01000. If
                ;19527  ;          there is no branch or a branch to the wrong place then issue error
                ;19528  ;          2.
                ;19529  ;
                ;19530  ; Error:
                ;19531  ;
                ;19532  ;       Error1 - Branch failed when EXP15 F3 asserted and EXP COUT unasserted.
                ;19533  ;       Error2 - Branch failed when EXP15 F3 unasserted and EXP COUT asserted.
                ;19534  ;
                ;19535  ; Debug:
                ;19536  ;
                ;19537  ;       Error1: If this error occurs it's likely that the cause of the error is
                ;19538  ;               with either the BRANCH 3 PAL or the BRANCH 1 PAL or the UBCTL
                ;19539  ;               bits. If the BRANCH 1 PAL isn't in error then the UBCTL bits
                ;19540  ;               should be checked to be 01000. If the UCBTL bits are in error
                ;19541  ;               the latch they came from should be checked for error. If the
                ;19542  ;               BRANCH 3 PAL isn't the source of the error then the UBCTL bits
                ;19543  ;               should be checked for error as before and EXP15 F3 should be
                ;19544  ;               checked to be asserted. If EXP15 F3 is in error then the
                ;19545  ;               overflow output of the 2901 should be checked for error.
                ;19546  ;       Error2: Same as error1 except that EXP COUT is asserted and EXP15 F3 is
                ;19547  ;               unasserted.
                ;19548  ;
                ;19549  ;
                ;19550  ;--
                ;19551  T.33:
U 1D4C, B65E,15 ;19552          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
```

```
                ;19553                                  ; STATUS WORD
U 1D4D, 3E80,15 ;19554          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19555                                  ; BIT 15 INDICATES START OF TEST TO
                ;19556                                  ; CONSOLE
U 1D4E, 10E0,15 ;19557          MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE
                ;19558  WAIT.T33.0:
U 1D4F, 09D4,F4 ;19559          JMP [WAIT.T33.0]        ;LOOP HERE WAITING FOR CONSOLE TO
                ;19560                                  ; RESPOND
U 1D50, 887E,EC ;19561          JSR [SETUP.10]          ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19562                                  ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1D51, B774,15 ;19563          MOV LS[T33.POINTER] TO WR[0] ;Initialize pointer for main memory
U 1D52, BE12,15 ;19564          MOV WR[0] TO LS[T9]     ; references.
U 1D53, B72E,15 ;19565          MOV LS[#2A7] TO WR[0]   ;Get address of the LOAD
U 1D54, BE14,15 ;19566          MOV WR[0] TO LS[T10]
U 1D55, 8870,3C ;19567          JSR [LO]                ;Get address of the shift
U 1D56, 3E16,15 ;19568          MOV WR[0] TO LS[T11]
U 1D57, 8870,3C ;19569          JSR [LO]                ;Get address of MOV ET0 TO ET2
U 1D58, 3E1C,15 ;19570          MOV WR[0] TO LS[T14]
U 1D59, 8870,3C ;19571          JSR [LO]                ;Get address of ADD ET2 TO ET0
U 1D5A, BE18,15 ;19572          MOV WR[0] TO LS[T12]
U 1D5B, 8870,3C ;19573          JSR [LO]                ;Get address of BRANCH
U 1D5C, 3E1A,15 ;19574          MOV WR[0] TO LS[T13]
U 1D5D, E5A2,15 ;19575  T33.1:  CLR LS[DATA.1]          ;Set up data to be passed
U 1D5E, 8877,DC ;19576          JSR [LOAD.DATA]         ;Load the data
U 1D5F, 0876,2C ;19577          JSR [SHIFT.LEFT.8]      ;Shift the data into the upper bits
                ;19578                                  ; of the exponent
U 1D60, 0878,AC ;19579          JSR [MOV.DATA]          ;Move the data from ET0 TO ET2
U 1D61, 365C,15 ;19580          MOV LS[BIT14] TO WR[0]  ;Set up data to be loaded
U 1D62, 3EA2,15 ;19581          MOV WR[0] TO LS[DATA.1]
U 1D63, 8877,DC ;19582          JSR [LOAD.DATA]         ;Load the data
U 1D64, 0876,2C ;19583          JSR [SHIFT.LEFT.8]      ;Shift the data into the upper bits
                ;19584                                  ; of the exponent
U 1D65, 3618,15 ;19585          MOV LS[T12] TO WR[0]    ;Set up for ADD
U 1D66, 361A,95 ;19586          MOV LS[T13] TO WR[1]    ;Set up for branch
U 1D67, 90F6,15 ;19587          MISC2 [TRAP.ACC] WR[0]  ;Force ADD
U 1D68, 10F6,95 ;19588          MISC2 [TRAP.ACC] WR[1]  ;Force the branch
U 1D69, 10F8,15 ;19589          MISC2 [READ.ACC.UPC]    ;Enable the NUA BUS onto the FPA BUS
U 1D6A, 3A1E,15 ;19590          MOV FPA TO LS[T15]      ;Enable the FPA BUS onto the Y BUS
U 1D6B, 361E,15 ;19591          MOV LS[T15] TO WR[0]    ;Set up received data
U 1D6C, B73E,95 ;19592          MOV LS[#301] TO WR[1]   ;Set up expected data
U 1D6D, 0869,3C ;19593          JSR [CHECK.RESULT]      ;Check for error
U 1D6E, 09D5,D4 ;19594          JMP [T33.1]             ;Loop on error
  1D6F, FF82,15 ;19595          INC LS[ERROR.NUMBER]    ;Go on to next error
U 1D70, 365C,15 ;19596  T33.2:  MOV LS[BIT14] TO WR[0]  ;Set up data to pass
U 1D71, 3EA2,15 ;19597          MOV WR[0] TO LS[DATA.1]
U 1D72, 8877,DC ;19598          JSR [LOAD.DATA]         ;Load the data
U 1D73, 0876,2C ;19599          JSR [SHIFT.LEFT.8]      ;Shift the data in the upper bits of
                ;19600                                  ; the exponent data path
U 1D74, 0878,AC ;19601          JSR [MOV.DATA]          ;MOV ET0 TO ET2
U 1D75, 3618,15 ;19602          MOV LS[T12] TO WR[0]    ;Set up for ADD ET0 TO ET2
U 1D76, 361A,95 ;19603          MOV LS[T13] TO WR[1]    ;Set up for branch
U 1D77, 90F6,15 ;19604          MISC2 [TRAP.ACC] WR[0]  ;Force ADD
U 1D78, 10F6,95 ;19605          MISC2 [TRAP.ACC] WR[1]  ;Force branch
U 1D79, 10F8,15 ;19606          MISC2 [READ.ACC.UPC]    ;Enable the NUA BUS onto the FPA BUS
U 1D7A, 3A1E,15 ;19607          MOV FPA TO LS[T15]      ;Enable the FPA BUS onto the Y BUS
```

```
U 1D7B, 361E,15 ;19608          MOV LS[T15] TO WR[0]          ;Set up received data
U 1D7C, B740,95 ;19609          MOV LS[#302] TO WR[1]         ;Set up expected data
U 1D7D, 0869,3C ;19610          JSR [CHECK.RESULT]            ;Check for error
U 1D7E, 09D7,04 ;19611          JMP [T33.2]                   ;Loop on error
                ;19612  T33.END:
```

```
;19613  .page
;19614  .TOC    "TEST 34 SIGN OUT AND OP2=0 BRANCH TEST(M8389 FPA MODULE)"
;19615  ;++
;19616  ;
;19617  ;
;19618  ; Test Description:
;19619  ;
;19620  ;       The purpose of this test is to check the SIGN OUT and OP2=0 branch con-
;19621  ;       ditions. The data path will be as follows: The branch conditions will
;19622  ;       set and the branch field will be set to 01001. If there is no branch or
;19623  ;       a branch to the wrong place then an error statement will be issued.
;19624  ;       This procedure will be repeated for the branch conditions when they're
;19625  ;       unasserted. SIGN OUT has been tested previously and can be asserted by
;19626  ;       enabling 1 into one of the OP SIGN bits and then enabling the OP SIGN
;19627  ;       bit into SIGN OUT. OP2=0 can be set by a variety of conditions which
;19628  ;       will be tested in this test.
;19629  ;
;19630  ; Assumptions:
;19631  ;
;19632  ;       It is assumed that the branch control logic has been tested previously
;19633  ;       and it works.
;19634  ;
;19635  ; Test Steps:
;19636  ;
;19637  ;       1) Issue a MISC instruction to load SIGN OUT with 1. This can be done
;19638  ;          by issuing a nop with the mod field set to nop, the clock field set
;19639  ;          to 100, and the A Address lines set to 0101.
;19640  ;       2) Issue a MISC instruction to asserted OP2=0. This can be done by
;19641  ;          loading zeros into the exponent data path then on the next clock
;19642  ;          cycle ORing the zeros with zeros with the clock field set to 000
;19643  ;          and the mod field set to nop and the branch field set to 01001. If
;19644  ;          there is no branch or an incorrect branch then issue error 1.
;19645  ;       3) Issue a nop with the branch bits set to 01001. This will check to
;19646  ;          that the OP2=0 stays asserted.
;19647  ;       4) Load the ET0 with ones then move ET0 to EQ then branch. This will
;19648  ;          check that the OP2=0 stays asserted when EXP0 is unasserted.
;19649  ;       5) Clock SIGN OUT with zero. Then move the contents of EQ to ET0 and
;19650  ;          CLOCK OP2=0 and branch. This should deassert OP2=0 and SIGN OUT.
;19651  ;       6) Issue a branch instruction with the branch bits set to 01001. This
;19652  ;          will check that OP2=0 stays unasserted when ENB CLK is unasserted.
;19653  ;
;19654  ; Error:
;19655  ;
;19656  ;       Error1 - Branch failed when OP2=0 and SIGN OUT were asserted and OP2=0
;19657  ;                was unasserted on the previous cycle.
;19658  ;       Error2 - Branch failed when OP2=0 and SIGN OUT were asserted and ENB
;19659  ;                CLK was unasserted.
;19660  ;       Error3 - Branch failed when OP2=0 and SIGN OUT were asserted and the
;19661  ;                exponent data path contained ones in the lower bits.
;19662  ;       Error4 - Branch failed when OP2=0 and SIGN OUT were unasserted and ENB
;19663  ;                CLK was asserted.
;19664  ;       Error5 - Branch failed when OP2=0 and SIGN OUT were unasserted and ENB
;19665  ;                CLK wasn't asserted.
;19666  ;
;19667  ; Debug:
```

```
                ;19668  ;
                ;19669  ;       Error1: If this error occurs because SIGN OUT failed then the BRANCH 1
                ;19670  ;               PAL should be checked for error. If the PAL isn't the cause of
                ;19671  ;               the error then the UBCTL bits should be checked to be 01001. If
                ;19672  ;               the UBCTL bits are in error then the latch they come from
                ;19673  ;               should be checked for error. Since the SIGN OUT bit has been
                ;19674  ;               checked before it not likely that SIGN OUT is the cause of
                ;19675  ;               error. If OP2=0 failed then the BRANCH 3 PAL should be checked
                ;19676  ;               for error. If the PAL isn't the cause of the error then ENB
                ;19677  ;               CLK, and EXP=0 should be checked to be asserted and EXP15 F3
                ;19678  ;               should be checked to be unasserted. If ENB CLK is in error then
                ;19679  ;               the EXT FUNC CTL PAL should be checked for error. If EXP=0 is
                ;19680  ;               in error then the 2901 in the exponent data path that it came
                ;19681  ;               from should be checked for error. If EXP15 F3 is in error then
                ;19682  ;               the most significant 2901 in the should be checked for error.
                ;19683  ;       Error2: Same as error1 except that OP2=0 was asserted on the previous
                ;19684  ;               cycle and ENB CLK is unasserted.
                ;19685  ;       Error3: Same as error1 except that OP2=0 was asserted on the previous
                ;19686  ;               cycle and EXP=0 is unasserted.
                ;19687  ;       Error4: Same as error1 except that SIGN OUT and OP2=0 are unasserted
                ;19688  ;               and ENB CLK was asserted.
                ;19689  ;       Error5: Same as error1 except that SIGN OUT is unasserted, OP2=0 is
                ;19690  ;               unasserted and ENB CLK is unasserted.
                ;19691  ;
                ;19692  ;
                ;19693  ;       T10 - LOAD
                ;19694  ;       T14 - MOV ETO TO EQ
                ;19695  ;       T15 - MOV EQ TO ETO AND CLOCK OP2=0
                ;19696  ;       T8 - CLOCK SIGN OUT WITH ONE
                ;19697  ;       T11 - CLOCK SIGN OUT WITH ZERO
                ;19698  ;       T13 - BRANCH ON SIGN OUT AND OP2=0
                ;19699  ;
                ;19700  ;--
                ;19701  T.34:
U 1D7F, B65E,15 ;19702          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;19703                                          ; STATUS WORD
U 1D80, 3E80,15 ;19704          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19705                                          ; BIT 15 INDICATES START OF TEST TO
                ;19706                                          ; CONSOLE
U 1D81, 10E0,15 ;19707          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19708  WAIT.T34.0:
U 1D82, 89D8,24 ;19709          JMP [WAIT.T34.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19710                                          ; RESPOND
U 1D83, 887E,EC ;19711          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19712                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1D84, B778,15 ;19713          MOV LS[T34.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1D85, BE12,15 ;19714          MOV WR[0] TO LS[T9]              ; references.
U 1D86, B72E,15 ;19715          MOV LS[#2A7] TO WR[0]           ;Get LOAD address
U 1D87, BE14,15 ;19716          MOV WR[0] TO LS[T10]
U 1D88, 8870,3C ;19717          JSR [LO]                        ;Get MOV ETO TO EQ address
U 1D89, 3E1C,15 ;19718          MOV WR[0] TO LS[T14]
U 1D8A, 8870,3C ;19719          JSR [LO]                        ;Get MOV EQ TO ETO and CLOCK OP2=0
U 1D8B, BE1E,15 ;19720          MOV WR[0] TO LS[T15]            ; address
U 1D8C, 8870,3C ;19721          JSR [LO]                        ;Get CLOCK SIGN OUT WITH ONE address
U 1D8D, 3E10,15 ;19722          MOV WR[0] TO LS[T8]
```

```
U 1D8E, 8870,3C ;19723          JSR [LO]                        ;Get CLOCK SIGN OUT WITH ZERO address
U 1D8F, 3E16,15 ;19724          MOV WR[0] TO LS[T11]
U 1D90, 8870,3C ;19725          JSR [LO]                        ;Get Branch address
U 1D91, 3E1A,15 ;19726          MOV WR[0] TO LS[T13]
U 1D92, B610,15 ;19727          MOV LS[T8] TO WR[0]             ;Set SIGN OUT to one
U 1D93, B716,95 ;19728          MOV LS[#300] TO WR[1]
U 1D94, 90F6,15 ;19729          MISC2 [TRAP.ACC] WR[0]
U 1D95, 10F6,95 ;19730          MISC2 [TRAP.ACC] WR[1]
U 1D96, E5A2,15 ;19731  T34.1:  CLR LS[DATA.1]                  ;Set data to zero
U 1D97, 8877,DC ;19732          JSR [LOAD.DATA]                 ;Load the data to ET0
U 1D98, 0878,AC ;19733          JSR [MOV.DATA]                  ;Move the data from ET0 to EQ
U 1D99, 361E,15 ;19734          MOV LS[T15] TO WR[0]            ;Move the data from EQ to ET0
U 1D9A, 361A,95 ;19735          MOV LS[T13] TO WR[1]            ; and set OP2=0 and branch
U 1D9B, 90F6,15 ;19736          MISC2 [TRAP.ACC] WR[0]
U 1D9C, 10F6,95 ;19737          MISC2 [TRAP.ACC] WR[1]
U 1D9D, 10F8,15 ;19738          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1D9E, 3A18,15 ;19739          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1D9F, 3618,15 ;19740          MOV LS[T12] TO WR[0]            ;Set up received data
U 1DA0, 3742,95 ;19741          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1DA1, 0869,3C ;19742          JSR [CHECK.RESULT]              ;check for error
U 1DA2, 89D9,64 ;19743          JMP [T34.1]                     ;Loop on error
U 1DA3, FF82,15 ;19744          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1DA4, B61A,15 ;19745  T34.2:  MOV LS[T13] TO WR[0]            ;Branch
U 1DA5, 90F6,15 ;19746          MISC2 [TRAP.ACC] WR[0]
U 1DA6, 10F8,15 ;19747          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1DA7, 3A18,15 ;19748          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1DA8, 3618,15 ;19749          MOV LS[T12] TO WR[0]            ;Set up received data
U 1DA9, 3742,95 ;19750          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1DAA, 0869,3C ;19751          JSR [CHECK.RESULT]              ;check for error
U 1DAB, 09DA,44 ;19752          JMP [T34.2]                     ;Loop on error
U 1DAC, FF82,15 ;19753          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1DAD, B69E,15 ;19754  T34.3:  MOV LS[#FFFFFFFF] TO WR[0]      ;Get data to be loaded
U 1DAE, 3EA2,15 ;19755          MOV WR[0] TO LS[DATA.1]
U 1DAF, 8877,DC ;19756          JSR [LOAD.DATA]                 ;Load the data into ET0
U 1DB0, 0878,AC ;19757          JSR [MOV.DATA]                  ;Move the contents of ET0 to EQ
U 1DB1, B61A,15 ;19758          MOV LS[T13] TO WR[0]            ;Branch
U 1DB2, 90F6,15 ;19759          MISC2 [TRAP.ACC] WR[0]
U 1DB3, 10F8,15 ;19760          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1DB4, 3A18,15 ;19761          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1DB5, 3618,15 ;19762          MOV LS[T12] TO WR[0]            ;Set up received data
U 1DB6, 3742,95 ;19763          MOV LS[#303] TO WR[1]           ;Set up expected data
U 1DB7, 0869,3C ;19764          JSR [CHECK.RESULT]              ;check for error
U 1DB8, 09DA,D4 ;19765          JMP [T34.3]                     ;Loop on error
U 1DB9, FF82,15 ;19766          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1DBA, B616,15 ;19767          MOV LS[T11] TO WR[0]            ;CLOCK SIGN OUT WITH 0
U 1DBB, B716,95 ;19768          MOV LS[#300] TO WR[1]
U 1DBC, 90F6,15 ;19769          MISC2 [TRAP.ACC] WR[0]
U 1DBD, 10F6,95 ;19770          MISC2 [TRAP.ACC] WR[1]
U 1DBE, 361E,15 ;19771  T34.4:  MOV LS[T15] TO WR[0]            ;MOV EQ TO ET0 AND CLOCK OP2=0
U 1DBF, 361A,95 ;19772          MOV LS[T13] TO WR[1]            ;Branch
U 1DC0, 90F6,15 ;19773          MISC2 [TRAP.ACC] WR[0]
U 1DC1, 10F6,95 ;19774          MISC2 [TRAP.ACC] WR[1]
U 1DC2, 10F8,15 ;19775          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1DC3, 3A18,15 ;19776          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1DC4, 3618,15 ;19777          MOV LS[T12] TO WR[0]            ;Set up received data
```

```
U 1DC5, B716,95 ;19778          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1DC6, 0869,3C ;19779          JSR [CHECK.RESULT]             ;check for error
U 1DC7, 89DB,E4 ;19780          JMP [T34.4]                    ;Loop on error
U 1DC8, FF82,15 ;19781          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1DC9, B61A,15 ;19782  T34.5:  MOV LS[T13] TO WR[0]           ;Branch
U 1DCA, 90F6,15 ;19783          MISC2 [TRAP.ACC] WR[0]
U 1DCB, 10F8,15 ;19784          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1DCC, 3A18,15 ;19785          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1DCD, 3618,15 ;19786          MOV LS[T12] TO WR[0]           ;Set up received data
U 1DCE, B716,95 ;19787          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1DCF, 0869,3C ;19788          JSR [CHECK.RESULT]             ;check for error
U 1DD0, 89DC,94 ;19789          JMP [T34.5]                    ;Loop on error
                ;19790  T34.END:
```

```
                ;19791  .page
                ;19792  .TOC    "TEST 35 FRAC55 F3 BRANCH TEST(M8389 FPA MODULE)"
                ;19793  ;++
                ;19794  ;
                ;19795  ;
                ;19796  ; Test Description:
                ;19797  ;
                ;19798  ;       The purpose of this test is to check the FRAC55 F3 branch condition.
                ;19799  ;       The data path is as follows: FRAC55 F3 will be asserted and the branch
                ;19800  ;       field will be set to 01110. If no branch occurs or an incorrect branch
                ;19801  ;       occurs then an error statement will be issued. This procedure will be
                ;19802  ;       repeated for FRAC55 F3 unasserted. FRAC55 F3 will be asserted if the
                ;19803  ;       sign bit of the most significant 2901 in the fraction data path is as-
                ;19804  ;       serted.
                ;19805  ;
                ;19806  ; Assumptions:
                ;19807  ;
                ;19808  ;       It is assumed that the branch control logic has been checked previously
                ;19809  ;       and works.
                ;19810  ;
                ;19811  ; Test Steps:
                ;19812  ;
                ;19813  ;       1) Load a WR with 0's and another WR with 1 in the MSB. Subtract the WR
                ;19814  ;          with 1 from the WR with 0's. Force a nop from the FPA with the
                ;19815  ;          branch field set to 01110. If there is no branch or an incorrect
                ;19816  ;          branch then issue error1.
                ;19817  ;       2) Load two WR's with 0's and issue an OR of the WR's. Force a nop with
                ;19818  ;          the branch field set to 01110. If there is any branch at all then
                ;19819  ;          issue error2.
                ;19820  ;
                ;19821  ; Error:
                ;19822  ;
                ;19823  ;       Error1 - Branch failed when FRAC55 F3 asserted.
                ;19824  ;       Error2 - Branch failed when FRAC55 F3 unasserted.
                ;19825  ;
                ;19826  ; Debug:
                ;19827  ;
                ;19828  ;       Error1: If this error occurs then the BRANCH 1 PAL should be checked
                ;19829  ;               error . If the PAL isn't the cause of the error then the UBCTL
                ;19830  ;               bits should be checked to be 01110 and FRAC55 F3 SAVE should be
                ;19831  ;               checked to be asserted. If the UBCTL bits are in error the
                ;19832  ;               latch they come from should be checked for error. If FRAC55 F3
                ;19833  ;               SAVE is in error then the STATUS REG 1 LATCH should be checked
                ;19834  ;               for error. If the latch isn't the cause of the error then the
                ;19835  ;               sign bit of the 2901 should be checked for error.
                ;19836  ;       Error2: Same as error1 except that FRAC55 F3 SAVE is unasserted.
                ;19837  ;
                ;19838  ;
                ;19839  ;--
                ;19840  T.35:
U 1DD1, B65E,15 ;19841          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;19842                                          ; STATUS WORD
U 1DD2, 3E80,15 ;19843          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19844                                          ; BIT 15 INDICATES START OF TEST TO
                ;19845                                          ; CONSOLE
```

```
U 1DD3, 10E0,15 ;19846          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19847  WAIT.T35.0:
U 1DD4, 89DD,44 ;19848          JMP [WAIT.T35.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19849                                          ; RESPOND
U 1DD5, 887E,EC ;19850          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19851                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1DD6, 377A,15 ;19852          MOV LS[T35.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1DD7, BE12,15 ;19853          MOV WR[0] TO LS[T9]
U 1DD8, B72E,15 ;19854          MOV LS[#2A7] TO WR[0]           ;Set up address for load data in T10
U 1DD9, BE14,15 ;19855          MOV WR[0] TO LS[T10]
U 1DDA, 8870,3C ;19856          JSR [LO]                        ;Get address of MOV FT0 TO FT2
U 1DDB, 3E1C,15 ;19857          MOV WR[0] TO LS[T14]
U 1DDC, 8870,3C ;19858          JSR [LO]                        ;Get address of CLR FT0
U 1DDD, 3E16,15 ;19859          MOV WR[0] TO LS[T11]
U 1DDE, 8870,3C ;19860          JSR [LO]                        ;Get address of ADD FT0 TO FT2
U 1DDF, 3E10,15 ;19861          MOV WR[0] TO LS[T8]
U 1DE0, 8870,3C ;19862          JSR [LO]                        ;Get branch address
U 1DE1, 3E1A,15 ;19863          MOV WR[0] TO LS[T13]
U 1DE2, E5A2,15 ;19864  T35.1:  CLR LS[DATA.1]                  ;Set up the data to be loaded
U 1DE3, 8877,DC ;19865          JSR [LOAD.DATA]                 ;Load data into FT0
U 1DE4, 0878,AC ;19866          JSR [MOV.DATA]                  ;Mov the data to FT2
U 1DE5, B616,15 ;19867          MOV LS[T11] TO WR[0]            ;Set up to clear FT0
U 1DE6, B716,95 ;19868          MOV LS[#300] TO WR[1]
U 1DE7, 90F6,15 ;19869          MISC2 [TRAP.ACC] WR[0]          ;Clear FT0
U 1DE8, 10F6,95 ;19870          MISC2 [TRAP.ACC] WR[1]
U 1DE9, B610,15 ;19871          MOV LS[T8] TO WR[0]             ;Set up to add FT0 TO FT2
U 1DEA, 361A,95 ;19872          MOV LS[T13] TO WR[1]            ; and set up for branch
U 1DEB, 90F6,15 ;19873          MISC2 [TRAP.ACC] WR[0]          ;Add FT0 TO FT2
U 1DEC, 10F6,95 ;19874          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 1DED, 10F8,15 ;19875          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1DEE, 3A18,15 ;19876          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1DEF, 3618,15 ;19877          MOV LS[T12] TO WR[0]            ;Set up received data
U 1DF0, B740,95 ;19878          MOV LS[#302] TO WR[1]           ;Set up expected data
U 1DF1, 0869,3C ;19879          JSR [CHECK.RESULT]              ;check for error
U 1DF2, 89DE,24 ;19880          JMP [T35.1]                     ;Loop on error
U 1DF3, FF82,15 ;19881          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1DF4, B616,15 ;19882  T35.2:  MOV LS[T11] TO WR[0]            ;Set up to clear FT0
U 1DF5, B716,95 ;19883          MOV LS[#300] TO WR[1]
U 1DF6, 90F6,15 ;19884          MISC2 [TRAP.ACC] WR[0]          ;Clear FT0
U 1DF7, 10F6,95 ;19885          MISC2 [TRAP.ACC] WR[1]
U 1DF8, 0878,AC ;19886          JSR [MOV.DATA]                  ;MOV FT0 TO FT2
U 1DF9, B610,15 ;19887          MOV LS[T8] TO WR[0]             ;Set up for add
U 1DFA, 361A,95 ;19888          MOV LS[T13] TO WR[1]            ;Set up for branch
U 1DFB, 90F6,15 ;19889          MISC2 [TRAP.ACC] WR[0]          ;ADD FT0 TO FT2
U 1DFC, 10F6,95 ;19890          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 1DFD, 10F8,15 ;19891          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1DFE, 3A18,15 ;19892          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1DFF, 3618,15 ;19893          MOV LS[T12] TO WR[0]            ;Set up received data
U 1E00, B716,95 ;19894          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1E01, 0869,3C ;19895          JSR [CHECK.RESULT]              ;check for error
U 1E02, 09DF,44 ;19896          JMP [T35.2]                     ;Loop on error
                ;19897  T35.END:
```

```
;19898  .page
;19899  .TOC    "TEST 36 F47, F3 and EXT00 Q0 BRANCH TEST(M8389 FPA MODULE)"
;19900  ;++
;19901  ;
;19902  ;
;19903  ; Test Description:
;19904  ;
;19905  ;       The purpose of this test is to check the branch conditions, F47, F3 and
;19906  ;       EXT00 Q0. The data path is as follows: the branch conditions will be
;19907  ;       asserted and the branch field will be set to 10001. If there is no
;19908  ;       branch or an incorrect branch then an error statement will be issued.
;19909  ;       This procedure will be repeated for the branch conditions unasserted.
;19910  ;       F46, F3 will be asserted if the sign bit of the 2901 on the left hand
;19911  ;       side of page K is asserted. EXT00 Q00 will be asserted if the shift
;19912  ;       bits are 10.
;19913  ;
;19914  ; Assumptions:
;19915  ;
;19916  ;       It is assumed that the branch control logic has been tested previously
;19917  ;       and it works.
;19918  ;
;19919  ; Test Steps:
;19920  ;
;19921  ;       1) Load a WR so that only bit 47 is set. A first float load will cause
;19922  ;          shift field to be 00 causing 00Q0 to be 0. The first float load will
;19923  ;          be followed by a branch.
;19924  ;       2) Load FWR with zeros to clear F47 F3. Then issue a shift left FWR[]
;19925  ;          and FQ IN[ONE] to set 00Q0 then branch on the following instruction.
;19926  ;
;19927  ; Error:
;19928  ;
;19929  ;       Error1 - Branch failed when F47 F3 was asserted and EXT00 Q0 was
;19930  ;                unasserted.
;19931  ;       Error2 - Branch failed when F47 F3 was unasserted and EXT00 Q0 was
;19932  ;                asserted.
;19933  ;
;19934  ; Debug:
;19935  ;
;19936  ;       Error1: If this error occurs and F47, F3 failed then the BRANCH 2 PAL
;19937  ;               should be checked for error. If the PAL is not the cause of
;19938  ;               error then the Control Store latch should be checked for error.
;19939  ;               If the latch isn't the cause of error then the sign bit of the
;19940  ;               2901 should be checked for error. If this error occurs because
;19941  ;               EXT00 Q0 failed then the BRANCH 1 PAL should be checked for
;19942  ;               error. If the PAL is not the cause of the error then the STATUS
;19943  ;               REG 2 LATCH should be checked for error. If the latch is not
;19944  ;               the cause of the error then the MUL/DIV MUX should be checked
;19945  ;               for error. In either case the UBCTL bits that are inputs to the
;19946  ;               PALs should be checked to be 10001.
;19947  ;       Error2: Same as error 1 except that F47 F3 is unasserted and EXT00Q0 is
;19948  ;               asserted.
;19949  ;
;19950  ;
;19951  ;--
;19952  T.36:
```

```
U 1E03, B65E,15 ;19953          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;19954                                          ; STATUS WORD
U 1E04, 3E80,15 ;19955          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;19956                                          ; BIT 15 INDICATES START OF TEST TO
                ;19957                                          ; CONSOLE
U 1E05, 10E0,15 ;19958          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;19959  WAIT.T36.0:
U 1E06, 89E0,64 ;19960          JMP [WAIT.T36.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;19961                                          ; RESPOND
U 1E07, 887E,EC ;19962          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;19963                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1E08, 377C,15 ;19964          MOV LS[T36.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1E09, BE12,15 ;19965          MOV WR[0] TO LS[T9]
U 1E0A, B72E,15 ;19966          MOV LS[#2A7] TO WR[0]           ;Set up address for load data in T10
U 1E0B, BE14,15 ;19967          MOV WR[0] TO LS[T10]
U 1E0C, 8870,3C ;19968          JSR [LO]                        ;Get address to set 00Q0
U 1E0D, 3E16,15 ;19969          MOV WR[0] TO LS[T11]
U 1E0E, 8870,3C ;19970          JSR [LO]                        ;Get branch address
U 1E0F, 3E1A,15 ;19971          MOV WR[0] TO LS[T13]
U 1E10, 8870,3C ;19972          JSR [LO]                        ;Get MOV FTO TO FTO address
U 1E11, 3E1C,15 ;19973          MOV WR[0] TO LS[T14]
U 1E12, 8870,3C ;19974          JSR [LO]                        ;Get MOV FWR[D.RND] TO FWR[FTO]
U 1E13, BE1E,15 ;19975          MOV WR[0] TO LS[T15]
U 1E14, 367E,15 ;19976  T36.1:  MOV LS[BIT31] TO WR[0]          ;Set up data.1
U 1E15, 3EA2,15 ;19977          MOV WR[0] TO LS[DATA.1]
U 1E16, 8877,DC ;19978          JSR [LOAD.DATA]                 ;Load data into FTO
U 1E17, B61C,15 ;19979          MOV LS[T14] TO WR[0]            ;MOV FTO TO FTO
U 1E18, 361A,95 ;19980          MOV LS[T13] TO WR[1]            ;Branch on F47 F3 and 00Q0
U 1E19, 90F6,15 ;19981          MISC2 [TRAP.ACC] WR[0]
U 1E1A, 10F6,95 ;19982          MISC2 [TRAP.ACC] WR[1]
U 1E1B, 10F8,15 ;19983          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1E1C, 3A18,15 ;19984          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1E1D, 3618,15 ;19985          MOV LS[T12] TO WR[0]            ;Set up received data
U 1E1E, B740,95 ;19986          MOV LS[#302] TO WR[1]           ;Set up expected data
U 1E1F, 0869,3C ;19987          JSR [CHECK.RESULT]              ;check for error
U 1E20, 89E1,44 ;19988          JMP [T36.1]                     ;Loop on error
U 1E21, FF82,15 ;19989          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1E22, E5A2,15 ;19990  T36.2:  CLR LS[DATA.1]                  ;Set up data for load
U 1E23, 8877,DC ;19991          JSR [LOAD.DATA]                 ;Load data
U 1E24, 361E,15 ;19992          MOV LS[T15] TO WR[0]            ;Load FWR[D.RND] with 0
U 1E25, B716,95 ;19993          MOV LS[#300] TO WR[1]
U 1E26, 90F6,15 ;19994          MISC2 [TRAP.ACC] WR[0]
U 1E27, 10F6,95 ;19995          MISC2 [TRAP.ACC] WR[1]
U 1E28, B616,15 ;19996          MOV LS[T11] TO WR[0]            ;Set up word to set 00Q0
U 1E29, 361A,95 ;19997          MOV LS[T13] TO WR[1]            ;Set up word for branch
U 1E2A, 90F6,15 ;19998          MISC2 [TRAP.ACC] WR[0]
U 1E2B, 10F6,95 ;19999          MISC2 [TRAP.ACC] WR[1]
U 1E2C, 10F8,15 ;20000          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1E2D, 3A18,15 ;20001          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1E2E, 3618,15 ;20002          MOV LS[T12] TO WR[0]            ;Set up received data
U 1E2F, B73E,95 ;20003          MOV LS[#301] TO WR[1]           ;Set up expected data
U 1E30, 0869,3C ;20004          JSR [CHECK.RESULT]              ;check for error
U 1E31, 89E2,24 ;20005          JMP [T36.2]                     ;Loop on error
                ;20006  T36.END:
```

```
;20007  .page
;20008  .TOC    "TEST 37 FRAC<47:16>=0 AND ZERO BRANCH TEST(M8389 FPA MODULE)"
;20009  ;++
;20010  ;
;20011  ;
;20012  ; Test Description:
;20013  ;
;20014  ;       The purpose of this test is to check the FRAC<47:16>=0 branch condition
;20015  ;       The data path is as follows: the branch conditicn will be asserted with
;20016  ;       the branch field set to 10011. If there is no branch or an incorrect
;20017  ;       branch then an error statement will be issued. This procedure will be
;20018  ;       repeated for the branch condition unasserted. FRAC<47:16>=0 will be
;20019  ;       assserted if there are zeros in bits 16 - 47.
;20020  ;
;20021  ; Assumptions:
;20022  ;
;20023  ;       It is assumed that the branch control logic has been tested previously
;20024  ;       and works.
;20025  ;
;20026  ; Test Steps:
;20027  ;
;20028  ;       1) Load a WR in the fraction data path with zeros in bits 16 - 47 and
;20029  ;          ones in the remaining bits and a second WR with zeros. OR the two
;20030  ;          WR's together with the branch field set to 10011. If there is no
;20031  ;          branch or an incorrect branch then issue error1.
;20032  ;       2) Load FTO with bit 47 set and then branch
;20033  ;       3) Repeat step 2 with bit 43 set
;20034  ;       4) Repeat step 2 with bit 39 set
;20035  ;       5) Repeat step 2 with bit 35 set
;20036  ;       6) Repeat step 2 with bit 31 set
;20037  ;       7) Repeat step 2 with bit 27 set
;20038  ;       8) Repeat step 2 with bit 23 set
;20039  ;       9) Repeat step 2 with bit 19 set
;20040  ;
;20041  ; Error:
;20042  ;
;20043  ;       Error1 - Branch failed when FRAC<47:16>=0 asserted.
;20044  ;       Error2 - Branch failed when FRAC<47:16>=0 unasserted.
;20045  ;       Error3 - Branch failed when FRAC<47:16>=0 unasserted.
;20046  ;       Error4 - Branch failed when FRAC<47:16>=0 unasserted.
;20047  ;       Error5 - Branch failed when FRAC<47:16>=0 unasserted.
;20048  ;       Error6 - Branch failed when FRAC<47:16>=0 unasserted.
;20049  ;       Error7 - Branch failed when FRAC<47:16>=0 unasserted.
;20050  ;       Error8 - Branch failed when FRAC<47:16>=0 unasserted.
;20051  ;       Error9 - Branch failed when FRAC<47:16>=0 unasserted.
;20052  ;
;20053  ; Debug:
;20054  ;
;20055  ;       Error1: If this error occurs then the BRANCH 2 PAL should be checked
;20056  ;               for error. If the PAL is not the cause of error then FRAC
;20057  ;               <47:32>=0 and FRAC<31:16>=0 should be checked to be asserted
;20058  ;               and the UBCTL bits should be checked to be 10011. If either
;20059  ;               FRAC<47:32>=0 or FRAC<31:16>=0 are in error then the STATUS REG
;20060  ;               2 LATCH should be checked for error. If the  latch isn't the
;20061  ;               cause of the error then the sign bits from the 2901's should
```

```
                    ;20062  ;               be checked for error. If UBCTL bits are in error then the
                    ;20063  ;               Control Store Latch should be checked for error.
                    ;20064  ;       Error2: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20065  ;       Error3: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20066  ;       Error4: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20067  ;       Error5: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20068  ;       Error6: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20069  ;       Error7: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20070  ;       Error8: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20071  ;       Error9: Same as error1 except that FRAC<47:16>=0 is unasserted.
                    ;20072  ;
                    ;20073  ;
                    ;20074  ;--
                    ;20075  T.37:
U 1E32, B65E,15     ;20076          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;20077                                          ; STATUS WORD
U 1E33, 3E80,15     ;20078          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;20079                                          ; BIT 15 INDICATES START OF TEST TO
                    ;20080                                          ; CONSOLE
U 1E34, 10E0,15     ;20081          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;20082  WAIT.T37.0:
U 1E35, 89E3,54     ;20083          JMP [WAIT.T37.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;20084                                          ; RESPOND
U 1E36, 887E,EC     ;20085          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;20086                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1E37, B77E,15     ;20087          MOV LS[T37.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1E38, BE12,15     ;20088          MOV WR[0] TO LS[T9]
U 1E39, 8870,3C     ;20089          JSR [L0]                        ;Get branch address
U 1E3A, 3E1A,15     ;20090          MOV WR[0] TO LS[T13]
U 1E3B, 8870,3C     ;20091          JSR [L0]                        ;Get MOV FT0 TO FT0 address
U 1E3C, 3E16,15     ;20092          MOV WR[0] TO LS[T11]
U 1E3D, 087C,1C     ;20093  T37.1:  JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 1E3E, B616,15     ;20094          MOV LS[T11] TO WR[0]            ;Set up MOV FT0 TO FT0
U 1E3F, 361A,95     ;20095          MOV LS[T13] TO WR[1]            ;Set up branch
U 1E40, 90F6,15     ;20096          MISC2 [TRAP.ACC] WR[0]
U 1E41, 10F6,95     ;20097          MISC2 [TRAP.ACC] WR[1]
U 1E42, 10F8,15     ;20098          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1E43, 3A18,15     ;20099          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1E44, 3618,15     ;20100          MOV LS[T12] TO WR[0]            ;Set up received data
U 1E45, B740,95     ;20101          MOV LS[#302] TO WR[1]           ;Set up expected data
U 1E46, 0869,3C     ;20102          JSR [CHECK.RESULT]              ;check for error
U 1E47, 09E3,D4     ;20103          JMP [T37.1]                     ;Loop on error
U 1E48, FF82,15     ;20104          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1E49, 65A6,15     ;20105  T37.2:  CLR LS[DATA.3]                  ;Set up data to be passed
U 1E4A, E5A4,15     ;20106          CLR LS[DATA.2]
U 1E4B, 367E,15     ;20107          MOV LS[BIT31] TO WR[0]
U 1E4C, 3EA2,15     ;20108          MOV WR[0] TO LS[DATA.1]
U 1E4D, 0878,FC     ;20109          JSR [LOAD.TRIPLE]               ;Load the data into ET0 and FT0
U 1E4E, B616,15     ;20110          MOV LS[T11] TO WR[0]            ;Set up MOV FT0 TO FT0
U 1E4F, 361A,95     ;20111          MOV LS[T13] TO WR[1]            ;Set up branch
U 1E50, 90F6,15     ;20112          MISC2 [TRAP.ACC] WR[0]
U 1E51, 10F6,95     ;20113          MISC2 [TRAP.ACC] WR[1]
U 1E52, 10F8,15     ;20114          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1E53, 3A18,15     ;20115          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1E54, 3618,15     ;20116          MOV LS[T12] TO WR[0]            ;Set up received data
```

```
U 1E55, B716,95 ;20117          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1E56, 0869,3C ;20118          JSR [CHECK.RESULT]             ;check for error
U 1E57, 09E4,94 ;20119          JMP [T37.2]                    ;Loop on error
U 1E58, FF82,15 ;20120          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1E59, 65A6,15 ;20121  T37.3:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1E5A, E5A4,15 ;20122          CLR LS[DATA.2]
U 1E5B, B676,15 ;20123          MOV LS[BIT27] TO WR[0]
U 1E5C, 3EA2,15 ;20124          MOV WR[0] TO LS[DATA.1]
U 1E5D, 0878,FC ;20125          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1E5E, B616,15 ;20126          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 1E5F, 361A,95 ;20127          MOV LS[T13] TO WR[1]           ;Set up branch
U 1E60, 90F6,15 ;20128          MISC2 [TRAP.ACC] WR[0]
U 1E61, 10F6,95 ;20129          MISC2 [TRAP.ACC] WR[1]
U 1E62, 10F8,15 ;20130          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1E63, 3A18,15 ;20131          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1E64, 3618,15 ;20132          MOV LS[T12] TO WR[0]           ;Set up received data
U 1E65, B716,95 ;20133          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1E66, 0869,3C ;20134          JSR [CHECK.RESULT]             ;check for error
U 1E67, 89E5,94 ;20135          JMP [T37.3]                    ;Loop on error
U 1E68, FF82,15 ;20136          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1E69, 65A6,15 ;20137  T37.4:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1E6A, E5A4,15 ;20138          CLR LS[DATA.2]
U 1E6B, B66E,15 ;20139          MOV LS[BIT23] TO WR[0]
U 1E6C, 3EA2,15 ;20140          MOV WR[0] TO LS[DATA.1]
U 1E6D, 0878,FC ;20141          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1E6E, B616,15 ;20142          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 1E6F, 361A,95 ;20143          MOV LS[T13] TO WR[1]           ;Set up branch
U 1E70, 90F6,15 ;20144          MISC2 [TRAP.ACC] WR[0]
U 1E71, 10F6,95 ;20145          MISC2 [TRAP.ACC] WR[1]
U 1E72, 10F8,15 ;20146          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1E73, 3A18,15 ;20147          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1E74, 3618,15 ;20148          MOV LS[T12] TO WR[0]           ;Set up received data
U 1E75, B716,95 ;20149          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1E76, 0869,3C ;20150          JSR [CHECK.RESULT]             ;check for error
U 1E77, 89E6,94 ;20151          JMP [T37.4]                    ;Loop on error
U 1E78, FF82,15 ;20152          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1E79, 65A6,15 ;20153  T37.5:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1E7A, E5A4,15 ;20154          CLR LS[DATA.2]
U 1E7B, 3666,15 ;20155          MOV LS[BIT19] TO WR[0]
U 1E7C, 3EA2,15 ;20156          MOV WR[0] TO LS[DATA.1]
U 1E7D, 0878,FC ;20157          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1E7E, B616,15 ;20158          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 1E7F, 361A,95 ;20159          MOV LS[T13] TO WR[1]           ;Set up branch
U 1E80, 90F6,15 ;20160          MISC2 [TRAP.ACC] WR[0]
U 1E81, 10F6,95 ;20161          MISC2 [TRAP.ACC] WR[1]
U 1E82, 10F8,15 ;20162          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1E83, 3A18,15 ;20163          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1E84, 3618,15 ;20164          MOV LS[T12] TO WR[0]           ;Set up received data
U 1E85, B716,95 ;20165          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1E86, 0869,3C ;20166          JSR [CHECK.RESULT]             ;check for error
U 1E87, 09E7,94 ;20167          JMP [T37.5]                    ;Loop on error
U 1E88, FF82,15 ;20168          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1E89, 65A6,15 ;20169  T37.6:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1E8A, E5A2,15 ;20170          CLR LS[DATA.1]
U 1E8B, B65E,15 ;20171          MOV LS[BIT15] TO WR[0]
```

```
U 1E8C, 3EA4,15 ;20172          MOV WR[0] TO LS[DATA.2]
U 1E8D, 0878,FC ;20173          JSR [LOAD.TRIPLE]           ;Load the data into ET0 and FT0
U 1E8E, B616,15 ;20174          MOV LS[T11] TO WR[0]        ;Set up MOV FT0 TO FT0
U 1E8F, 361A,95 ;20175          MOV LS[T13] TO WR[1]        ;Set up branch
U 1E90, 90F6,15 ;20176          MISC2 [TRAP.ACC] WR[0]
U 1E91, 10F6,95 ;20177          MISC2 [TRAP.ACC] WR[1]
U 1E92, 10F8,15 ;20178          MISC2 [READ.ACC.UPC]        ;Enable the NUA BUS onto the FPA BUS
U 1E93, 3A18,15 ;20179          MOV FPA TO LS[T12]          ;Enable the FPA BUS onto the Y BUS
U 1E94, 3618,15 ;20180          MOV LS[T12] TO WR[0]        ;Set up received data
U 1E95, B716,95 ;20181          MOV LS[#300] TO WR[1]       ;Set up expected data
U 1E96, 0869,3C ;20182          JSR [CHECK.RESULT]          ;check for error
U 1E97, 09E8,94 ;20183          JMP [T37.6]                 ;Loop on error
U 1E98, FF82,15 ;20184          INC LS[ERROR.NUMBER]        ;Go onto the next error number
U 1E99, 65A6,15 ;20185  T37.7:  CLR LS[DATA.3]              ;Set up data to be passed
U 1E9A, E5A2,15 ;20186          CLR LS[DATA.1]
U 1E9B, 3656,15 ;20187          MOV LS[BIT11] TO WR[0]
U 1E9C, 3EA4,15 ;20188          MOV WR[0] TO LS[DATA.2]
U 1E9D, 0878,FC ;20189          JSR [LOAD.TRIPLE]           ;Load the data into ET0 and FT0
U 1E9E, B616,15 ;20190          MOV LS[T11] TO WR[0]        ;Set up MOV FT0 TO FT0
U 1E9F, 361A,95 ;20191          MOV LS[T13] TO WR[1]        ;Set up branch
U 1EA0, 90F6,15 ;20192          MISC2 [TRAP.ACC] WR[0]
U 1EA1, 10F6,95 ;20193          MISC2 [TRAP.ACC] WR[1]
U 1EA2, 10F8,15 ;20194          MISC2 [READ.ACC.UPC]        ;Enable the NUA BUS onto the FPA BUS
U 1EA3, 3A18,15 ;20195          MOV FPA TO LS[T12]          ;Enable the FPA BUS onto the Y BUS
U 1EA4, 3618,15 ;20196          MOV LS[T12] TO WR[0]        ;Set up received data
U 1EA5, B716,95 ;20197          MOV LS[#300] TO WR[1]       ;Set up expected data
U 1EA6, 0869,3C ;20198          JSR [CHECK.RESULT]          ;check for error
U 1EA7, 89E9,94 ;20199          JMP [T37.7]                 ;Loop on error
U 1EA8, FF82,15 ;20200          INC LS[ERROR.NUMBER]        ;Go onto the next error number
U 1EA9, 65A6,15 ;20201  T37.8:  CLR LS[DATA.3]              ;Set up data to be passed
U 1EAA, E5A2,15 ;20202          CLR LS[DATA.1]
U 1EAB, 364E,15 ;20203          MOV LS[BIT7] TO WR[0]
U 1EAC, 3EA4,15 ;20204          MOV WR[0] TO LS[DATA.2]
U 1EAD, 0878,FC ;20205          JSR [LOAD.TRIPLE]           ;Load the data into ET0 and FT0
U 1EAE, B616,15 ;20206          MOV LS[T11] TO WR[0]        ;Set up MOV FT0 TO FT0
U 1EAF, 361A,95 ;20207          MOV LS[T13] TO WR[1]        ;Set up branch
U 1EB0, 90F6,15 ;20208          MISC2 [TRAP.ACC] WR[0]
U 1EB1, 10F6,95 ;20209          MISC2 [TRAP.ACC] WR[1]
U 1EB2, 10F8,15 ;20210          MISC2 [READ.ACC.UPC]        ;Enable the NUA BUS onto the FPA BUS
U 1EB3, 3A18,15 ;20211          MOV FPA TO LS[T12]          ;Enable the FPA BUS onto the Y BUS
U 1EB4, 3618,15 ;20212          MOV LS[T12] TO WR[0]        ;Set up received data
U 1EB5, B716,95 ;20213          MOV LS[#300] TO WR[1]       ;Set up expected data
U 1EB6, 0869,3C ;20214          JSR [CHECK.RESULT]          ;check for error
U 1EB7, 89EA,94 ;20215          JMP [T37.8]                 ;Loop on error
U 1EB8, FF82,15 ;20216          INC LS[ERROR.NUMBER]        ;Go onto the next error number
U 1EB9, 65A6,15 ;20217  T37.9:  CLR LS[DATA.3]              ;Set up data to be passed
U 1EBA, E5A2,15 ;20218          CLR LS[DATA.1]
U 1EBB, B646,15 ;20219          MOV LS[BIT3] TO WR[0]
U 1EBC, 3EA4,15 ;20220          MOV WR[0] TO LS[DATA.2]
U 1EBD, 0878,FC ;20221          JSR [LOAD.TRIPLE]           ;Load the data into ET0 and FT0
U 1EBE, B616,15 ;20222          MOV LS[T11] TO WR[0]        ;Set up MOV FT0 TO FT0
U 1EBF, 361A,95 ;20223          MOV LS[T13] TO WR[1]        ;Set up branch
U 1EC0, 90F6,15 ;20224          MISC2 [TRAP.ACC] WR[0]
U 1EC1, 10F6,95 ;20225          MISC2 [TRAP.ACC] WR[1]
U 1EC2, 10F8,15 ;20226          MISC2 [READ.ACC.UPC]        ;Enable the NUA BUS onto the FPA BUS
```

```
U 1EC3, 3A18,15 ;20227          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1EC4, 3618,15 ;20228          MOV LS[T12] TO WR[0]            ;Set up received data
U 1EC5, B716,95 ;20229          MOV LS[#300] TO WR[1]           ;Set up expected data
U 1EC6, 0869,3C ;20230          JSR [CHECK.RESULT]              ;check for error
U 1EC7, 09EB,94 ;20231          JMP [T37.9]                     ;Loop on error
                ;20232  T37.END:
```

```
;20233  .page
;20234  .TOC    "TEST 38 FRAC<55:00>=0 AND CPU RCV DATA BRANCH TEST(M8389 FPA MODULE)"
;20235  ;++
;20236  ;
;20237  ;
;20238  ; Test Description:
;20239  ;
;20240  ;       The purpose of this test is to check the FRAC<55:00>=0 and CPU RCV DATA
;20241  ;       branch conditions. The data path will be as follows: the branch condi-
;20242  ;       tions will be asserted with the branch field set to 10100. This proce-
;20243  ;       dure will be repeated with the branch conditions unasserted. If there
;20244  ;       are no branches or incorrect branches then the an error statement will
;20245  ;       be issued. FRAC<55:00>=0 will be asserted if bits 0 - 55 are zero. CPU
;20246  ;       RCV DATA will be asserted if the CPU issues a MISC instruction with bit
;20247  ;       18 clear and the MISC FUN 1 field set to SET PORT I/O MODE.
;20248  ;
;20249  ; Assumptions:
;20250  ;
;20251  ;       It is assumed that the branch control logic has been tested previously
;20252  ;       and works.
;20253  ;
;20254  ; Test Steps:
;20255  ;
;20256  ;       1) Issue a MISC instruction to load a WR with zeros. Issue a MISC
;20257  ;          instruction to AND the WR with zero. Issue a third MISC isntruction
;20258  ;          force a nop with the branch field to 10101. Then issue a MISC
;20259  ;          instruction in the CPU that will asserted CPU RCV DATA. This should
;20260  ;          cause branches on both conditions. If there is no branch or an
;20261  ;          incorrect branch then issue error 1.
;20262  ;       2) Load FT0 with zeros. This will cause the hidden bit to be set. Then
;20263  ;          issue a branch on FRAC<55:0>.
;20264  ;       3) Load FT0 with bit 2 set in the first float. Shift FT0 LEFT and ZERO
;20265  ;          in. Then branch on FRAC<55:0>.
;20266  ;       4) Repeat step 3 with bit 30 set in the first float.
;20267  ;       5) Repeat step 3 with bit 14 set in the second float.
;20268  ;       6) Repeat step 3 with bit 18 set in the second float.
;20269  ;       7) Repeat step 3 with bit 22 set in the second float.
;20270  ;       8) Repeat step 3 with bit 26 set in the second float.
;20271  ;       9) Repeat step 3 with bit 30 set in the second float.
;20272  ;
;20273  ; Error:
;20274  ;
;20275  ;       Error1 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA asserted.
;20276  ;       Error2 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20277  ;       Error3 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20278  ;       Error4 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20279  ;       Error5 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20280  ;       Error6 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20281  ;       Error7 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20282  ;       Error8 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20283  ;       Error9 - Branch failed when FRAC<55:00>=0 and CPU RCV DATA unasserted.
;20284  ;
;20285  ; Debug:
;20286  ;
;20287  ;       Error1: If this error occurs because FRAC<55:00>=0 failed then the
```

```
                ;20288  ;               BRANCH 2 PAL should be checked for error. If the PAL is not
                ;20289  ;               the cause of the error then the UBCTL bits should be checked
                ;20290  ;               to be 10101 and FRAC<55:48>=0, FRAC<47:32>=0, FRAC<31:16>=0,
                ;20291  ;               and FRAC<15:00>=0 should be checked to be asserted. If FRAC
                ;20292  ;               <15:00>=0 is in error then the STATUS REG 2 LATCH should be
                ;20293  ;               checked for error. If the latch isn't in error then the sign
                ;20294  ;               bit of the 2901 should be checked for error. If this error
                ;20295  ;               occurs because CPU RCV DATA failed then te BRANCH 1 PAL should
                ;20296  ;               be checked for error. If the PAL is not the source of error
                ;20297  ;               then CPU RCV DATA should be checked to be asserted. If CPU RCV
                ;20298  ;               DATA is in error then the gates at it comes from should be
                ;20299  ;               checked for error. The UBCTL bits that are inputs to BRANCH 1
                ;20300  ;               PAL should be checked to be 10100.
                ;20301  ;       Error2: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20302  ;               unasserted.
                ;20303  ;       Error3: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20304  ;               unasserted.
                ;20305  ;       Error4: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20306  ;               unasserted.
                ;20307  ;       Error5: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20308  ;               unasserted.
                ;20309  ;       Error6: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20310  ;               unasserted.
                ;20311  ;       Error7: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20312  ;               unasserted.
                ;20313  ;       Error8: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20314  ;               unasserted.
                ;20315  ;       Error9: Same as error1 except that FRAC<55:00>=0 and CPU RCV DATA are
                ;20316  ;               unasserted.
                ;20317  ;
                ;20318  ;
                ;20319  ;--
                ;20320  T.38:
U 1EC8, B65E,15 ;20321          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;20322                                          ; STATUS WORD
U 1EC9, 3E80,15 ;20323          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;20324                                          ; BIT 15 INDICATES START OF TEST TO
                ;20325                                          ; CONSOLE
U 1ECA, 10E0,15 ;20326          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;20327  WAIT.T38.0:
U 1ECB, 09EC,B4 ;20328          JMP [WAIT.T38.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;20329                                          ; RESPOND
U 1ECC, 887E,EC ;20330          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;20331                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1ECD, 3780,15 ;20332          MOV LS[T38.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1ECE, BE12,15 ;20333          MOV WR[0] TO LS[T9]
U 1ECF, B72E,15 ;20334          MOV LS[#2A7] TO WR[0]           ;Set up address for load data in T10
U 1ED0, BE14,15 ;20335          MOV WR[0] TO LS[T10]
U 1ED1, 8870,3C ;20336          JSR [L0]                        ;Get branch address
U 1ED2, 3E1A,15 ;20337          MOV WR[0] TO LS[T13]
U 1ED3, 8870,3C ;20338          JSR [L0]                        ;Get MOV FT0 TO FT0 address
U 1ED4, 3E16,15 ;20339          MOV WR[0] TO LS[T11]
U 1ED5, 8870,3C ;20340          JSR [L0]                        ;Set up CLR FT0
U 1ED6, 3E1C,15 ;20341          MOV WR[0] TO LS[T14]
U 1ED7, 8870,3C ;20342          JSR [L0]                        ;Set up SHIFT LEFT FWR[FT0] IN[ZERO]
```

```
U 1ED8, BE1E,15 ;20343          MOV WR[0] TO LS[T15]
U 1ED9, B61C,15 ;20344  T38.1:  MOV LS[T14] TO WR[0]          ;Set up CLR FT0
U 1EDA, 361A,95 ;20345          MOV LS[T13] TO WR[1]          ;Set up branch
U 1EDB, 90F6,15 ;20346          MISC2 [TRAP.ACC] WR[0]
U 1EDC, 10F6,95 ;20347          MISC2 [TRAP.ACC] WR[1]
U 1EDD, 3A18,15 ;20348          MOV FPA TO LS[T12]            ;Assert CPU RCV DATA
U 1EDE, 10F8,15 ;20349          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1EDF, 3A18,15 ;20350          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1EE0, 3618,15 ;20351          MOV LS[T12] TO WR[0]          ;Set up received data
U 1EE1, 3742,95 ;20352          MOV LS[#303] TO WR[1]         ;Set up expected data
U 1EE2, 0869,3C ;20353          JSR [CHECK.RESULT]            ;check for error
U 1EE3, 09ED,94 ;20354          JMP [T38.1]                   ;Loop on error
U 1EE4, FF82,15 ;20355          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1EE5, 087C,1C ;20356  T38.2:  JSR [CLR.FT0]                 ;LOAD ZEROS INTO FT0
U 1EE6, B616,15 ;20357          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 1EE7, 361A,95 ;20358          MOV LS[T13] TO WR[1]          ;Set up branch
U 1EE8, 90F6,15 ;20359          MISC2 [TRAP.ACC] WR[0]
U 1EE9, 10F6,95 ;20360          MISC2 [TRAP.ACC] WR[1]
U 1EEA, 10F8,15 ;20361          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1EEB, 3A18,15 ;20362          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1EEC, 3618,15 ;20363          MOV LS[T12] TO WR[0]          ;Set up received data
U 1EED, B716,95 ;20364          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1EEE, 0869,3C ;20365          JSR [CHECK.RESULT]            ;check for error
U 1EEF, 09EE,54 ;20366          JMP [T38.2]                   ;Loop on error
U 1EF0, FF82,15 ;20367          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1EF1, 65A6,15 ;20368  T38.3:  CLR LS[DATA.3]                ;Set up data to be passed
U 1EF2, E5A4,15 ;20369          CLR LS[DATA.2]
U 1EF3, 3644,15 ;20370          MOV LS[BIT2] TO WR[0]
U 1EF4, 3EA2,15 ;20371          MOV WR[0] TO LS[DATA.1]
U 1EF5, 0878,FC ;20372          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 1EF6, 361E,15 ;20373          MOV LS[T15] TO WR[0]          ;Shift left one
U 1EF7, B716,95 ;20374          MOV LS[#300] TO WR[1]
U 1EF8, 90F6,15 ;20375          MISC2 [TRAP.ACC] WR[0]
U 1EF9, 10F6,95 ;20376          MISC2 [TRAP.ACC] WR[1]
U 1EFA, B616,15 ;20377          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 1EFB, 361A,95 ;20378          MOV LS[T13] TO WR[1]          ;Set up branch
U 1EFC, 90F6,15 ;20379          MISC2 [TRAP.ACC] WR[0]
U 1EFD, 10F6,95 ;20380          MISC2 [TRAP.ACC] WR[1]
U 1EFE, 10F8,15 ;20381          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1EFF, 3A18,15 ;20382          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F00, 3618,15 ;20383          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F01, B716,95 ;20384          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F02, 0869,3C ;20385          JSR [CHECK.RESULT]            ;check for error
U 1F03, 09EF,14 ;20386          JMP [T38.3]                   ;Loop on error
U 1F04, FF82,15 ;20387          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F05, 65A6,15 ;20388  T38.4:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F06, E5A4,15 ;20389          CLR LS[DATA.2]
U 1F07, B67C,15 ;20390          MOV LS[BIT30] TO WR[0]
U 1F08, 3EA2,15 ;20391          MOV WR[0] TO LS[DATA.1]
U 1F09, 0878,FC ;20392          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 1F0A, 361E,15 ;20393          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F0B, B716,95 ;20394          MOV LS[#300] TO WR[1]
U 1F0C, 90F6,15 ;20395          MISC2 [TRAP.ACC] WR[0]
U 1F0D, 10F6,95 ;20396          MISC2 [TRAP.ACC] WR[1]
U 1F0E, B616,15 ;20397          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
```

```
U 1F0F, 361A.95 ;20398          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F10, 90F6.15 ;20399          MISC2 [TRAP.ACC] WR[0]
U 1F11, 10F6.95 ;20400          MISC2 [TRAP.ACC] WR[1]
U 1F12, 10F8.15 ;20401          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F13, 3A18.15 ;20402          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F14, 3618.15 ;20403          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F15, B716.95 ;20404          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F16, 0869.3C ;20405          JSR [CHECK.RESULT]            ;check for error
U 1F17, 09F0.54 ;20406          JMP [T38.4]                   ;Loop on error
U 1F18, FF82.15 ;20407          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F19, 65A6.15 ;20408  T38.5:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F1A, E5A2.15 ;20409          CLR LS[DATA.1]
U 1F1B, 365C.15 ;20410          MOV LS[BIT14] TO WR[0]
U 1F1C, 3EA4.15 ;20411          MOV WR[0] TO LS[DATA.2]
U 1F1D, 0878.FC ;20412          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 1F1E, 361E.15 ;20413          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F1F, B716.95 ;20414          MOV LS[#300] TO WR[1]
U 1F20, 90F6.15 ;20415          MISC2 [TRAP.ACC] WR[0]
U 1F21, 10F6.95 ;20416          MISC2 [TRAP.ACC] WR[1]
U 1F22, B616.15 ;20417          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 1F23, 361A.95 ;20418          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F24, 90F6.15 ;20419          MISC2 [TRAP.ACC] WR[0]
U 1F25, 10F6.95 ;20420          MISC2 [TRAP.ACC] WR[1]
U 1F26, 10F8.15 ;20421          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F27, 3A18.15 ;20422          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F28, 3618.15 ;20423          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F29, B716.95 ;20424          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F2A, 0869.3C ;20425          JSR [CHECK.RESULT]            ;check for error
U 1F2B, 89F1.94 ;20426          JMP [T38.5]                   ;Loop on error
U 1F2C, FF82.15 ;20427          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F2D, 65A6.15 ;20428  T38.6:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F2E, E5A2.15 ;20429          CLR LS[DATA.1]
U 1F2F, B664.15 ;20430          MOV LS[BIT18] TO WR[0]
U 1F30, 3EA4.15 ;20431          MOV WR[0] TO LS[DATA.2]
U 1F31, 0878.FC ;20432          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 1F32, 361E.15 ;20433          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F33, B716.95 ;20434          MOV LS[#300] TO WR[1]
U 1F34, 90F6.15 ;20435          MISC2 [TRAP.ACC] WR[0]
U 1F35, 10F6.95 ;20436          MISC2 [TRAP.ACC] WR[1]
U 1F36, B616.15 ;20437          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 1F37, 361A.95 ;20438          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F38, 90F6.15 ;20439          MISC2 [TRAP.ACC] WR[0]
U 1F39, 10F6.95 ;20440          MISC2 [TRAP.ACC] WR[1]
U 1F3A, 10F8.15 ;20441          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F3B, 3A18.15 ;20442          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F3C, 3618.15 ;20443          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F3D, B716.95 ;20444          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F3E, 0869.3C ;20445          JSR [CHECK.RESULT]            ;check for error
U 1F3F, 09F2.D4 ;20446          JMP [T38.6]                   ;Loop on error
U 1F40, FF82.15 ;20447          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F41, 65A6.15 ;20448  T38.7:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F42, E5A2.15 ;20449          CLR LS[DATA.1]
U 1F43, 366C.15 ;20450          MOV LS[BIT22] TO WR[0]
U 1F44, 3EA4.15 ;20451          MOV WR[0] TO LS[DATA.2]
U 1F45, 0878.FC ;20452          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
```

```
U 1F46, 361E.15 ;20453          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F47, B716.95 ;20454          MOV LS[#300] TO WR[1]
U 1F48, 90F6.15 ;20455          MISC2 [TRAP.ACC] WR[0]
U 1F49, 10F6.95 ;20456          MISC2 [TRAP.ACC] WR[1]
U 1F4A, B616.15 ;20457          MOV LS[T11] TO WR[0]          ;Set up MOV FTO TO FTO
U 1F4B, 361A.95 ;20458          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F4C, 90F6.15 ;20459          MISC2 [TRAP.ACC] WR[0]
U 1F4D, 10F6.95 ;20460          MISC2 [TRAP.ACC] WR[1]
U 1F4E, 10F8.15 ;20461          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F4F, 3A18.15 ;20462          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F50, 3618.15 ;20463          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F51, B716.95 ;20464          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F52, 0869.3C ;20465          JSR [CHECK.RESULT]            ;check for error
U 1F53, 09F4.14 ;20466          JMP [T38.7]                   ;Loop on error
U 1F54, FF82.15 ;20467          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F55, 65A6.15 ;20468  T38.8:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F56, E5A2.15 ;20469          CLR LS[DATA.1]
U 1F57, 3674.15 ;20470          MOV LS[BIT26] TO WR[0]
U 1F58, 3EA4.15 ;20471          MOV WR[0] TO LS[DATA.2]
U 1F59, 0878.FC ;20472          JSR [LOAD.TRIPLE]             ;Load the data into ETO and FTO
U 1F5A, 361E.15 ;20473          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F5B, B716.95 ;20474          MOV LS[#300] TO WR[1]
U 1F5C, 90F6.15 ;20475          MISC2 [TRAP.ACC] WR[0]
U 1F5D, 10F6.95 ;20476          MISC2 [TRAP.ACC] WR[1]
U 1F5E, B616.15 ;20477          MOV LS[T11] TO WR[0]          ;Set up MOV FTO TO FTO
U 1F5F, 361A.95 ;20478          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F60, 90F6.15 ;20479          MISC2 [TRAP.ACC] WR[0]
U 1F61, 10F6.95 ;20480          MISC2 [TRAP.ACC] WR[1]
U 1F62, 10F8.15 ;20481          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F63, 3A18.15 ;20482          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F64, 3618.15 ;20483          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F65, B716.95 ;20484          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F66, 0869.3C ;20485          JSR [CHECK.RESULT]            ;check for error
U 1F67, 09F5.54 ;20486          JMP [T38.8]                   ;Loop on error
U 1F68, FF82.15 ;20487          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 1F69, 65A6.15 ;20488  T38.9:  CLR LS[DATA.3]                ;Set up data to be passed
U 1F6A, E5A2.15 ;20489          CLR LS[DATA.1]
U 1F6B, B67C.15 ;20490          MOV LS[BIT30] TO WR[0]
U 1F6C, 3EA4.15 ;20491          MOV WR[0] TO LS[DATA.2]
U 1F6D, 0878.FC ;20492          JSR [LOAD.TRIPLE]             ;Load the data into ETO and FTO
U 1F6E, 361E.15 ;20493          MOV LS[T15] TO WR[0]          ;Shift left one
U 1F6F, B716.95 ;20494          MOV LS[#300] TO WR[1]
U 1F70, 90F6.15 ;20495          MISC2 [TRAP.ACC] WR[0]
U 1F71, 10F6.95 ;20496          MISC2 [TRAP.ACC] WR[1]
U 1F72, B616.15 ;20497          MOV LS[T11] TO WR[0]          ;Set up MOV FTO TO FTO
U 1F73, 361A.95 ;20498          MOV LS[T13] TO WR[1]          ;Set up branch
U 1F74, 90F6.15 ;20499          MISC2 [TRAP.ACC] WR[0]
U 1F75, 10F6.95 ;20500          MISC2 [TRAP.ACC] WR[1]
U 1F76, 10F8.15 ;20501          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 1F77, 3A18.15 ;20502          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 1F78, 3618.15 ;20503          MOV LS[T12] TO WR[0]          ;Set up received data
U 1F79, B716.95 ;20504          MOV LS[#300] TO WR[1]         ;Set up expected data
U 1F7A, 0869.3C ;20505          JSR [CHECK.RESULT]            ;check for error
U 1F7B, 09F6.94 ;20506          JMP [T38.9]                   ;Loop on error
                ;20507  T38.END:
```

```
;20508  .page
;20509  .TOC    "TEST 39 NULL BRANCH AND CPU RCV DATA BRANCH TEST(M8389 FPA MODULE)"
;20510  ;++
;20511  ;
;20512  ;
;20513  ; Test Description:
;20514  ;
;20515  ;       The purpose of this test is to check the NULL branch condition and
;20516  ;       the CPU RCV DATA branch condition. The data path will be as follows:
;20517  ;       the branch condition will be asserted or in the case of the NULL branch
;20518  ;       a random instruction will be chosen. The branch field of the instruc-
;20519  ;       tion will be 10101 in the case of the NULL branch and 10110 in the case
;20520  ;       of the CPU RCV DATA branch condition. A branch should never occur on
;20521  ;       the NULL branch. CPU RCV DATA will be asserted if the CPU issues a MISC
;20522  ;       instruction with bit 18 clear and the MISC FUNC 1 field set to SET PORT
;20523  ;       I/O MODE. CPU RCV DATA will also cause ACC SYNC to be asserted if the
;20524  ;       instruction encode bits are either SUB OR CMP. ACC SYNC is an input to
;20525  ;       the CPU, and is a skip condition.
;20526  ;
;20527  ; Assumptions:
;20528  ;
;20529  ;       It is assumed that the branch control logic has been tested previously
;20530  ;       and works.
;20531  ;
;20532  ; Test Steps:
;20533  ;
;20534  ;       1) Issue an OR of two WR in the fraction and exponent data path with
;20535  ;          the branch field set to 10101. If there is any branch at all then
;20536  ;          issue error1.
;20537  ;       2) Issue a MOV instuction from the CPU with the MDP field set to 5.
;20538  ;          This will should cause CPU RCV DATA to be asserted.  At the same
;20539  ;          time an FPA instruction should be issued that's a nop with the
;20540  ;          branch field set to 10110. If there is no branch or an incorrect
;20541  ;          branch then issue error2.
;20542  ;       3) Issue an OR of two WR's in the fraction data path with the branch
;20543  ;          field set to 10110. If there is any branch at all then issue error3.
;20544  ;
;20545  ; Error:
;20546  ;
;20547  ;       Error1 - Branch occured when the branch condition was NULL.
;20548  ;       Error2 - Bracnh failed when CPU RCV DATA was asserted.
;20549  ;       Error3 - Branch failed when CPU RCV DATA was unasserted.
;20550  ;
;20551  ; Debug:
;20552  ;
;20553  ;       Error1: If this error occurs then the BRANCH 1 PAL and the BRANCH 2 PAL
;20554  ;               should be checked for error. If the PAL's are not the cause of
;20555  ;               the error then the UBCTL bits should be checked to be 10100. If
;20556  ;               the UBCTL bits are in error then the CS Latch they come from
;20557  ;               should be checked for error.
;20558  ;       Error2: If this error occurs then the BRANCH 1 PAL and the BRANCH 2 PAL
;20559  ;               should be checked for error. If the PAL's are not the cause of
;20560  ;               the error then the UBCTL bits should be checked to be 10110 and
;20561  ;               CPU RCV DATA should be checkded to be asserted. If the UBCTL
;20562  ;               bits are in error then the CS Latch they come from should be
```

```
                ;20563  ;               checked for error. If CPU RCV DATA is in error then the gates
                ;20564  ;               that CPU RCV DATA come from should be checked for error.
                ;20565  ;       Error3: Same as error2 except that CPU RCV DATA is unasserted.
                ;20566  ;
                ;20567  ;
                ;20568  ;--
                ;20569  T.39:
U 1F7C, B65E,15 ;20570          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;20571                                          ; STATUS WORD
U 1F7D, 3E80,15 ;20572          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;20573                                          ; BIT 15 INDICATES START OF TEST TO
                ;20574                                          ; CONSOLE
U 1F7E, 10E0,15 ;20575          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;20576  WAIT.T39.0:
U 1F7F, 89F7,F4 ;20577          JMP [WAIT.T39.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;20578                                          ; RESPOND
U 1F80, 887E,EC ;20579          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;20580                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1F81, B782,15 ;20581          MOV LS[T39.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1F82, BE12,15 ;20582          MOV WR[0] TO LS[T9]             ; references.
U 1F83, 0870,8C ;20583          JSR [L1]                        ;Get expected data.
U 1F84, BE16,95 ;20584          MOV WR[1] TO LS[T11]            ;Save expected data
U 1F85, 8871,2C ;20585          JSR [T84]                       ;Set address to be forced.
U 1F86, 3708,15 ;20586  T39.1:  MOV LS[T84] TO WR[0]            ;Load address to be forced into WR
U 1F87, 90F6,15 ;20587          MISC2 [TRAP.ACC] WR[0]          ;Force null branch
U 1F88, 10F8,15 ;20588          MISC2 [READ.ACC.UPC]            ;Enable the NUA lines onto FPA BUS.
U 1F89, 3A18,15 ;20589          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS.
U 1F8A, 3618,15 ;20590          MOV LS[T12] TO WR[0]            ;Put received data in WR[0].
U 1F8B, 3616,95 ;20591          MOV LS[T11] TO WR[1]            ;Setup expected data
U 1F8C, 0869,3C ;20592          JSR [CHECK.RESULT]              ;report an error if there is one
U 1F8D, 89F8,64 ;20593          JMP [T39.1]                     ;Loop on error
U 1F8E, FF82,15 ;20594          INC LS[ERROR.NUMBER]            ;Go on to error 2.
U 1F8F, 0870,8C ;20595          JSR [L1]                        ;Get expected data.
U 1F90, BE16,95 ;20596          MOV WR[1] TO LS[T11]            ;Save expected data
U 1F91, 8871,2C ;20597          JSR [T84]                       ;Get an address to be forced.
U 1F92, 3708,15 ;20598  T39.2:  MOV LS[T84] TO WR[0]            ;load address to be forced in WR[0]
U 1F93, 90F6,15 ;20599          MISC2 [TRAP.ACC] WR[0]          ;Force CPU RCV DATA branch
U 1F94, 3A18,15 ;20600          MOV FPA TO LS[T12]              ;Asserts CPU RCV DATA
U 1F95, 10F8,15 ;20601          MISC2 [READ.ACC.UPC]            ;Enable the NUA onto the FPA.
U 1F96, 3A18,15 ;20602          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS.
U 1F97, 3618,15 ;20603          MOV LS[T12] TO WR[0]            ;Put received data in WR[0]
U 1F98, 3616,95 ;20604          MOV LS[T11] TO WR[1]            ;Setup expected data
U 1F99, 0869,3C ;20605          JSR [CHECK.RESULT]              ;Report an error if there is one.
U 1F9A, 89F9,24 ;20606          JMP [T39.2]                     ;Loop on error
U 1F9B, FF82,15 ;20607          INC LS[ERROR.NUMBER]            ;Go on to error 3.
U 1F9C, 0870,8C ;20608          JSR [L1]                        ;Get expected data.
U 1F9D, BE16,95 ;20609          MOV WR[1] TO LS[T11]            ;Save expected data
U 1F9E, 3708,15 ;20610  T39.3:  MOV LS[T84] TO WR[0]            ;load address to be forced in WR[0]
U 1F9F, 90F6,15 ;20611          MISC2 [TRAP.ACC] WR[0]          ;Force CPU RCV DATA branch
U 1FA0, 10F8,15 ;20612          MISC2 [READ.ACC.UPC]            ;Enable the NUA onto the FPA.
U 1FA1, 3A18,15 ;20613          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS.
U 1FA2, 3618,15 ;20614          MOV LS[T12] TO WR[0]            ;Put received data in WR[0]
U 1FA3, 3616,95 ;20615          MOV LS[T11] TO WR[1]            ;Setup expected data
U 1FA4, 0869,3C ;20616          JSR [CHECK.RESULT]              ;Report an error if there is one.
U 1FA5, 89F9,E4 ;20617          JMP [T39.3]                     ;Loop on error
```

```
                ;20618  T39.END:
```

```
;20619  .page
;20620  .TOC    "TEST 3A FRAC<55:7>=0 BRANCH TEST(M8389 FPA MODULE)"
;20621  ;++
;20622  ;
;20623  ;
;20624  ; Test Description:
;20625  ;
;20626  ;       The purpose of this test is to check the FRAC<55:7>=0 branch condition.
;20627  ;       The data path will be as follows: the branch condition will be asserted
;20628  ;       with the branch field set to 10111. If there is no branch or an incor-
;20629  ;       rect branch then an error statement will be issued. This procedure will
;20630  ;       be repeated for the branch condition unasserted. FRAC<55:7>=0 will be
;20631  ;       asserted if FRAC<55:48>=0, FRAC<47:32>=0, FRAC<31:16>=0, FRAC<15:00>=0,
;20632  ;       and EXT<7:0>=0 are asserted.
;20633  ;
;20634  ; Assumptions:
;20635  ;
;20636  ;       It is assumed that the branch control logic has been tested previously
;20637  ;       and works.
;20638  ;
;20639  ; Test Steps:
;20640  ;
;20641  ;       1) Load a WR of the fraction data path and the extension data path with
;20642  ;          zeros. Issue an OR of the WR with 0 and set the branch field to
;20643  ;          10111. If there is no branch or an incorrect branch then issue
;20644  ;          error1.
;20645  ;       2) Load FTO with zeros. The hidden bit will be set causing the branch
;20646  ;          condition to be unasserted.
;20647  ;       3) Load FTO with bit 30 set. Shift left to shift the hidden bit out.
;20648  ;          Then branch on FRAC<55-7>. If there is a branch then issue error
;20649  ;          3.
;20650  ;       4) Repeat step 3 with bit 14 set in the second float.
;20651  ;       5) Repeat step 3 with bit 30 set in the second float.
;20652  ;       6) Repeat step 3 with bit 10 set in the third float.
;20653  ;       7) Repeat step 3 with bit 14 set in the third float.
;20654  ;
;20655  ; Error:
;20656  ;
;20657  ;       Error1 - Branch failed when FRAC<55:7>=0 was asserted.
;20658  ;       Error2 - Branch failed when FRAC<55:7>=0 was unasserted.
;20659  ;       Error3 - Branch failed when FRAC<55:7>=0 was unasserted.
;20660  ;       Error4 - Branch failed when FRAC<55:7>=0 was unasserted.
;20661  ;       Error5 - Branch failed when FRAC<55:7>=0 was unasserted.
;20662  ;       Error6 - Branch failed when FRAC<55:7>=0 was unasserted.
;20663  ;       Error7 - Branch failed when FRAC<55:7>=0 was unasserted.
;20664  ;
;20665  ; Debug:
;20666  ;
;20667  ;       Error1: If this error occurs then the BRANCH 1 PAL and the BRANCH 2 PAL
;20668  ;               should be checked for error. If the PAL's are not the cause of
;20669  ;               the error then the UBCTL bits should be checked to be 10111 and
;20670  ;               FRAC<55:48>=0, FRAC<47:32>=0, FRAC<31:16>=0, FRAC<15:00>=0, and
;20671  ;               EXT<7:0>=0 should be checked to be asserted. If the UBCTL bits
;20672  ;               are in error then the Control Store Latch that they come from
;20673  ;               should be checked for error. If any of the other signals
```

```
                    ;20674  ;               mentioned are in error then the STATUS REG 2 LATCH should be
                    ;20675  ;               checked for error. If the latch isn't the cause of the error
                    ;20676  ;               then the sign bits of the 2901's should be checked for error.
                    ;20677  ;       Error2: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20678  ;       Error3: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20679  ;       Error4: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20680  ;       Error5: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20681  ;       Error6: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20682  ;       Error7: Same as error1 except that FRAC<55:7>=0 is unasserted.
                    ;20683  ;
                    ;20684  ;
                    ;20685  ;--
                    ;20686  T.3A:
U 1FA6, B65E,15 ;20687          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;20688                                      ; STATUS WORD
U 1FA7, 3E80,15 ;20689          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;20690                                      ; BIT 15 INDICATES START OF TEST TO
                    ;20691                                      ; CONSOLE
U 1FA8, 10E0,15 ;20692          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;20693  WAIT.T3A.0:
U 1FA9, 09FA,94 ;20694          JMP [WAIT.T3A.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;20695                                      ; RESPOND
U 1FAA, 887E,EC ;20696          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;20697                                      ; AND SIZE BITS. CHECK LOWER 10 BITS
U 1FAB, B784,15 ;20698          MOV LS[T3A.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 1FAC, BE12,15 ;20699          MOV WR[0] TO LS[T9]              ; references.
U 1FAD, B72E,15 ;20700          MOV LS[#2A7] TO WR[0]           ;Set up address for load data in T10
U 1FAE, BE14,15 ;20701          MOV WR[0] TO LS[T10]
U 1FAF, 8870,3C ;20702          JSR [L0]                        ;Get branch address
U 1FB0, 3E1A,15 ;20703          MOV WR[0] TO LS[T13]
U 1FB1, 8870,3C ;20704          JSR [L0]                        ;Get MOV FT0 TO FT0 address
U 1FB2, 3E16,15 ;20705          MOV WR[0] TO LS[T11]
U 1FB3, 8870,3C ;20706          JSR [L0]                        ;Set up CLR FT0
U 1FB4, 3E1C,15 ;20707          MOV WR[0] TO LS[T14]
U 1FB5, 8870,3C ;20708          JSR [L0]                        ;Set up SHIFT LEFT FWR[FT0] IN[ZERO]
U 1FB6, BE1E,15 ;20709          MOV WR[0] TO LS[T15]
U 1FB7, B61C,15 ;20710  T3A.1:  MOV LS[T14] TO WR[0]            ;Set up CLR FT0
U 1FB8, 361A,95 ;20711          MOV LS[T13] TO WR[1]            ;Set up branch
U 1FB9, 90F6,15 ;20712          MISC2 [TRAP.ACC] WR[0]
U 1FBA, 10F6,95 ;20713          MISC2 [TRAP.ACC] WR[1]
U 1FBB, 10F8,15 ;20714          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1FBC, 3A18,15 ;20715          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1FBD, 3618,15 ;20716          MOV LS[T12] TO WR[0]            ;Set up received data
U 1FBE, B740,95 ;20717          MOV LS[#302] TO WR[1]           ;Set up expected data
U 1FBF, 0869,3C ;20718          JSR [CHECK.RESULT]              ;check for error
U 1FC0, 09FB,74 ;20719          JMP [T3A.1]                     ;Loop on error
U 1FC1, FF82,15 ;20720          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 1FC2, 087C,1C ;20721  T3A.2:  JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 1FC3, B616,15 ;20722          MOV LS[T11] TO WR[0]            ;Set up MOV FT0 TO FT0
U 1FC4, 361A,95 ;20723          MOV LS[T13] TO WR[1]            ;Set up branch
U 1FC5, 90F6,15 ;20724          MISC2 [TRAP.ACC] WR[0]
U 1FC6, 10F6,95 ;20725          MISC2 [TRAP.ACC] WR[1]
U 1FC7, 10F8,15 ;20726          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 1FC8, 3A18,15 ;20727          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 1FC9, 3618,15 ;20728          MOV LS[T12] TO WR[0]            ;Set up received data
```

```
U 1FCA, B716,95 ;20729          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1FCB, 0869,3C ;20730          JSR [CHECK.RESULT]             ;check for error
U 1FCC, 89FC,24 ;20731          JMP [T3A.2]                    ;Loop on error
U 1FCD, FF82,15 ;20732          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1FCE, 65A6,15 ;20733  T3A.3:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1FCF, E5A4,15 ;20734          CLR LS[DATA.2]
U 1FD0, B67C,15 ;20735          MOV LS[BIT30] TO WR[0]
U 1FD1, 3EA2,15 ;20736          MOV WR[0] TO LS[DATA.1]
U 1FD2, 0878,FC ;20737          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1FD3, 361E,15 ;20738          MOV LS[T15] TO WR[0]           ;Shift left one
U 1FD4, B716,95 ;20739          MOV LS[#300] TO WR[1]
U 1FD5, 90F6,15 ;20740          MISC2 [TRAP.ACC] WR[0]
U 1FD6, 10F6,95 ;20741          MISC2 [TRAP.ACC] WR[1]
U 1FD7, B616,15 ;20742          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 1FD8, 361A,95 ;20743          MOV LS[T13] TO WR[1]           ;Set up branch
U 1FD9, 90F6,15 ;20744          MISC2 [TRAP.ACC] WR[0]
U 1FDA, 10F6,95 ;20745          MISC2 [TRAP.ACC] WR[1]
U 1FDB, 10F8,15 ;20746          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1FDC, 3A18,15 ;20747          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1FDD, 3618,15 ;20748          MOV LS[T12] TO WR[0]           ;Set up received data
U 1FDE, B716,95 ;20749          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1FDF, 0869,3C ;20750          JSR [CHECK.RESULT]             ;check for error
U 1FE0, 89FC,E4 ;20751          JMP [T3A.3]                    ;Loop on error
U 1FE1, FF82,15 ;20752          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1FE2, 65A6,15 ;20753  T3A.4:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1FE3, E5A2,15 ;20754          CLR LS[DATA.1]
U 1FE4, 365C,15 ;20755          MOV LS[BIT14] TO WR[0]
U 1FE5, 3EA4,15 ;20756          MOV WR[0] TO LS[DATA.2]
U 1FE6, 0878,FC ;20757          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1FE7, 361E,15 ;20758          MOV LS[T15] TO WR[0]           ;Shift left one
U 1FE8, B716,95 ;20759          MOV LS[#300] TO WR[1]
U 1FE9, 90F6,15 ;20760          MISC2 [TRAP.ACC] WR[0]
U 1FEA, 10F6,95 ;20761          MISC2 [TRAP.ACC] WR[1]
U 1FEB, B616,15 ;20762          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 1FEC, 361A,95 ;20763          MOV LS[T13] TO WR[1]           ;Set up branch
U 1FED, 90F6,15 ;20764          MISC2 [TRAP.ACC] WR[0]
U 1FEE, 10F6,95 ;20765          MISC2 [TRAP.ACC] WR[1]
U 1FEF, 10F8,15 ;20766          MISC2 [READ.ACC.UPC]           ;Enable the NUA BUS onto the FPA BUS
U 1FF0, 3A18,15 ;20767          MOV FPA TO LS[T12]             ;Enable the FPA BUS onto the Y BUS
U 1FF1, 3618,15 ;20768          MOV LS[T12] TO WR[0]           ;Set up received data
U 1FF2, B716,95 ;20769          MOV LS[#300] TO WR[1]          ;Set up expected data
U 1FF3, 0869,3C ;20770          JSR [CHECK.RESULT]             ;check for error
U 1FF4, 09FE,24 ;20771          JMP [T3A.4]                    ;Loop on error
U 1FF5, FF82,15 ;20772          INC LS[ERROR.NUMBER]           ;Go onto the next error number
U 1FF6, 65A6,15 ;20773  T3A.5:  CLR LS[DATA.3]                 ;Set up data to be passed
U 1FF7, E5A2,15 ;20774          CLR LS[DATA.1]
U 1FF8, B67C,15 ;20775          MOV LS[BIT30] TO WR[0]
U 1FF9, 3EA4,15 ;20776          MOV WR[0] TO LS[DATA.2]
U 1FFA, 0878,FC ;20777          JSR [LOAD.TRIPLE]              ;Load the data into ET0 and FT0
U 1FFB, 361E,15 ;20778          MOV LS[T15] TO WR[0]           ;Shift left one
U 1FFC, B716,95 ;20779          MOV LS[#300] TO WR[1]
U 1FFD, 90F6,15 ;20780          MISC2 [TRAP.ACC] WR[0]
U 1FFE, 10F6,95 ;20781          MISC2 [TRAP.ACC] WR[1]
U 1FFF, B616,15 ;20782          MOV LS[T11] TO WR[0]           ;Set up MOV FT0 TO FT0
U 2000, 361A,95 ;20783          MOV LS[T13] TO WR[1]           ;Set up branch
```

```
U 2001, 90F6,15 ;20784          MISC2 [TRAP.ACC] WR[0]
U 2002, 10F6,95 ;20785          MISC2 [TRAP.ACC] WR[1]
U 2003, 10F8,15 ;20786          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 2004, 3A18,15 ;20787          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 2005, 3618,15 ;20788          MOV LS[T12] TO WR[0]          ;Set up received data
U 2006, B716,95 ;20789          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2007, 0869,3C ;20790          JSR [CHECK.RESULT]            ;check for error
U 2008, 09FF,64 ;20791          JMP [T3A.5]                   ;Loop on error
U 2009, FF82,15 ;20792          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 200A, E5A2,15 ;20793  T3A.6:  CLR LS[DATA.1]                ;Set up data to be passed
U 200B, E5A4,15 ;20794          CLR LS[DATA.2]
U 200C, B654,15 ;20795          MOV LS[BIT10] TO WR[0]
U 200D, BEA6,15 ;20796          MOV WR[0] TO LS[DATA.3]
U 200E, 0878,FC ;20797          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 200F, 361E,15 ;20798          MOV LS[T15] TO WR[0]          ;Shift left one
U 2010, B716,95 ;20799          MOV LS[#300] TO WR[1]
U 2011, 90F6,15 ;20800          MISC2 [TRAP.ACC] WR[0]
U 2012, 10F6,95 ;20801          MISC2 [TRAP.ACC] WR[1]
U 2013, B616,15 ;20802          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 2014, 361A,95 ;20803          MOV LS[T13] TO WR[1]          ;Set up branch
U 2015, 90F6,15 ;20804          MISC2 [TRAP.ACC] WR[0]
U 2016, 10F6,95 ;20805          MISC2 [TRAP.ACC] WR[1]
U 2017, 10F8,15 ;20806          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 2018, 3A18,15 ;20807          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 2019, 3618,15 ;20808          MOV LS[T12] TO WR[0]          ;Set up received data
U 201A, B716,95 ;20809          MOV LS[#300] TO WR[1]         ;Set up expected data
U 201B, 0869,3C ;20810          JSR [CHECK.RESULT]            ;check for error
U 201C, 0A00,A4 ;20811          JMP [T3A.6]                   ;Loop on error
U 201D, FF82,15 ;20812          INC LS[ERROR.NUMBER]          ;Go onto the next error number
U 201E, E5A2,15 ;20813  T3A.7:  CLR LS[DATA.1]                ;Set up data to be passed
U 201F, E5A4,15 ;20814          CLR LS[DATA.2]
U 2020, 365C,15 ;20815          MOV LS[BIT14] TO WR[0]
U 2021, BEA6,15 ;20816          MOV WR[0] TO LS[DATA.3]
U 2022, 0878,FC ;20817          JSR [LOAD.TRIPLE]             ;Load the data into ET0 and FT0
U 2023, 361E,15 ;20818          MOV LS[T15] TO WR[0]          ;Shift left one
U 2024, B716,95 ;20819          MOV LS[#300] TO WR[1]
U 2025, 90F6,15 ;20820          MISC2 [TRAP.ACC] WR[0]
U 2026, 10F6,95 ;20821          MISC2 [TRAP.ACC] WR[1]
U 2027, B616,15 ;20822          MOV LS[T11] TO WR[0]          ;Set up MOV FT0 TO FT0
U 2028, 361A,95 ;20823          MOV LS[T13] TO WR[1]          ;Set up branch
U 2029, 90F6,15 ;20824          MISC2 [TRAP.ACC] WR[0]
U 202A, 10F6,95 ;20825          MISC2 [TRAP.ACC] WR[1]
U 202B, 10F8,15 ;20826          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 202C, 3A18,15 ;20827          MOV FPA TO LS[T12]            ;Enable the FPA BUS onto the Y BUS
U 202D, 3618,15 ;20828          MOV LS[T12] TO WR[0]          ;Set up received data
U 202E, B716,95 ;20829          MOV LS[#300] TO WR[1]         ;Set up expected data
U 202F, 0869,3C ;20830          JSR [CHECK.RESULT]            ;check for error
U 2030, 0A01,E4 ;20831          JMP [T3A.7]                   ;Loop on error
                ;20832  T3A.END:
```

```
;20833  .page
;20834  .TOC    "TEST 3B EXPONENT=0 AND EXP15 F3 BRANCH TEST(M8389 FPA MODULE)"
;20835  ;++
;20836  ;
;20837  ;
;20838  ; Test Description:
;20839  ;
;20840  ;       The purpose of this test is to check the EXPONENT=0 and EXP15 F3 branch
;20841  ;       conditions. The data path will be as follows: the branch conditions
;20842  ;       will be asserted with the branch field set to 11000. If there is no
;20843  ;       branch or an incorrect branch then an error statement will be issued.
;20844  ;       This procedure will be repeated for the branch condiitons unasserted.
;20845  ;       EXPONENT=0 will be asserted if exponent data path has zeros in it.
;20846  ;       EXP15 F3 will be asserted if the MSB bit of the exponent data path
;20847  ;       is set.
;20848  ;
;20849  ; Assumptions:
;20850  ;
;20851  ;       It is assumed that the branch control logic has been tested previously
;20852  ;       and works.
;20853  ;
;20854  ; Test Steps:
;20855  ;
;20856  ;       1) Issue a MISC instruction to load the exponent data path with zeros.
;20857  ;          AND the the WR in the exponent data path with zero. Issue a nop
;20858  ;          with the branch field set to 11000. If there is no branch or an
;20859  ;          incorrect branch then issue error 1.
;20860  ;       2) Issue MISC instruction to load a WR of the exponent data path with
;20861  ;          zero. Issue an OR of the WR with zero, then issue nop with the
;20862  ;          branch field set to 11000. If there is no branch or an incorrect
;20863  ;          branch then issue error 2.
;20864  ;
;20865  ; Error:
;20866  ;
;20867  ;       Error1 - Branch failed when EXPONENT=0 is asserted and EXP15 F3 SV is
;20868  ;                unasserted.
;20869  ;       Error2 - Branch failed when EXPONENT=0 is unasserted and EXP15 F3 SV
;20870  ;                is asserted.
;20871  ;
;20872  ; Debug:
;20873  ;
;20874  ;       Error1: If this error occurs then the BRANCH 3 PAL should be checked
;20875  ;               for error. If the PAL isn't the cause of the error then the
;20876  ;               UBCTL bits should be checked to be 11000 and EXP EQ 0 should
;20877  ;               be asserted and EXP 15 F3 SV should be unasserted. If EXP EQ 0
;20878  ;               is in error then the sign bit of the 2901's in the exponent
;20879  ;               data path should be checked for error. If EXP15 F3 SV is in
;20880  ;               error then the BRANCH 3 PAL should be checked for error. If
;20881  ;               the PAL is not the cause of the error then EXP15 F3 should be
;20882  ;               checked to be unasserted. If EXP15 F3 is not unasserted then
;20883  ;               the MSB bit of the 2901's in the exponent data path should be
;20884  ;               checked for error.
;20885  ;       Error2: Same as error1 except that EXPONENT=0 is unasserted and EXP15
;20886  ;               F3 SV is asserted.
;20887  ;
```

```
                ;20888  ;
                ;20889  ;--
                ;20890  T.3B:
U 2031, B65E,15 ;20891          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;20892                                          ; STATUS WORD
U 2032, 3E80,15 ;20893          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;20894                                          ; BIT 15 INDICATES START OF TEST TO
                ;20895                                          ; CONSOLE
U 2033, 10E0,15 ;20896          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;20897  WAIT.T3B.0:
U 2034, 8A03,44 ;20898          JMP [WAIT.T3B.C]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;20899                                          ; RESPOND
U 2035, 887E,EC ;20900          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;20901                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 2036, 3786,15 ;20902          MOV LS[T3B.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2037, BE12,15 ;20903          MOV WR[0] TO LS[T9]             ; references.
U 2038, B72E,15 ;20904          MOV LS[#2A7] TO WR[0]           ;Set up load address
U 2039, BE14,15 ;20905          MOV WR[0] TO LS[T10]
U 203A, 8870,3C ;20906          JSR [L0]                        ;Get address of MOV ET0 TO ET0
U 203B, 3E1C,15 ;20907          MOV WR[0] TO LS[T14]
U 203C, 8870,3C ;20908          JSR [L0]                        ;Get address of SHIFT ET0
U 203D, 3E16,15 ;20909          MOV WR[0] TO LS[T11]
U 203E, 8870,3C ;20910          JSR [L0]                        ;Branch on EXP EQ 0 and EXP F3
U 203F, 3E1A,15 ;20911          MOV WR[0] TO LS[T13]
U 2040, E5A2,15 ;20912  T3B.1:  CLR LS[DATA.1]
U 2041, 8877,DC ;20913          JSR [LOAD.DATA]                 ;Load data into ET0
U 2042, B61C,15 ;20914          MOV LS[T14] TO WR[0]
U 2043, 361A,95 ;20915          MOV LS[T13] TO WR[1]
U 2044, 90F6,15 ;20916          MISC2 [TRAP.ACC] WR[0]          ;MOV ET0 TO ET0
U 2045, 10F6,95 ;20917          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2046, 10F8,15 ;20918          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2047, 3A18,15 ;20919          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2048, 3618,15 ;20920          MOV LS[T12] TO WR[0]            ;Set up received data
U 2049, B740,95 ;20921          MOV LS[#302] TO WR[1]           ;Set up expected data
U 204A, 0869,3C ;20922          JSR [CHECK.RESULT]              ;check for error
U 204B, 8A04,04 ;20923          JMP [T3B.1]                     ;Loop on error
U 204C, FF82,15 ;20924          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 204D, 365C,15 ;20925  T3B.2:  MOV LS[BIT14] TO WR[0]          ;Set up data
U 204E, 3EA2,15 ;20926          MOV WR[0] TO LS[DATA.1]
U 204F, 8877,DC ;20927          JSR [LOAD.DATA]                 ;Load data into ET0
U 2050, 0876,2C ;20928          JSR [SHIFT.LEFT.8]              ;Shift the data left 8
U 2051, B61C,15 ;20929          MOV LS[T14] TO WR[0]
U 2052, 361A,95 ;20930          MOV LS[T13] TO WR[1]
U 2053, 90F6,15 ;20931          MISC2 [TRAP.ACC] WR[0]          ;MOV ET0 TO ET0
U 2054, 10F6,95 ;20932          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2055, 10F8,15 ;20933          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2056, 3A18,15 ;20934          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2057, 3618,15 ;20935          MOV LS[T12] TO WR[0]            ;Set up received data
U 2058, B73E,95 ;20936          MOV LS[#301] TO WR[1]           ;Set up expected data
U 2059, 0869,3C ;20937          JSR [CHECK.RESULT]              ;check for error
U 205A, 0A04,D4 ;20938          JMP [T3B.2]                     ;Loop on error
                ;20939  T3B.END:
```

```
;20940  .page
;20941  .TOC    "TEST 3C OP1=0 AND OP2=0 BRANCH TEST(M8389 FPA MODULE)"
;20942  ;++
;20943  ;
;20944  ;
;20945  ; Test Description:
;20946  ;
;20947  ;       The purpose is to check the OP1=0 and OP2=0 branch conditions. The data
;20948  ;       path is the following: the branch conditions will be asserted with the
;20949  ;       branch field set to 11001. If there is no branch or an incorrect branch
;20950  ;       then an error statement will be issued. This procedure will be repeated
;20951  ;       for the branch conditions unasserted. OP1=0 will be asserted if the
;20952  ;       exponent of the first operand equals zero. OP2=0 will be asserted if
;20953  ;       the exponent of the second operand equals zero.
;20954  ;
;20955  ; Assumptions:
;20956  ;
;20957  ;       It is assumed that the branch control logic has been tested previously
;20958  ;       and works.
;20959  ;
;20960  ; Test Steps:
;20961  ;
;20962  ;       1) Assert OP2=0 and OP1=0 by loading zeros into the exponent data path
;20963  ;          and clocking operand eq 0 twice, once to assert OP2=0 and once to
;20964  ;          assert OP1=0. On the cycle following the assertion of OP1=0 branch.
;20965  ;       2) This error will check to see that OP1=0 and OP2=0 will stay asserted
;20966  ;          if CLOCK OPERAND EQ 0 is not asserted and the exponent doesn't
;20967  ;          change.
;20968  ;       3) Load the Exponent data path with ones. execute a CLOCK OPERAND EQ 0
;20969  ;          This will cause OP2=0 to be unasserted but OP1=0 should stay
;20970  ;          asserted.
;20971  ;       4) CLOCK OPERAND EQ 0. This should cause OP1=0 to become unasserted.
;20972  ;          Then branch.
;20973  ;       5) This checks if OP2=0 and OP1=0 are unasserted when a NOP is excuted.
;20974  ;
;20975  ; Error:
;20976  ;
;20977  ;       Error1 - Branch failed when both OP1=0 and OP2=0 were asserted.
;20978  ;       Error2 - Branch failed when both OP1=0 and OP2=0 were asserted.
;20979  ;       Error3 - Branch failed when OP1=0 was asserted and OP2=0 was unasserted
;20980  ;       Error4 - Branch failed when OP1=0 was unasserted and OP2=0 was
;20981  ;                unasserted.
;20982  ;       Error5 - Branch failed when both OP1=0 and OP2=0 were unasserted.
;20983  ;
;20984  ; Debug:
;20985  ;
;20986  ;       Error1: If this error occurs then the BRANCH 3 PAL should be checked
;20987  ;               for error. If the PAL is not the cause of the error then OP1
;20988  ;               and OP2 should be checked to be asserted. If either of these
;20989  ;               signals are in error then the BRANCH 3 PAL should be checked
;20990  ;               for error. If the PAL is not in error then EXP=0 and OP2=0
;20991  ;               should be checked to be asserted on the precvious cycle.
;20992  ;       Error2: Same as error 1.
;20993  ;       Error3: Same as error 1 except that OP2=0 is unasserted.
;20994  ;       Error4: Same as error 1 except that both OP2=0 and OP1=0 are
```

```
                ;20995  ;               unasserted.
                ;20996  ;       Error5: Same as error 1 except that both OP1=0 and OP2=0 are
                ;20997  ;               unasserted.
                ;20998  ;
                ;20999  ;
                ;21000  ;--
                ;21001  T.3C:
U 205B, B65E,15 ;21002          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;21003                                          ; STATUS WORD
U 205C, 3E80,15 ;21004          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;21005                                          ; BIT 15 INDICATES START OF TEST TO
                ;21006                                          ; CONSOLE
U 205D, 10E0,15 ;21007          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;21008  WAIT.T3C.0:
U 205E, 8A05,E4 ;21009          JMP [WAIT.T3C.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;21010                                          ; RESPOND
U 205F, 887E,EC ;21011          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;21012                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 2060, B788,15 ;21013          MOV LS[T3C.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2061, BE12,15 ;21014          MOV WR[0] TO LS[T9]             ; references.
U 2062, B72E,15 ;21015          MOV LS[#2A7] TO WR[0]           ;Get LOAD address
U 2063, BE14,15 ;21016          MOV WR[0] TO LS[T10]
U 2064, 8870,3C ;21017          JSR [LO]                        ;Get MOV ETO TO EQ address
U 2065, 3E1C,15 ;21018          MOV WR[0] TO LS[T14]
U 2066, 8870,3C ;21019          JSR [LO]                        ;Get MOV EQ TO ETO and CLOCK OP1=0/OP2=0
U 2067, BE1E,15 ;21020          MOV WR[0] TO LS[T15]            ; address
U 2068, 8870,3C ;21021          JSR [LO]                        ;Get Branch address
U 2069, 3E1A,15 ;21022          MOV WR[0] TO LS[T13]
U 206A, E5A2,15 ;21023  T3C.1:  CLR LS[DATA.1]                  ;Set data to zero
U 206B, 8877,DC ;21024          JSR [LOAD.DATA]                 ;Load the data to ETO
U 206C, 0878,AC ;21025          JSR [MOV.DATA]                  ;Move the data from ETO to EQ
U 206D, 361E,15 ;21026          MOV LS[T15] TO WR[0]            ;Move the data from EQ to ETO
U 206E, B716,95 ;21027          MOV LS[#300] TO WR[1]           ; and set OP2=0
U 206F, 90F6,15 ;21028          MISC2 [TRAP.ACC] WR[0]
U 2070, 10F6,95 ;21029          MISC2 [TRAP.ACC] WR[1]
U 2071, 361E,15 ;21030          MOV LS[T15] TO WR[0]            ;MOV the data from EQ TO ETO and set
U 2072, 361A,95 ;21031          MOV LS[T13] TO WR[1]            ; OP1=0 and branch
U 2073, 90F6,15 ;21032          MISC2 [TRAP.ACC] WR[0]
U 2074, 10F6,95 ;21033          MISC2 [TRAP.ACC] WR[1]
U 2075, 10F8,15 ;21034          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2076, 3A18,15 ;21035          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2077, 3618,15 ;21036          MOV LS[T12] TO WR[0]            ;Set up received data
U 2078, 3742,95 ;21037          MOV LS[#303] TO WR[1]           ;Set up expected data
U 2079, 0869,3C ;21038          JSR [CHECK.RESULT]              ;check for error
U 207A, 0A06,A4 ;21039          JMP [T3C.1]                     ;Loop on error
U 207B, FF82,15 ;21040          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 207C, B61A,15 ;21041  T3C.2:  MOV LS[T13] TO WR[0]            ;Branch
U 207D, 90F6,15 ;21042          MISC2 [TRAP.ACC] WR[0]
U 207E, 10F8,15 ;21043          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 207F, 3A18,15 ;21044          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2080, 3618,15 ;21045          MOV LS[T12] TO WR[0]            ;Set up received data
U 2081, 3742,95 ;21046          MOV LS[#303] TO WR[1]           ;Set up expected data
U 2082, 0869,3C ;21047          JSR [CHECK.RESULT]              ;check for error
U 2083, 8A07,C4 ;21048          JMP [T3C.2]                     ;Loop on error
U 2084, FF82,15 ;21049          INC LS[ERROR.NUMBER]            ;Go onto the next error number
```

```
U 2085, E5A2.15 ;21050  T3C.3:  CLR LS[DATA.1]                  ;Set data to zero
U 2086, 8877.DC ;21051          JSR [LOAD.DATA]                 ;Load the data to ET0
U 2087, 0878.AC ;21052          JSR [MOV.DATA]                  ;Move the data from ET0 to EQ
U 2088, 361E.15 ;21053          MOV LS[T15] TO WR[0]            ;Move the data from EQ to ET0
U 2089, B716.95 ;21054          MOV LS[#300] TO WR[1]           ; and set OP2=0
U 208A, 90F6.15 ;21055          MISC2 [TRAP.ACC] WR[0]
U 208B, 10F6.95 ;21056          MISC2 [TRAP.ACC] WR[1]
U 208C, 361E.15 ;21057          MOV LS[T15] TO WR[0]            ;MOV the data from EQ TO ET0 and set
U 208D, 90F6.15 ;21058          MISC2 [TRAP.ACC] WR[0]          ; OP1=0
U 208E, 10F6.95 ;21059          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 208F, B69E.15 ;21060          MOV LS[#FFFFFFFF] TO WR[0]      ;Get data to be loaded
U 2090, 3EA2.15 ;21061          MOV WR[0] TO LS[DATA.1]
U 2091, 8877.DC ;21062          JSR [LOAD.DATA]                 ;Load the data into ET0
U 2092, 0878.AC ;21063          JSR [MOV.DATA]                  ;Move the contents of ET0 to EQ
U 2093, 361E.15 ;21064          MOV LS[T15] TO WR[0]            ;MOV EQ TO ET0 AND CLOCK OP2=0
U 2094, 361A.95 ;21065          MOV LS[T13] TO WR[1]            ;Branch
U 2095, 90F6.15 ;21066          MISC2 [TRAP.ACC] WR[0]
U 2096, 10F6.95 ;21067          MISC2 [TRAP.ACC] WR[1]
U 2097, 10F8.15 ;21068          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2098, 3A18.15 ;21069          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2099, 3618.15 ;21070          MOV LS[T12] TO WR[0]            ;Set up received data
U 209A, B740.95 ;21071          MOV LS[#302] TO WR[1]           ;Set up expected data
U 209B, 0869.3C ;21072          JSR [CHECK.RESULT]              ;check for error
U 209C, 8A08.54 ;21073          JMP [T3C.3]                     ;Loop on error
U 209D, FF82.15 ;21074          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 209E, 361E.15 ;21075  T3C.4:  MOV LS[T15] TO WR[0]            ;MOV EQ TO ET0 AND CLOCK OP1=0
U 209F, 361A.95 ;21076          MOV LS[T13] TO WR[1]            ;Branch
U 20A0, 90F6.15 ;21077          MISC2 [TRAP.ACC] WR[0]
U 20A1, 10F6.95 ;21078          MISC2 [TRAP.ACC] WR[1]
U 20A2, 10F8.15 ;21079          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 20A3, 3A18.15 ;21080          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 20A4, 3618.15 ;21081          MOV LS[T12] TO WR[0]            ;Set up received data
U 20A5, B716.95 ;21082          MOV LS[#300] TO WR[1]           ;Set up expected data
U 20A6, 0869.3C ;21083          JSR [CHECK.RESULT]              ;check for error
U 20A7, 8A09.E4 ;21084          JMP [T3C.4]                     ;Loop on error
U 20A8, FF82.15 ;21085          INC LS[ERROR.NUMBER]            ;Go onto the next error number
U 20A9, B61A.15 ;21086  T3C.5:  MOV LS[T13] TO WR[0]            ;Branch
U 20AA, 90F6.15 ;21087          MISC2 [TRAP.ACC] WR[0]
U 20AB, 10F8.15 ;21088          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 20AC, 3A18.15 ;21089          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 20AD, 3618.15 ;21090          MOV LS[T12] TO WR[0]            ;Set up received data
U 20AE, B716.95 ;21091          MOV LS[#300] TO WR[1]           ;Set up expected data
U 20AF, 0869.3C ;21092          JSR [CHECK.RESULT]              ;check for error
U 20B0, 0A0A.94 ;21093          JMP [T3C.5]                     ;Loop on error
                ;21094  T3C.END:
```

```
;21095  .page
;21096  .TOC    "TEST 3C CALL SUBROUTINE BRANCH TEST(M8389 FPA MODULE)"
;21097  ;++
;21098  ;
;21099  ;
;21100  ; Test Description:
;21101  ;
;21102  ;       The purpose of this test is to check the subroutine branch. The data
;21103  ;       path will be as follows: the conditions for a jump to subroutine will
;21104  ;       be created with the branch field set to 11010. If there is no jump to
;21105  ;       subroutine then an error statement will be issued. In order to insure
;21106  ;       that a it was a subroutine jump and not just a branch a return from
;21107  ;       subroutine. A jump to subroutine will occur if the branch field is
;21108  ;       set to 11010. A return will occur if the branch field is set to 11110.
;21109  ;
;21110  ; Assumptions:
;21111  ;
;21112  ;       It is assumed that the branch control logic has been tested previously
;21113  ;       and works.
;21114  ;
;21115  ; Test Steps:
;21116  ;
;21117  ;       1) Force a Call to subroutine. Read the UPC to see if it jumped to the
;21118  ;          right place. Check for error.
;21119  ;       2) Force a return from subroutine. Read the UPC to see if it returned
;21120  ;          to 1+ the calling address. Check for error.
;21121  ;       3) Force a Call to a subroutine at location 1FF. Force a return from
;21122  ;          the Call. This will test the carries across the three micro sequen-
;21123  ;          cer chips.
;21124  ;       4) Test the four stack locations with a pattern of 212
;21125  ;       5) Test the four stack locations with a pattern of 121
;21126  ;       6) Test the four stack locations with a pattern of 344
;21127  ;       7) Test the four stack locations with a pattern of 88
;21128  ;       8) Push four different locations onto the stack then pop one at a time
;21129  ;          to check that the push pop mechanism of the stacks works for all
;21130  ;          four loactions.
;21131  ;
;21132  ; Error:
;21133  ;
;21134  ;       Error1 - Jump to subroutine failed.
;21135  ;       Error2 - Return from subroutine failed.
;21136  ;       Error3 - The carries in the microseguncer chip failed.
;21137  ;       Error4 - One of the four stack loctions failed when the pattern was 212
;21138  ;       Error5 - One of the four stack loctions failed when the pattern was 212
;21139  ;       Error6 - One of the four stack loctions failed when the pattern was 212
;21140  ;       Error7 - One of the four stack loctions failed when the pattern was 212
;21141  ;       Error8 - One of the four stack loctions failed when the pattern was 121
;21142  ;       Error9 - One of the four stack loctions failed when the pattern was 121
;21143  ;       ErrorA - One of the four stack loctions failed when the pattern was 121
;21144  ;       ErrorB - One of the four stack loctions failed when the pattern was 121
;21145  ;       ErrorC - One of the four stack loctions failed when the pattern was 344
;21146  ;       ErrorD - One of the four stack loctions failed when the pattern was 344
;21147  ;       ErrorE - One of the four stack loctions failed when the pattern was 344
;21148  ;       ErrorF - One of the four stack loctions failed when the pattern was 344
;21149  ;       Error10 - One of the four stack loctions failed when the pattern was 88
```

```
                ;21150  ;       Error11 - One of the four stack loctions failed when the pattern was 88
                ;21151  ;       Error12 - One of the four stack loctions failed when the pattern was 88
                ;21152  ;       Error13 - One of the four stack loctions failed when the pattern was 88
                ;21153  ;       Error14 - One of the four push-pops failed
                ;21154  ;       Error15 - One of the four push-pops failed
                ;21155  ;       Error16 - One of the four push-pops failed
                ;21156  ;       Error17 - One of the four push-pops failed
                ;21157  ;
                ;21158  ; Debug:
                ;21159  ;
                ;21160  ;       Error1: If this error occurs then the BRANCH 2 PAL should be checked
                ;21161  ;               for error. If the PAL is not the cause of the error then the
                ;21162  ;               UBCTL bits should be checked to be 11010. If the UBCTL bits
                ;21163  ;               are not in error then the file enb bit to the stack control
                ;21164  ;               should be checked to be asserted and the push bit to the stack
                ;21165  ;               control should be checked to be unasserted. The select bits
                ;21166  ;               should be checked to be 01. The register enable should be
                ;21167  ;               checked to be asserted.
                ;21168  ;       Error2: If this error occur then the BRANC 2 PAL should be checked for
                ;21169  ;               error. If the PAL is not the cause of the error then the stack
                ;21170  ;               control bits to the microsequencer should be checked to be
                ;21171  ;               assserted. The select bits to the micro sequencer should be
                ;21172  ;               checked to be 10 and register enable should be asserted.
                ;21173  ;       Error3: If this error occurs then the caries from one of the microse-
                ;21174  ;               quencer chips failed.
                ;21175  ;       Error4: If this error occurs then there is something wrong with one of
                ;21176  ;               the memory locations in the stack and the microsequncer chip
                ;21177  ;               with the error should probably be replaced.
                ;21178  ;       Error5: Same as error4.
                ;21179  ;       Error6: Same as error4.
                ;21180  ;       Error7: Same as error4.
                ;21181  ;       Error8: Same as error4.
                ;21182  ;       Error9: Same as error4.
                ;21183  ;       ErrorA: Same as error4.
                ;21184  ;       ErrorB: Same as error4.
                ;21185  ;       ErrorC: Same as error4.
                ;21186  ;       ErrorD: Same as error4.
                ;21187  ;       ErrorE: Same as error4.
                ;21188  ;       ErrorF: Same as error4.
                ;21189  ;       Error10: Same as error4.
                ;21190  ;       Error11: Same as error4.
                ;21191  ;       Error12: Same as error4.
                ;21192  ;       Error13: Same as error4.
                ;21193  ;       Error14: If this error occurs then there is something wrong with the
                ;21194  ;               push pop mechanism in the stack of the microsequencer chip.
                ;21195  ;               Depending on which bits are in error isolation to one of the
                ;21196  ;               three microsequencer chips should be possible.
                ;21197  ;       Error15: Same as error13.
                ;21198  ;       Error16: Same as error13.
                ;21199  ;       Error17: Same as error13.
                ;21200  ;
                ;21201  ;
                ;21202  ;--
                ;21203  T.3D:
U 20B1, B65E,15 ;21204          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
```

```
                ;21205                                          ; STATUS WORD
U 20B2, 3E80,15 ;21206          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;21207                                          ; BIT 15 INDICATES START OF TEST TO
                ;21208                                          ; CONSOLE
U 20B3, 10E0,15 ;21209          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;21210  WAIT.T3D.0:
U 20B4, 0A0B,44 ;21211          JMP [WAIT.T3D.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;21212                                          ; RESPOND
U 20B5, 887E,EC ;21213          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;21214                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 20B6, 378A,15 ;21215          MOV LS[T3D.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 20B7, BE12,15 ;21216          MOV WR[0] TO LS[T9]             ; references.
U 20B8, B72E,15 ;21217          MOV LS[#2A7] TO WR[0]           ;Get LOAD address
U 20B9, BE14,15 ;21218          MOV WR[0] TO LS[T10]
U 20BA, 8870,3C ;21219          JSR [L0]                        ;Get address of Call #1
U 20BB, 3EAE,15 ;21220          MOV WR[0] TO LS[T57]
U 20BC, 8870,3C ;21221          JSR [L0]                        ;Get address of Call #2
U 20BD, 3EB0,15 ;21222          MOV WR[0] TO LS[T58]
U 20BE, 8870,3C ;21223          JSR [L0]                        ;Get address of Call #3
U 20BF, BEB2,15 ;21224          MOV WR[0] TO LS[T59]
U 20C0, 8870,3C ;21225          JSR [L0]                        ;Get address of Call #4
U 20C1, 3E10,15 ;21226          MOV WR[0] TO LS[T8]
U 20C2, 8870,3C ;21227          JSR [L0]                        ;Get address of Call #5
U 20C3, BF08,15 ;21228          MOV WR[0] TO LS[T84]
U 20C4, 8870,3C ;21229          JSR [L0]                        ;Get address of Return
U 20C5, 3E1A,15 ;21230          MOV WR[0] TO LS[T13]
U 20C6, 8870,3C ;21231          JSR [L0]                        ;Get address to return to
U 20C7, 3E1C,15 ;21232          MOV WR[0] TO LS[T14]            ; from call #1
U 20C8, 8870,3C ;21233          JSR [L0]                        ;Get address to return to from
U 20C9, BF8C,15 ;21234          MOV WR[0] TO LS[TC5]            ; call #2
U 20CA, 8870,3C ;21235          JSR [L0]                        ;Get address to return to
U 20CB, 3E16,15 ;21236          MOV WR[0] TO LS[T11]            ; from call #3
U 20CC, 8870,3C ;21237          JSR [L0]                        ;Get address to return to
U 20CD, 3F90,15 ;21238          MOV WR[0] TO LS[TC8]            ; from call #4
U 20CE, 8870,3C ;21239          JSR [L0]                        ;Get address to return from
U 20CF, BF92,15 ;21240          MOV WR[0] TO LS[TC9]            ; from call #5
U 20D0, B6AE,15 ;21241  T3D.1:  MOV LS[T57] TO WR[0]
U 20D1, 90F6,15 ;21242          MISC2 [TRAP.ACC] WR[0]          ;Force Call at 210
U 20D2, 10F8,15 ;21243          MISC2 [READ.ACC.UPC]            ;Read the address that the call is
                ;21244                                          ; going to
U 20D3, 3A18,15 ;21245          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 20D4, 3618,15 ;21246          MOV LS[T12] TO WR[0]            ;Set up received data
U 20D5, B648,95 ;21247          MOV LS[#10] TO WR[1]            ;Set up expected data
U 20D6, 2102,95 ;21248          DEC WR[1]
U 20D7, 0869,3C ;21249          JSR [CHECK.RESULT]              ;Check for error
U 20D8, 8A0D,04 ;21250          JMP [T3D.1]
U 20D9, FF82,15 ;21251          INC LS[ERROR.NUMBER]
U 20DA, B61A,15 ;21252  T3D.2:  MOV LS[T13] TO WR[0]
U 20DB, 90F6,15 ;21253          MISC2 [TRAP.ACC] WR[0]          ;Force Return
U 20DC, 10F8,15 ;21254          MISC2 [READ.ACC.UPC]
U 20DD, 3A18,15 ;21255          MOV FPA TO LS[T12]              ;Get address returned to
U 20DE, 3618,15 ;21256          MOV LS[T12] TO WR[0]            ;Setup received data
U 20DF, 361C,95 ;21257          MOV LS[T14] TO WR[1]            ;Set up expected data
U 20E0, 0869,3C ;21258          JSR [CHECK.RESULT]              ;Check for error
U 20E1, 8A0D,A4 ;21259          JMP [T3D.2]                     ;Loop on error
```

```
U 20E2, FF82,15 ;21260          INC LS[ERROR.NUMBER]
U 20E3, 3708,15 ;21261  T3D.3:  MOV LS[T84] TO WR[0]
U 20E4, 90F6,15 ;21262          MISC2 [TRAP.ACC] WR[0]          ;Force call to subroutine
U 20E5, 361A,95 ;21263          MOV LS[T13] TO WR[1]
U 20E6, 10F6,95 ;21264          MISC2 [TRAP.ACC] WR[1]          ;Force return from subroutine
U 20E7, 10F8,15 ;21265          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 20E8, 3A18,15 ;21266          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 20E9, 3618,15 ;21267          MOV LS[T12] TO WR[0]            ;Setup received data
U 20EA, B792,95 ;21268          MOV LS[TC9] TO WR[1]            ;Setup expected data
U 20EB, 0869,3C ;21269          JSR [CHECK.RESULT]              ;Check for error
U 20EC, 8A0E,34 ;21270          JMP [T3D.3]                     ;Loop on error
U 20ED, FF82,15 ;21271          INC LS[ERROR.NUMBER]
U 20EE, 36AF,15 ;21272  T3D.4:  MOV LS[T57] TO WR[2]            ;Check all stack locations for #1
U 20EF, B61D,95 ;21273          MOV LS[T14] TO WR[3]
U 20F0, 0A12,DC ;21274          JSR [CHECK.STACK]
U 20F1, FF82,15 ;21275          INC LS[ERROR.NUMBER]
U 20F2, 36B1,15 ;21276  T3D.5:  MOV LS[T58] TO WR[2]            ;Check all stack locations for #2
U 20F3, 378D,95 ;21277          MOV LS[TC5] TO WR[3]
U 20F4, 0A12,DC ;21278          JSR [CHECK.STACK]
U 20F5, FF82,15 ;21279          INC LS[ERROR.NUMBER]
U 20F6, B6B3,15 ;21280  T3D.6:  MOV LS[T59] TO WR[2]            ;Check all stack locations for #3
U 20F7, B617,95 ;21281          MOV LS[T11] TO WR[3]
U 20F8, 0A12,DC ;21282          JSR [CHECK.STACK]
U 20F9, FF82,15 ;21283          INC LS[ERROR.NUMBER]
U 20FA, 3611,15 ;21284  T3D.7:  MOV LS[T8] TO WR[2]             ;Check all stack locations for #4
U 20FB, B791,95 ;21285          MOV LS[TC8] TO WR[3]
U 20FC, 0A12,DC ;21286          JSR [CHECK.STACK]
U 20FD, FF82,15 ;21287          INC LS[ERROR.NUMBER]
U 20FE, 3648,15 ;21288  T3D.8:  MOV LS[#10] TO WR[0]            ;Set error number to 14
U 20FF, 4744,15 ;21289          BIS LS[#4] TO WR[0]
U 2100, BE82,15 ;21290          MOV WR[0] TO LS[ERROR.NUMBER]
U 2101, B6AE,15 ;21291          MOV LS[T57] TO WR[0]            ;Force Call #1
U 2102, 90F6,15 ;21292          MISC2 [TRAP.ACC] WR[0]
U 2103, B6B0,15 ;21293          MOV LS[T58] TO WR[0]            ;Force Call #2
U 2104, 90F6,15 ;21294          MISC2 [TRAP.ACC] WR[0]
U 2105, 36B2,15 ;21295          MOV LS[T59] TO WR[0]            ;Force Call #3
U 2106, 90F6,15 ;21296          MISC2 [TRAP.ACC] WR[0]
U 2107, B610,15 ;21297          MOV LS[T8] TO WR[0]             ;Force Call #4
U 2108, 90F6,15 ;21298          MISC2 [TRAP.ACC] WR[0]
U 2109, B61A,15 ;21299          MOV LS[T13] TO WR[0]            ;Force return
U 210A, 90F6,15 ;21300          MISC2 [TRAP.ACC] WR[0]
U 210B, 10F8,15 ;21301          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 210C, 3A18,15 ;21302          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 210D, 3618,15 ;21303          MOV LS[T12] TO WR[0]            ;Setup received data
U 210E, 3790,95 ;21304          MOV LS[TC8] TO WR[1]            ;Setup expected data
U 210F, 0869,3C ;21305          JSR [CHECK.RESULT]              ;Check for error
U 2110, 8A0F,E4 ;21306          JMP [T3D.8]                     ;Loop on error
U 2111, FF82,15 ;21307          INC LS[ERROR.NUMBER]            ;Go onto error 15
U 2112, B61A,15 ;21308          MOV LS[T13] TO WR[0]            ;Force return
U 2113, 90F6,15 ;21309          MISC2 [TRAP.ACC] WR[0]
U 2114, 10F8,15 ;21310          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2115, 3A18,15 ;21311          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2116, 3618,15 ;21312          MOV LS[T12] TO WR[0]            ;Setup received data
U 2117, 3616,95 ;21313          MOV LS[T11] TO WR[1]            ;Setup expected data
U 2118, 0869,3C ;21314          JSR [CHECK.RESULT]              ;Check for error
```

```
U 2119, 8A0F,E4 ;21315          JMP [T3D.8]                     ;Loop on error
U 211A, FF82,15 ;21316          INC LS[ERROR.NUMBER]            ;Go onto error 16
U 211B, B61A,15 ;21317          MOV LS[T13] TO WR[0]            ;Force return
U 211C, 90F6,15 ;21318          MISC2 [TRAP.ACC] WR[0]
U 211D, 10F8,15 ;21319          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 211E, 3A18,15 ;21320          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 211F, 3618,15 ;21321          MOV LS[T12] TO WR[0]            ;Setup received data
U 2120, B78C,95 ;21322          MOV LS[TC5] TO WR[1]            ;Setup expected data
U 2121, 0869,3C ;21323          JSR [CHECK.RESULT]              ;Check for error
U 2122, 8A0F,E4 ;21324          JMP [T3D.8]                     ;Loop on error
U 2123, FF82,15 ;21325          INC LS[ERROR.NUMBER]            ;Go onto error 17
U 2124, B61A,15 ;21326          MOV LS[T13] TO WR[0]            ;Force return
U 2125, 90F6,15 ;21327          MISC2 [TRAP.ACC] WR[0]
U 2126, 10F8,15 ;21328          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2127, 3A18,15 ;21329          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2128, 3618,15 ;21330          MOV LS[T12] TO WR[0]            ;Setup received data
U 2129, 361C,95 ;21331          MOV LS[T14] TO WR[1]            ;Setup expected data
U 212A, 0869,3C ;21332          JSR [CHECK.RESULT]              ;Check for error
U 212B, 8A0F,E4 ;21333          JMP [T3D.8]                     ;Loop on error
U 212C, 0A16,B4 ;21334          JMP [T3D.END]
                ;21335
                ;21336  ; CHECK STACK SUNROUTINE
                ;21337  ;
                ;21338  ;       The purpose of this subroutine is to check the four stack locations
                ;21339  ;       for a pattern that is passed in WR[2].
                ;21340
                ;21341  CHECK.STACK:
U 212D, 2004,15 ;21342          MOV WR[2] TO WR[0]
U 212E, 90F6,15 ;21343          MISC2 [TRAP.ACC] WR[0]          ;Force Call
U 212F, B61A,15 ;21344          MOV LS[T13] TO WR[0]
U 2130, 90F6,15 ;21345          MISC2 [TRAP.ACC] WR[0]          ;Force return
U 2131, 10F8,15 ;21346          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2132, 3A18,15 ;21347          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2133, 3716,15 ;21348          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 2134, 90F6,15 ;21349          MISC2 [TRAP.ACC] WR[0]          ;Loop self
U 2135, 3618,15 ;21350          MOV LS[T12] TO WR[0]            ;Set up received data
U 2136, 2006,95 ;21351          MOV WR[3] TO WR[1]              ;Set up expected data
U 2137, 0869,3C ;21352          JSR [CHECK.RESULT]              ;Check for error
U 2138, 8A12,D4 ;21353          JMP [CHECK.STACK]               ;Loop on error
U 2139, FF82,15 ;21354          INC LS[ERROR.NUMBER]
U 213A, B61A,15 ;21355          MOV LS[T13] TO WR[0]
U 213B, 90F6,15 ;21356          MISC2 [TRAP.ACC] WR[0]          ;Force return to set stack to next
U 213C, 2004,15 ;21357  T31.A:  MOV WR[2] TO WR[0]
U 213D, 90F6,15 ;21358          MISC2 [TRAP.ACC] WR[0]          ;Force Call
U 213E, B61A,15 ;21359          MOV LS[T13] TO WR[0]
U 213F, 90F6,15 ;21360          MISC2 [TRAP.ACC] WR[0]          ;Force return
U 2140, 10F8,15 ;21361          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2141, 3A18,15 ;21362          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2142, 3716,15 ;21363          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 2143, 90F6,15 ;21364          MISC2 [TRAP.ACC] WR[0]
U 2144, 3618,15 ;21365          MOV LS[T12] TO WR[0]            ;Set up received data
U 2145, 2006,95 ;21366          MOV WR[3] TO WR[1]              ;Set up expected data
U 2146, 0869,3C ;21367          JSR [CHECK.RESULT]              ;Check for error
U 2147, 8A13,C4 ;21368          JMP [T31.A]                     ;Loop on error
U 2148, FF82,15 ;21369          INC LS[ERROR.NUMBER]
```

```
U 2149, B61A,15 ;21370          MOV LS[T13] TO WR[0]
U 214A, 90F6,15 ;21371          MISC2 [TRAP.ACC] WR[0]          ;Force return to set stack to next
U 214B, 2004,15 ;21372  T31.B:  MOV WR[2] TO WR[0]
U 214C, 90F6,15 ;21373          MISC2 [TRAP.ACC] WR[0]          ;Force Call
U 214D, B61A,15 ;21374          MOV LS[T13] TO WR[0]
U 214E, 90F6,15 ;21375          MISC2 [TRAP.ACC] WR[0]          ;Force return
U 214F, 10F8,15 ;21376          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2150, 3A18,15 ;21377          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2151, 3716,15 ;21378          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 2152, 90F6,15 ;21379          MISC2 [TRAP.ACC] WR[0]          ;Loop self
U 2153, 3618,15 ;21380          MOV LS[T12] TO WR[0]            ;Set up received data
U 2154, 2006,95 ;21381          MOV WR[3] TO WR[1]              ;Setup expected data
U 2155, 0869,3C ;21382          JSR [CHECK.RESULT]              ;Check for error
U 2156, 8A14,B4 ;21383          JMP [T31.B]                     ;Loop on error
U 2157, FF82,15 ;21384          INC LS[ERROR.NUMBER]
U 2158, B61A,15 ;21385          MOV LS[T13] TO WR[0]
U 2159, 90F6,15 ;21386          MISC2 [TRAP.ACC] WR[0]          ;Force return to set stack to next
U 215A, 2004,15 ;21387  T31.C:  MOV WR[2] TO WR[0]
U 215B, 90F6,15 ;21388          MISC2 [TRAP.ACC] WR[0]          ;Force Call
U 215C, B61A,15 ;21389          MOV LS[T13] TO WR[0]
U 215D, 90F6,15 ;21390          MISC2 [TRAP.ACC] WR[0]          ;Force return
U 215E, 10F8,15 ;21391          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 215F, 3A18,15 ;21392          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2160, 3716,15 ;21393          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 2161, 90F6,15 ;21394          MISC2 [TRAP.ACC] WR[0]
U 2162, 3618,15 ;21395          MOV LS[T12] TO WR[0]            ;Set up received data
U 2163, 2006,95 ;21396          MOV WR[3] TO WR[1]              ;Set up expected data
U 2164, 0869,3C ;21397          JSR [CHECK.RESULT]              ;Check for error
U 2165, 8A15,A4 ;21398          JMP [T31.C]                     ;Loop on error
U 2166, B61A,15 ;21399          MOV LS[T13] TO WR[0]
U 2167, 90F6,15 ;21400          MISC2 [TRAP.ACC] WR[0]          ;Force return to set stack to next
U 2168, 3716,15 ;21401          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 2169, 90F6,15 ;21402          MISC2 [TRAP.ACC] WR[0]          ;Loop self
U 216A, 5B00,14 ;21403          RETURN
                ;21404
                ;21405  T3D.END:
```

```
                    ;21406  .page
                    ;21407  .TOC    "TEST 3E RETURN (OP1+OP2)/=0 AND RETURN EXP15 F3 TEST"
                    ;21408  ;++
                    ;21409  ;
                    ;21410  ; Test Description:
                    ;21411  ;
                    ;21412  ;       The purpose of this test is to check the return from subroutine
                    ;21413  ;       instructions with branch conditions. There are two return from
                    ;21414  ;       from subroutine instructions with branch conditions. If the branch
                    ;21415  ;       conditions is asserted then the return will be +2 instead of +1.
                    ;21416  ;       The branch conditions that are returned on are (OP1+OP2)/=0 and
                    ;21417  ;       EXP15 F3. Both branch conditions will be tested asserted and
                    ;21418  ;       unasserted.
                    ;21419  ;
                    ;21420  ; Assumptions:
                    ;21421  ;
                    ;21422  ;       It is assumed that the Call and the Return with no branch condition
                    ;21423  ;       have been tested previously and are working.
                    ;21424  ;
                    ;21425  ; Test Steps:
                    ;21426  ;
                    ;21427  ;       1) Force a Call to subroutine. Force a return from subroutine with
                    ;21428  ;          the branch condition (OP1+OP2)/=0. Then check for error.
                    ;21429  ;       2) Assert (OP1+OP2)/=0. Force a Call to subroutine. Force a return
                    ;21430  ;          from subroutine with the branch condition (OP1+OP2)/=0. Then
                    ;21431  ;          Check for error.
                    ;21432  ;       3) Deassert EXP15 F3. Force a Call to subroutine. Force a return from
                    ;21433  ;          subroutine with the branch condition EXP15 F3. Then check for error
                    ;21434  ;       4) Assert EXP15 F3. Force a Call to subroutine. Force a Return from
                    ;21435  ;          subroutine with the branch condition EXP15 F3. Then check for error.
                    ;21436  ;
                    ;21437  ; Error:
                    ;21438  ;
                    ;21439  ;       Error1 - Return with (OP1+OP2)/=0 branch failed with (OP1+OP2)/=0
                    ;21440  ;                unasserted.
                    ;21441  ;       Error2 - Return with (OP1+OP2)/=0 branch failed with (OP1+OP2)/=0
                    ;21442  ;                asserted.
                    ;21443  ;       Error3 - Return with EXP15 F3 branch failed with EXP15 F3 unasserted.
                    ;21444  ;       Error4 - Return with EXP15 F3 branch failed with EXP15 F3 asserted.
                    ;21445  ;
                    ;21446  ; Debug:
                    ;21447  ;
                    ;21448  ;       Error1: If this error occurs then the BRANCH 2 PAL and the BRANCH 3
                    ;21449  ;               PAL should be checked for error. If the PAL's are not in error
                    ;21450  ;               then the UCBTL bits should be checked to be 1C.
                    ;21451  ;       Error2: Same as error 1 except that (OP1+OP2)/=0 is asserted.
                    ;21452  ;       Error3: Same as error 1 except that the UCBTL bits should be 1F.
                    ;21453  ;       Error4: Same as error 3 except that EXP15 F3 is asserted.
                    ;21454  ;
                    ;21455  ;
                    ;21456  ;--
                    ;21457  T.3E:
U 216B, B65E,15     ;21458          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                    ;21459                                          ; STATUS WORD
U 216C, 3E80,15     ;21460          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
```

```
                ;21461                                  ; BIT 15 INDICATES START OF TEST TO
                ;21462                                  ; CONSOLE
U 216D, 10E0,15 ;21463          MISC [SET.CP.ATTN]      ;ISSUE CPU ATTN TO CONSOLE
                ;21464  WAIT.T3E.0:
U 216E, 0A16,E4 ;21465          JMP [WAIT.T3E.0]        ;LOOP HERE WAITING FOR CONSOLE TO
                ;21466                                  ; RESPOND
U 216F, 887E,EC ;21467          JSR [SETUP.10]          ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;21468                                  ; AND SIZE BITS. CHECK LOWER 10 BITS
U 2170, 3794,15 ;21469          MOV LS[T3E.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2171, BE12,15 ;21470          MOV WR[0] TO LS[T9]     ; references.
U 2172, B72E,15 ;21471          MOV LS[#2A7] TO WR[0]   ;Get load address
U 2173, BE14,15 ;21472          MOV WR[0] TO LS[T10]
U 2174, 8870,3C ;21473          JSR [LO]                ;Get address of Call to subroutine
U 2175, 3E10,15 ;21474          MOV WR[0] TO LS[T8]
U 2176, 8870,3C ;21475          JSR [LO]                ;Get address of Return(OP1+OP2)/=0
U 2177, BE1E,15 ;21476          MOV WR[0] TO LS[T15]
U 2178, 8870,3C ;21477          JSR [LO]                ;Get address of Return(EXP15 F3)
U 2179, 3EAE,15 ;21478          MOV WR[0] TO LS[T57]
U 217A, 8870,3C ;21479          JSR [LO]                ;Get address of MOV ET0 TO EQ
U 217B, 3E1C,15 ;21480          MOV WR[0] TO LS[T14]
U 217C, 8870,3C ;21481          JSR [LO]                ;Get address of MOV EQ TO ET0 and
U 217D, 3E1A,15 ;21482          MOV WR[0] TO LS[T13]    ; CLOCK OP2=0
U 217E, 8870,3C ;21483          JSR [LO]                ;Get address of SHIFT LEFT ET0
U 217F, 3E16,15 ;21484          MOV WR[0] TO LS[T11]
U 2180, 8870,3C ;21485          JSR [LO]                ;Get address of the RETURN NUA
U 2181, 3EB0,15 ;21486          MOV WR[0] TO LS[T58]
U 2182, 8870,3C ;21487          JSR [LO]                ;Get address of then RETURN+1 NUA
U 2183, BEB2,15 ;21488          MOV WR[0] TO LS[T59]
U 2184, 8870,3C ;21489          JSR [LO]                ;Get address of return when EXP15 F3
U 2185, BF08,15 ;21490          MOV WR[0] TO LS[T84]    ; condition is asserted
U 2186, B69E,15 ;21491  T3E.1:  MOV LS[#FFFFFFFF] TO WR[0]      ;Set up DATA.1
U 2187, 3EA2,15 ;21492          MOV WR[0] TO LS[DATA.1]
U 2188, 8877,DC ;21493          JSR [LOAD.DATA]         ;Load ones into ET0
U 2189, 0878,AC ;21494          JSR [MOV.DATA]          ;Load ones into EQ
U 218A, B61A,15 ;21495          MOV LS[T13] TO WR[0]
U 218B, 3610,95 ;21496          MOV LS[T8] TO WR[1]
U 218C, 90F6,15 ;21497          MISC2 [TRAP.ACC] WR[0]  ;CLOCK OP1=0
U 218D, 90F6,15 ;21498          MISC2 [TRAP.ACC] WR[0]  ;CLOCK OP2=0
U 218E, 10F6,95 ;21499          MISC2 [TRAP.ACC] WR[1]  ;Call to subroutine
U 218F, 361E,15 ;21500          MOV LS[T15] TO WR[0]
U 2190, 90F6,15 ;21501          MISC2 [TRAP.ACC] WR[0]  ;Return from subroutine
U 2191, 10F8,15 ;21502          MISC2 [READ.ACC.UPC]    ;Enable the NUA BUS onto the FPA BUS
U 2192, 3A18,15 ;21503          MOV FPA TO LS[T12]
U 2193, 3716,15 ;21504          MOV LS[#300] TO WR[0]   ;Setup to Loop self
U 2194, 90F6,15 ;21505          MISC2 [TRAP.ACC] WR[0]  ;Loop self
U 2195, 3618,15 ;21506          MOV LS[T12] TO WR[0]    ;Set up received data
U 2196, B6B2,95 ;21507          MOV LS[T59] TO WR[1]    ;Set up expected data
U 2197, 0869,3C ;21508          JSR [CHECK.RESULT]      ;Check for error
U 2198, 0A18,64 ;21509          JMP [T3E.1]             ;Loop on error
U 2199, FF82,15 ;21510          INC LS[ERROR.NUMBER]    ;Go onto next error
U 219A, E5A2,15 ;21511  T3E.2:  CLR LS[DATA.1]
U 219B, 8877,DC ;21512          JSR [LOAD.DATA]         ;Load ones into ET0
U 219C, 0878,AC ;21513          JSR [MOV.DATA]          ;Load ones into EQ
U 219D, B61A,15 ;21514          MOV LS[T13] TO WR[0]
U 219E, 3610,95 ;21515          MOV LS[T8] TO WR[1]
```

```
U 219F, 90F6,15 ;21516          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1=0
U 21A0, 90F6,15 ;21517          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2=0
U 21A1, 10F6,95 ;21518          MISC2 [TRAP.ACC] WR[1]          ;Call to subroutine
U 21A2, 361E,15 ;21519          MOV LS[T15] TO WR[0]
U 21A3, 90F6,15 ;21520          MISC2 [TRAP.ACC] WR[0]          ;Return from subroutine
U 21A4, 10F8,15 ;21521          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 21A5, 3A18,15 ;21522          MOV FPA TO LS[T12]
U 21A6, 3716,15 ;21523          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 21A7, 90F6,15 ;21524          MISC2 [TRAP.ACC] WR[0]
U 21A8, 3618,15 ;21525          MOV LS[T12] TO WR[0]            ;Set up received data
U 21A9, 36B0,95 ;21526          MOV LS[T58] TO WR[1]            ;Set up expected data
U 21AA, 0869,3C ;21527          JSR [CHECK.RESULT]              ;Check for error
U 21AB, 8A19,A4 ;21528          JMP [T3E.2]                     ;Loop on error
U 21AC, FF82,15 ;21529          INC LS[ERROR.NUMBER]            ;Go onto next error
U 21AD, E5A2,15 ;21530  T3E.3:  CLR LS[DATA.1]                  ;Load zeros into ET0
U 21AE, 8877,DC ;21531          JSR [LOAD.DATA]
U 21AF, B610,15 ;21532          MOV LS[T8] TO WR[0]             ;Call to subroutine
U 21B0, 90F6,15 ;21533          MISC2 [TRAP.ACC] WR[0]
U 21B1, B6AE,15 ;21534          MOV LS[T57] TO WR[0]            ;Return from subroutine
U 21B2, 90F6,15 ;21535          MISC2 [TRAP.ACC] WR[0]
U 21B3, 10F8,15 ;21536          MISC2 [READ.ACC.UPC]            ;Enable the NUS BUS onto the FPA BUS
U 21B4, 3A18,15 ;21537          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 21B5, 3716,15 ;21538          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 21B6, 90F6,15 ;21539          MISC2 [TRAP.ACC] WR[0]          ;Loop self
U 21B7, 3618,15 ;21540          MOV LS[T12] TO WR[0]            ;Set up received data
U 21B8, 36B0,95 ;21541          MOV LS[T58] TO WR[1]            ;Set up expected data
U 21B9, 0869,3C ;21542          JSR [CHECK.RESULT]              ;Check for error
U 21BA, 0A1A,D4 ;21543          JMP [T3E.3]                     ;Loop on error
U 21BB, FF82,15 ;21544          INC LS[ERROR.NUMBER]            ;Go onto next error
U 21BC, B69E,15 ;21545  T3E.4:  MOV LS[#FFFFFFFF] TO WR[0]      ;Load ET0 with ones
U 21BD, 3EA2,15 ;21546          MOV WR[0] TO LS[DATA.1]
U 21BE, 8877,DC ;21547          JSR [LOAD.DATA]
U 21BF, 0876,2C ;21548          JSR [SHIFT.LEFT.8]              ;Load upper bits with ones
U 21C0, B610,15 ;21549          MOV LS[T8] TO WR[0]             ;Call to subroutine
U 21C1, 90F6,15 ;21550          MISC2 [TRAP.ACC] WR[0]
U 21C2, B6AE,15 ;21551          MOV LS[T57] TO WR[0]            ;Return from subroutine
U 21C3, 90F6,15 ;21552          MISC2 [TRAP.ACC] WR[0]
U 21C4, 10F8,15 ;21553          MISC2 [READ.ACC.UPC]            ;Enable the NUS BUS onto the FPA BUS
U 21C5, 3A18,15 ;21554          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 21C6, 3716,15 ;21555          MOV LS[#300] TO WR[0]           ;Setup to Loop self
U 21C7, 90F6,15 ;21556          MISC2 [TRAP.ACC] WR[0]          ;Loop self
U 21C8, 3618,15 ;21557          MOV LS[T12] TO WR[0]            ;Set up received data
U 21C9, B708,95 ;21558          MOV LS[T84] TO WR[1]            ;Set up expected data
U 21CA, 0869,3C ;21559          JSR [CHECK.RESULT]              ;Check for error
U 21CB, 0A1B,C4 ;21560          JMP [T3E.4]                     ;Loop on error
                ;21561  T3E.END:
```

```
;21562  .page
;21563  .TOC    "TEST 3F SUMPATH BRANCH TEST(M8389 FPA MODULE)"
;21564  ;++
;21565  ;
;21566  ;
;21567  ; Test Description:
;21568  ;
;21569  ;       The purpose of this test is to check the SUMPATH branch condition. The
;21570  ;       data path will be as follows: the branch condition will be asserted
;21571  ;       with the branch field set to 11011. If there is no branch or an incor-
;21572  ;       rect branch then an error statement will be issued. This procedure will
;21573  ;       be repeated for the branch condition unasserted. SUMPATH will be
;21574  ;       asserted under a variety of conditions that will be tested below.
;21575  ;
;21576  ; Assumptions:
;21577  ;
;21578  ;       It is assumed that the branch control logic has been tested previously
;21579  ;       and works.
;21580  ;
;21581  ; Test Steps:
;21582  ;
;21583  ;       1) Set the instruction bits to 00100. Assert OP2S and ENB CLK4. Branch
;21584  ;          on SUMPATH. SUMPATH should be unasserted.
;21585  ;       2) Branch on SUMPATH. SUMPATH should be unasserted.
;21586  ;       3) Assert OP2S and SIGN OUT. Branch on SUMPATH. SUMPATH should be
;21587  ;          asserted.
;21588  ;       4) Assert OP1S and SIGN OUT. Branch on SUMPATH. SUMPATH should be
;21589  ;          asserted.
;21590  ;       5) Set the instruction bits to 00000. Branch on SUMPATH. SUMPATH should
;21591  ;          be asserted.
;21592  ;       6) Assert OP1S. Branch on SUMPATH. SUMPATH should be unasserted.
;21593  ;       7) Assert OP2S. Branch on SUMPATH. SUMPATH should be unasserted.
;21594  ;       8) Assert OP1S and OP2S. Branch on SUMPATH. SUMPATH should be asserted.
;21595  ;       9) Set the instruction bits to 00011. Assert OP1S and OP2S. Branch on
;21596  ;          SUMPATH. SUMPATH should be unasserted.
;21597  ;      10) Assert OP1S. Branch on SUMPATH. SUMPATH should be asserted.
;21598  ;      11) Set the instruction bits to 00010. Assert OP2S and ENB CLK7. Branch
;21599  ;          on SUMPATH. SUMPATH should be asserted.
;21600  ;      12) Set the instruciton bits to 00101. Assert OP2S and ENB CLK4. Branch
;21601  ;          on SUMPATH. SUMPATH should be unasserted.
;21602  ;      13) Set the instruction bits to 00001. Branch on SUMPATH. SUMPATH should
;21603  ;          be unasserted.
;21604  ;      14) Set the instruction bits to 00110. Assert OP2S. Branch on SUMPATH.
;21605  ;          SUMPATH should be unasserted.
;21606  ;      15) Set the instruction bits to 10011. Assert OP2S. Branch on SUMPATH.
;21607  ;          SUMPATH should be unasserted.
;21608  ;      16) Set the instruction bits to 01011. Assert OP2S. Branch on SUMPATH.
;21609  ;          SUMPATH should be unasserted.
;21610  ;
;21611  ; Error:
;21612  ;
;21613  ;       Error1 - Branch on SUMPATH unasserted and the conditions mentioned in
;21614  ;                step1.
;21615  ;       Error2 - Branch on SUMPATH unasserted and the conditions mentioned in
;21616  ;                step2.
```

```
;21617  ;       Error3 - Branch on SUMPATH asserted and the conditions mentioned in
;21618  ;                step3.
;21619  ;       Error4 - Branch on SUMPATH asserted and the conditions mentioned in
;21620  ;                step4.
;21621  ;       Error5 - Branch on SUMPATH asserted and the conditions mentioned in
;21622  ;                step5.
;21623  ;       Error6 - Branch on SUMPATH unasserted and the conditions mentioned in
;21624  ;                step6.
;21625  ;       Error7 - Branch on SUMPATH unasserted and the conditions mentioned in
;21626  ;                step7.
;21627  ;       Error8 - Branch on SUMPATH asserted and the conditions mentioned in
;21628  ;                step8.
;21629  ;       Error9 - Branch on SUMPATH unasserted and the conditions mentioned in
;21630  ;                step9.
;21631  ;       ErrorA - Branch on SUMPATH asserted and the conditions mentioned in
;21632  ;                 step10.
;21633  ;       ErrorB - Branch on SUMPATH asserted and the conditions mentioned in
;21634  ;                 step11.
;21635  ;       ErrorC - Branch on SUMPATH unasserted and the conditions mentioned in
;21636  ;                 step12.
;21637  ;       ErrorD - Branch on SUMPATH unasserted and the conditions mentioned in
;21638  ;                 step13.
;21639  ;       ErrorE - Branch on SUMPATH unasserted and the conditions mentioned in
;21640  ;                 step14.
;21641  ;       ErrorF - Branch on SUMPATH unasserted and the conditions mentioned in
;21642  ;                 step15.
;21643  ;       Error10 - Branch on SUMPATH unasserted and the conditions mentioned in
;21644  ;                 step16.
;21645  ;
;21646  ; Debug:
;21647  ;
;21648  ;       Error1: If this error occurs then the BRANCH 3 PAL should be checked
;21649  ;               for error. If the PAL is not the cause of the error then the
;21650  ;               UBCTL bits should be checked to be 11011 and SUMPATH should
;21651  ;               be checked to be asserted or unasserted according to the test
;21652  ;               steps. If SUMPATH is in error then the SIGN CTL PAL should be
;21653  ;               checked for error. If the SIGN CTL PAL is not in error then the
;21654  ;               POLY, ADD + SUB, ADD, OP1 SIGN , OP2 SIGN, ENB CLK4 and ENB
;21655  ;               CLK7 should be checked for error. If either ADD + SUB or ADD
;21656  ;               are in error then the INSTRUCTION PAL should be checked for
;21657  ;               error. If the PAL is not the cause of the error then the
;21658  ;               instruction encode bits should be 000XX or 00X00 for ADD + SUB
;21659  ;               and 00000 for ADD if ADD or ADD + SUB are asserted. If POLY is
;21660  ;               in error then the BRANCH 0 PAL should be checked for error. If
;21661  ;               the PAL is not in error then the instruction encode bits should
;21662  ;               be checked to be 00100 if POLY is asserted.
;21663  ;       Error2: Same as error1.
;21664  ;       Error3: Same as error1.
;21665  ;       Error4: Same as error1.
;21666  ;       Error5: Same as error1.
;21667  ;       Error6: Same as error1.
;21668  ;       Error7: Same as error1.
;21669  ;       Error8: Same as error1.
;21670  ;       Error9: Same as error1.
;21671  ;       ErrorA: Same as error1.
```

```
                    ;21672  ;       ErrorB: Same as error1.
                    ;21673  ;       ErrorC: Same as error1.
                    ;21674  ;       ErrorD: Same as error1.
                    ;21675  ;       ErrorE: Same as error1.
                    ;21676  ;       ErrorF: Same as error1.
                    ;21677  ;       Error10: Same as error1.
                    ;21678  ;
                    ;21679  ;
                    ;21680  ;--
                    ;21681  T.3F:
U 21CC, B65E,15 ;21682          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                    ;21683                                      ; STATUS WORD
U 21CD, 3E80,15 ;21684          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                    ;21685                                      ; BIT 15 INDICATES START OF TEST TO
                    ;21686                                      ; CONSOLE
U 21CE, 10E0,15 ;21687          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                    ;21688  WAIT.T3F.0:
U 21CF, 8A1C,F4 ;21689          JMP [WAIT.T3F.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                    ;21690                                      ; RESPOND
U 21D0, 887E,EC ;21691          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                    ;21692                                      ; AND SIZE BITS. CHECK LOWER 10 BITS
U 21D1, B796,15 ;21693          MOV LS[T3F.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 21D2, BE12,15 ;21694          MOV WR[0] TO LS[T9]             ; references.
U 21D3, 8870,3C ;21695          JSR [L0]                        ;Get address of CLOCK SIGN OUT WITH 0
U 21D4, BE14,15 ;21696          MOV WR[0] TO LS[T10]
U 21D5, 8870,3C ;21697          JSR [L0]                        ;Get address of CLOCK SIGN OUT WITH 1
U 21D6, 3E16,15 ;21698          MOV WR[0] TO LS[T11]
U 21D7, 8870,3C ;21699          JSR [L0]                        ;Get address of CLOCK OP1 SIGN
U 21D8, 3E1A,15 ;21700          MOV WR[0] TO LS[T13]
U 21D9, 8870,3C ;21701          JSR [L0]                        ;Get address of CLOCK OP2 SIGN
U 21DA, 3E1C,15 ;21702          MOV WR[0] TO LS[T14]
U 21DB, 8870,3C ;21703          JSR [L0]                        ;Get address of Branch
U 21DC, BE1E,15 ;21704          MOV WR[0] TO LS[T15]
U 21DD, B75C,15 ;21705  T3F.1:  MOV LS[#41300] TO WR[0]         ;Set the instruction and size bits
U 21DE, 90F6,15 ;21706          MISC2 [TRAP.ACC] WR[0]
U 21DF, B61A,15 ;21707          MOV LS[T13] TO WR[0]
U 21E0, B716,95 ;21708          MOV LS[#300] TO WR[1]
U 21E1, 90F6,15 ;21709          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 21E2, 10F6,95 ;21710          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 21E3, B61C,15 ;21711          MOV LS[T14] TO WR[0]
U 21E4, B716,95 ;21712          MOV LS[#300] TO WR[1]
U 21E5, 90F6,15 ;21713          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 21E6, 10F6,95 ;21714          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 21E7, B616,15 ;21715          MOV LS[T11] TO WR[0]
U 21E8, B716,95 ;21716          MOV LS[#300] TO WR[1]
U 21E9, 90F6,15 ;21717          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH 1
U 21EA, 10F6,95 ;21718          MISC2 [TRAP.ACC] WR[1]
U 21EB, 361E,15 ;21719          MOV LS[T15] TO WR[0]
U 21EC, 90F6,15 ;21720          MISC2 [TRAP.ACC] WR[0]          ;BRANCH
U 21ED, 10F8,15 ;21721          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 21EE, 3A18,15 ;21722          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 21EF, 3618,15 ;21723          MOV LS[T12] TO WR[0]            ;Set up received data
U 21F0, B716,95 ;21724          MOV LS[#300] TO WR[1]           ;Set up expected data
U 21F1, 0869,3C ;21725          JSR [CHECK.RESULT]              ;Check for error
U 21F2, 8A1D,D4 ;21726          JMP [T3F.1]                     ;Loop on error
```

```
U 21F3, FF82,15 ;21727          INC LS[ERROR.NUMBER]          ;Go on to next error
U 21F4, B61C,15 ;21728  T3F.2:  MOV LS[T14] TO WR[0]
U 21F5, B776,95 ;21729          MOV LS[#8300] TO WR[1]
U 21F6, 90F6,15 ;21730          MISC2 [TRAP.ACC] WR[0]        ;CLOCK OP2 SIGN
U 21F7, 10F6,95 ;21731          MISC2 [TRAP.ACC] WR[1]        ;Enable data onto bus
U 21F8, B616,15 ;21732          MOV LS[T11] TO WR[0]          ;CLOCK SIGN OUT WITH 1
U 21F9, B61E,95 ;21733          MOV LS[T15] TO WR[1]
U 21FA, 90F6,15 ;21734          MISC2 [TRAP.ACC] WR[0]
U 21FB, 10F6,95 ;21735          MISC2 [TRAP.ACC] WR[1]        ;Branch
U 21FC, 10F8,15 ;21736          MISC2 [READ.ACC.UPC]          ;Enable NUA BUS onto FPA BUS
U 21FD, 3A18,15 ;21737          MOV FPA TO LS[T12]            ;Enable FPA BUS onto Y BUS
U 21FE, 3618,15 ;21738          MOV LS[T12] TO WR[0]          ;Set up received data
U 21FF, B740,95 ;21739          MOV LS[#302] TO WR[1]         ;Set up expected data
U 2200, 0869,3C ;21740          JSR [CHECK.RESULT]            ;Check for error
U 2201, 0A1F,44 ;21741          JMP [T3F.2]                   ;Loop on error
U 2202, FF82,15 ;21742          INC LS[ERROR.NUMBER]          ;Go on to next error
U 2203, B61A,15 ;21743  T3F.3:  MOV LS[T13] TO WR[0]
U 2204, B716,95 ;21744          MOV LS[#300] TO WR[1]
U 2205, 90F6,15 ;21745          MISC2 [TRAP.ACC] WR[0]        ;CLOCK OP1 SIGN
U 2206, 10F6,95 ;21746          MISC2 [TRAP.ACC] WR[1]        ;Enable data onto bus
U 2207, 3614,15 ;21747          MOV LS[T10] TO WR[0]
U 2208, B716,95 ;21748          MOV LS[#300] TO WR[1]
U 2209, 90F6,15 ;21749          MISC2 [TRAP.ACC] WR[0]        ;CLOCK SIGN OUT WITH 0
U 220A, 10F6,95 ;21750          MISC2 [TRAP.ACC] WR[1]
U 220B, 361E,15 ;21751          MOV LS[T15] TO WR[0]
U 220C, 90F6,15 ;21752          MISC2 [TRAP.ACC] WR[0]        ;BRANCH
U 220D, 10F8,15 ;21753          MISC2 [READ.ACC.UPC]          ;Enable NUA BUS onto FPA BUS
U 220E, 3A18,15 ;21754          MOV FPA TO LS[T12]            ;Enable FPA BUS onto Y BUS
U 220F, 3618,15 ;21755          MOV LS[T12] TO WR[0]          ;Set up received data
U 2210, B716,95 ;21756          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2211, 0869,3C ;21757          JSR [CHECK.RESULT]            ;Check for error
U 2212, 8A20,34 ;21758          JMP [T3F.3]                   ;Loop on error
U 2213, FF82,15 ;21759          INC LS[ERROR.NUMBER]          ;Go on to next error
U 2214, B61C,15 ;21760  T3F.4:  MOV LS[T14] TO WR[0]
U 2215, B716,95 ;21761          MOV LS[#300] TO WR[1]
U 2216, 90F6,15 ;21762          MISC2 [TRAP.ACC] WR[0]        ;CLOCK OP2 SIGN
U 2217, 10F6,95 ;21763          MISC2 [TRAP.ACC] WR[1]        ;Enable data onto bus
U 2218, B61A,15 ;21764          MOV LS[T13] TO WR[0]
U 2219, B776,95 ;21765          MOV LS[#8300] TO WR[1]
U 221A, 90F6,15 ;21766          MISC2 [TRAP.ACC] WR[0]        ;CLOCK OP1 SIGN
U 221B, 10F6,95 ;21767          MISC2 [TRAP.ACC] WR[1]        ;Enable data onto bus
U 221C, B61E,95 ;21768          MOV LS[T15] TO WR[1]
U 221D, 10F6,95 ;21769          MISC2 [TRAP.ACC] WR[1]        ;Branch
U 221E, 10F8,15 ;21770          MISC2 [READ.ACC.UPC]          ;Enable NUA BUS onto FPA BUS
U 221F, 3A18,15 ;21771          MOV FPA TO LS[T12]            ;Enable FPA BUS onto Y BUS
U 2220, 3618,15 ;21772          MOV LS[T12] TO WR[0]          ;Set up received data
U 2221, B740,95 ;21773          MOV LS[#302] TO WR[1]         ;Set up expected data
U 2222, 0869,3C ;21774          JSR [CHECK.RESULT]            ;Check for error
U 2223, 8A21,44 ;21775          JMP [T3F.4]                   ;Loop on error
U 2224, FF82,15 ;21776          INC LS[ERROR.NUMBER]          ;Go on to next error
U 2225, B718,15 ;21777  T3F.5:  MOV LS[#00040300] TO WR[0]    ;Set the instruction and size bits
U 2226, 90F6,15 ;21778          MISC2 [TRAP.ACC] WR[0]
U 2227, B61C,15 ;21779          MOV LS[T14] TO WR[0]
U 2228, B776,95 ;21780          MOV LS[#8300] TO WR[1]
U 2229, 90F6,15 ;21781          MISC2 [TRAP.ACC] WR[0]        ;CLOCK OP2 SIGN
```

```
U 222A, 10F6,95 ;21782          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 222B, B61E,95 ;21783          MOV LS[T15] TO WR[1]
U 222C, 10F6,95 ;21784          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 222D, 10F8,15 ;21785          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 222E, 3A18,15 ;21786          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 222F, 3618,15 ;21787          MOV LS[T12] TO WR[0]            ;Set up received data
U 2230, B740,95 ;21788          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2231, 0869,3C ;21789          JSR [CHECK.RESULT]              ;Check for error
U 2232, 0A22,54 ;21790          JMP [T3F.5]                     ;Loop on error
U 2233, FF82,15 ;21791          INC LS[ERROR.NUMBER]            ;Go on to next error
U 2234, B61A,15 ;21792  T3F.6:  MOV LS[T13] TO WR[0]
U 2235, B716,95 ;21793          MOV LS[#300] TO WR[1]
U 2236, 90F6,15 ;21794          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 2237, 10F6,95 ;21795          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2238, B61E,95 ;21796          MOV LS[T15] TO WR[1]
U 2239, 10F6,95 ;21797          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 223A, 10F8,15 ;21798          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 223B, 3A18,15 ;21799          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 223C, 3618,15 ;21800          MOV LS[T12] TO WR[0]            ;Set up received data
U 223D, B716,95 ;21801          MOV LS[#300] TO WR[1]           ;Set up expected data
U 223E, 0869,3C ;21802          JSR [CHECK.RESULT]              ;Check for error
U 223F, 0A23,44 ;21803          JMP [T3F.6]                     ;Loop on error
U 2240, FF82,15 ;21804          INC LS[ERROR.NUMBER]            ;Go on to next error
U 2241, B61A,15 ;21805  T3F.7:  MOV LS[T13] TO WR[0]
U 2242, B776,95 ;21806          MOV LS[#8300] TO WR[1]
U 2243, 90F6,15 ;21807          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 2244, 10F6,95 ;21808          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2245, B61C,15 ;21809          MOV LS[T14] TO WR[0]
U 2246, B716,95 ;21810          MOV LS[#300] TO WR[1]
U 2247, 90F6,15 ;21811          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 2248, 10F6,95 ;21812          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2249, B61E,95 ;21813          MOV LS[T15] TO WR[1]
U 224A, 10F6,95 ;21814          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 224B, 10F8,15 ;21815          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 224C, 3A18,15 ;21816          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 224D, 3618,15 ;21817          MOV LS[T12] TO WR[0]            ;Set up received data
U 224E, B716,95 ;21818          MOV LS[#300] TO WR[1]           ;Set up expected data
U 224F, 0869,3C ;21819          JSR [CHECK.RESULT]              ;Check for error
U 2250, 8A24,14 ;21820          JMP [T3F.7]                     ;Loop on error
U 2251, FF82,15 ;21821          INC LS[ERROR.NUMBER]            ;Go on to next error
U 2252, B61A,15 ;21822  T3F.8:  MOV LS[T13] TO WR[0]
U 2253, B716,95 ;21823          MOV LS[#300] TO WR[1]
U 2254, 90F6,15 ;21824          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 2255, 10F6,95 ;21825          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2256, B61E,95 ;21826          MOV LS[T15] TO WR[1]
U 2257, 10F6,95 ;21827          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2258, 10F8,15 ;21828          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 2259, 3A18,15 ;21829          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 225A, 3618,15 ;21830          MOV LS[T12] TO WR[0]            ;Set up received data
U 225B, B740,95 ;21831          MOV LS[#302] TO WR[1]           ;Set up expected data
U 225C, 0869,3C ;21832          JSR [CHECK.RESULT]              ;Check for error
U 225D, 0A25,24 ;21833          JMP [T3F.8]                     ;Loop on error
U 225E, FF82,15 ;21834          INC LS[ERROR.NUMBER]            ;Go on to next error
U 225F, B718,15 ;21835  T3F.9:  MOV LS[#00040300] TO WR[0]      ;Set the instruction and size bits
U 2260, 4756,15 ;21836          BIS LS[BIT11] TO WR[0]
```

```
U 2261, 90F6,15 ;21837          MISC2 [TRAP.ACC] WR[0]
U 2262, B61E,95 ;21838          MOV LS[T15] TO WR[1]
U 2263, 10F6,95 ;21839          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2264, 10F8,15 ;21840          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 2265, 3A18,15 ;21841          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 2266, 3618,15 ;21842          MOV LS[T12] TO WR[0]            ;Set up received data
U 2267, B716,95 ;21843          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2268, 0869,3C ;21844          JSR [CHECK.RESULT]              ;Check for error
U 2269, 8A25,F4 ;21845          JMP [T3F.9]                     ;Loop on error
U 226A, FF82,15 ;21846          INC LS[ERROR.NUMBER]            ;Go on to next error
U 226B, B61C,15 ;21847  T3F.A:  MOV LS[T14] TO WR[0]
U 226C, B776,95 ;21848          MOV LS[#8300] TO WR[1]
U 226D, 90F6,15 ;21849          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 226E, 10F6,95 ;21850          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 226F, B61A,15 ;21851          MOV LS[T13] TO WR[0]
U 2270, B776,95 ;21852          MOV LS[#8300] TO WR[1]
U 2271, 90F6,15 ;21853          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 2272, 10F6,95 ;21854          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2273, B61E,95 ;21855          MOV LS[T15] TO WR[1]
U 2274, 10F6,95 ;21856          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2275, 10F8,15 ;21857          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 2276, 3A18,15 ;21858          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 2277, 3618,15 ;21859          MOV LS[T12] TO WR[0]            ;Set up received data
U 2278, B716,95 ;21860          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2279, 0869,3C ;21861          JSR [CHECK.RESULT]              ;Check for error
U 227A, 0A26,B4 ;21862          JMP [T3F.A]                     ;Loop on error
U 227B, FF82,15 ;21863          INC LS[ERROR.NUMBER]            ;Go on to next error
U 227C, B61A,15 ;21864  T3F.B:  MOV LS[T13] TO WR[0]
U 227D, B716,95 ;21865          MOV LS[#300] TO WR[1]
U 227E, 90F6,15 ;21866          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 227F, 10F6,95 ;21867          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 2280, B61E,95 ;21868          MOV LS[T15] TO WR[1]
U 2281, 10F6,95 ;21869          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2282, 10F8,15 ;21870          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2283, 3A18,15 ;21871          MOV FPA TO LS[T12]              ;Enable the FPA BUS onto the Y BUS
U 2284, 3618,15 ;21872          MOV LS[T12] TO WR[0]            ;Set up received data
U 2285, B740,95 ;21873          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2286, 0869,3C ;21874          JSR [CHECK.RESULT]              ;Check for error
U 2287, 0A27,C4 ;21875          JMP [T3F.B]                     ;Loop on error
U 2288, FF82,15 ;21876          INC LS[ERROR.NUMBER]            ;Go on to next error
U 2289, B718,15 ;21877  T3F.C:  MOV LS[#00040300] TO WR[0]      ;Set up instruction and size bits
U 228A, 475C,15 ;21878          BIS LS[BIT14] TO WR[0]
U 228B, 4058,15 ;21879          ADD LS[BIT12] TO WR[0]
U 228C, 90F6,15 ;21880          MISC2 [TRAP.ACC] WR[0]
U 228D, B61E,95 ;21881          MOV LS[T15] TO WR[1]
U 228E, 10F6,95 ;21882          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 228F, 10F8,15 ;21883          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 2290, 3A18,15 ;21884          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 2291, 3618,15 ;21885          MOV LS[T12] TO WR[0]            ;Set up received data
U 2292, B716,95 ;21886          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2293, 0869,3C ;21887          JSR [CHECK.RESULT]              ;Check for error
U 2294, 0A28,94 ;21888          JMP [T3F.C]                     ;Loop on error
U 2295, FF82,15 ;21889          INC LS[ERROR.NUMBER]            ;Go on to next error
U 2296, B718,15 ;21890  T3F.D:  MOV LS[#00040300] TO WR[0]      ;Set up the instruction and size bits
U 2297, C754,15 ;21891          BIS LS[BIT10] TO WR[0]
```

```
U 2298, 90F6.15 ;21892          MISC2 [TRAP.ACC] WR[0]
U 2299, B61A.15 ;21893          MOV LS[T13] TO WR[0]
U 229A, B776.95 ;21894          MOV LS[#8300] TO WR[1]
U 229B, 90F6.15 ;21895          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 229C, 10F6.95 ;21896          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 229D, B61C.15 ;21897          MOV LS[T14] TO WR[0]
U 229E, B716.95 ;21898          MOV LS[#300] TO WR[1]
U 229F, 90F6.15 ;21899          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 22A0, 10F6.95 ;21900          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 22A1, B616.15 ;21901          MOV LS[T11] TO WR[0]
U 22A2, 90F6.15 ;21902          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH 1
U 22A3, B61E.95 ;21903          MOV LS[T15] TO WR[1]
U 22A4, 10F6.95 ;21904          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 22A5, 10F8.15 ;21905          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 22A6, 3A18.15 ;21906          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 22A7, 3618.15 ;21907          MOV LS[T12] TO WR[0]            ;Set up received data
U 22A8, B740.95 ;21908          MOV LS[#302] TO WR[1]           ;Set up expected data
U 22A9, 0869.3C ;21909          JSR [CHECK.RESULT]              ;Check for error
U 22AA, 8A29.64 ;21910          JMP [T3F.D]                     ;Loop on error
U 22AB, FF82.15 ;21911          INC LS[ERROR.NUMBER]            ;Go on to next error
U 22AC, B718.15 ;21912  T3F.E:  MOV LS[#00040300] TO WR[0]      ;Set up instruction and size bits
U 22AD, 475A.15 ;21913          BIS LS[BIT13] TO WR[0]
U 22AE, C758.15 ;21914          BIS LS[BIT12] TO WR[0]
U 22AF, 90F6.15 ;21915          MISC2 [TRAP.ACC] WR[0]
U 22B0, B61A.15 ;21916          MOV LS[T13] TO WR[0]
U 22B1, B716.95 ;21917          MOV LS[#300] TO WR[1]
U 22B2, 90F6.15 ;21918          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP1 SIGN
U 22B3, 10F6.95 ;21919          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 22B4, B61C.15 ;21920          MOV LS[T14] TO WR[0]
U 22B5, B776.95 ;21921          MOV LS[#8300] TO WR[1]
U 22B6, 90F6.15 ;21922          MISC2 [TRAP.ACC] WR[0]          ;CLOCK OP2 SIGN
U 22B7, 10F6.95 ;21923          MISC2 [TRAP.ACC] WR[1]          ;Enable data onto bus
U 22B8, 3614.15 ;21924          MOV LS[T10] TO WR[0]
U 22B9, 90F6.15 ;21925          MISC2 [TRAP.ACC] WR[0]          ;CLOCK SIGN OUT WITH 0
U 22BA, B61E.95 ;21926          MOV LS[T15] TO WR[1]
U 22BB, 10F6.95 ;21927          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 22BC, 10F8.15 ;21928          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 22BD, 3A18.15 ;21929          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 22BE, 3618.15 ;21930          MOV LS[T12] TO WR[0]            ;Set up received data
U 22BF, B716.95 ;21931          MOV LS[#300] TO WR[1]           ;Set up expected data
U 22C0, 0869.3C ;21932          JSR [CHECK.RESULT]              ;Check for error
U 22C1, 8A2A.C4 ;21933          JMP [T3F.E]                     ;Loop on error
U 22C2, FF82.15 ;21934          INC LS[ERROR.NUMBER]            ;Go on to next error
U 22C3, B718.15 ;21935  T3F.F:  MOV LS[#00040300] TO WR[0]      ;Set up instruction and size bits
U 22C4, 475C.15 ;21936          BIS LS[BIT14] TO WR[0]
U 22C5, 90F6.15 ;21937          MISC2 [TRAP.ACC] WR[0]
U 22C6, B61E.95 ;21938          MOV LS[T15] TO WR[1]
U 22C7, 10F6.95 ;21939          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 22C8, 10F8.15 ;21940          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 22C9, 3A18.15 ;21941          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 22CA, 3618.15 ;21942          MOV LS[T12] TO WR[0]            ;Set up received data
U 22CB, B716.95 ;21943          MOV LS[#300] TO WR[1]           ;Set up expected data
U 22CC, 0869.3C ;21944          JSR [CHECK.RESULT]              ;Check for error
U 22CD, 8A2C.34 ;21945          JMP [T3F.F]                     ;Loop on error
U 22CE, FF82.15 ;21946          INC LS[ERROR.NUMBER]            ;Go on to next error
```

```
U 22CF, B718,15 ;21947  T3F.10: MOV LS[#00040300] TO WR[0]      ;Set up instruction and size bits
U 22D0, 475A,15 ;21948          BIS LS[BIT13] TO WR[0]
U 22D1, 90F6,15 ;21949          MISC2 [TRAP.ACC] WR[0]
U 22D2, B61E,95 ;21950          MOV LS[T15] TO WR[1]
U 22D3, 10F6,95 ;21951          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 22D4, 10F8,15 ;21952          MISC2 [READ.ACC.UPC]            ;Enable NUA BUS onto FPA BUS
U 22D5, 3A18,15 ;21953          MOV FPA TO LS[T12]              ;Enable FPA BUS onto Y BUS
U 22D6, 3618,15 ;21954          MOV LS[T12] TO WR[0]            ;Set up received data
U 22D7, B716,95 ;21955          MOV LS[#300] TO WR[1]           ;Set up expected data
U 22D8, 0869,3C ;21956          JSR [CHECK.RESULT]              ;Check for error
U 22D9, 8A2C,F4 ;21957          JMP [T3F.10]                    ;Loop on error
                ;21958  T3F.END:
```

```
;21959  .page
;21960  .TOC    "TEST 40 EXTENDED BRANCH TEST(M8389 FPA MODULE)"
;21961  ;++
;21962  ;
;21963  ;
;21964  ; Test Description:
;21965  ;
;21966  ;       The purpose of this test is to check the extended branch conditions.
;21967  ;       When an extended branch instruction is issued the clock field is set
;21968  ;       to extend branch and the mod field is set to extend clock. The data
;21969  ;       path to test the extended branch instructions will be to issue an
;21970  ;       instruction with the branch field bits set to one of the four exteneded
;21971  ;       branch conditions. An extended branch will cause a 32 way branch since
;21972  ;       there are 5 branch conditions that can be branched on. It will be nec-
;21973  ;       cesary to check to see that all 32 possible branches can be taken. The
;21974  ;       first set of five branch conditions will be tested for all possible
;21975  ;       branches. The remaining branch conditions will only be tested for the
;21976  ;       three extended branch conditions in the asserted and unasserted cases.
;21977  ;
;21978  ; Assumptions:
;21979  ;
;21980  ;       It is assumed that the branch control logic associated with the non-
;21981  ;       extended branch has been tested previously and works.
;21982  ;
;21983  ; Test Steps:
;21984  ;
;21985  ;       1) Set the instruction bits to 00000. Then branch with the branch
;21986  ;          control bits set to 1.
;21987  ;       2) Set the instruction bits to 00001. Then branch with the branch
;21988  ;          control bits set to 1.
;21989  ;       3) Set the instruction bits to 00010. Then branch with the branch
;21990  ;          control bits set to 1.
;21991  ;       4) Set the instruction bits to 00011. Then branch with the branch
;21992  ;          control bits set to 1.
;21993  ;       5) Set the instruction bits to 00100. Then branch with the branch
;21994  ;          control bits set to 1.
;21995  ;       6) Set the instruction bits to 00101. Then branch with the branch
;21996  ;          control bits set to 1.
;21997  ;       7) Set the instruction bits to 00110. Then branch with the branch
;21998  ;          control bits set to 1.
;21999  ;       8) Set the instruction bits to 00111. Then branch with the branch
;22000  ;          control bits set to 1.
;22001  ;       9) Issue a FORCE instruction to set the size bits to 00. Issue a
;22002  ;          nop with the branch field set to 01010.
;22003  ;      10) Issue a FORCE instruction to set the size bits to 11. Issue a
;22004  ;          nop with the branch field set to 01010.
;22005  ;      11) Assert DOUB OPER and branch with the branch control bits set to
;22006  ;          15. Assert DOUB OPER with Instruction bits set to 00000.
;22007  ;      12) Assert DOUB OPER and branch with the branch control bits set to
;22008  ;          15. Assert DOUB OPER with the instruction bits set to 11000.
;22009  ;      13) De-assert DOUB OPER and branch with the branch control bits set to
;22010  ;          15, and the instruction bits set to 10000.
;22011  ;      14) De-assert DOUB OPER and branch with the branch control bits set to
;22012  ;          15, and the instruction bits set to 01000.
;22013  ;      15) Set the instruction bits to 00111 and the size bits to 00. Then
```

```
;22014  ;          branch with the branch control bits set to 1B.
;22015  ;       16) Set the instruction bits to 00000 and the size bits to 00. Then
;22016  ;          branch with the branch control bits set to 1B.
;22017  ;
;22018  ; Error:
;22019  ;
;22020  ;       Error1 - Branch failed when the branch control bits were 00 and the
;22021  ;                OR bits should be 00000.
;22022  ;       Error2 - Branch failed when the branch control bits were 00 and the
;22023  ;                OR bits should be 00100.
;22024  ;       Error3 - Branch failed when the branch control bits were 00 and the
;22025  ;                OR bits should be 01000.
;22026  ;       Error4 - Branch failed when the branch control bits were 00 and the
;22027  ;                OR bits should be 01100.
;22028  ;       Error5 - Branch failed when the branch control bits were 00 and the
;22029  ;                OR bits should be 10000.
;22030  ;       Error6 - Branch failed when the branch control bits were 00 and the
;22031  ;                OR bits should be 10100.
;22032  ;       Error7 - Branch failed when the branch control bits were 00 and the
;22033  ;                OR bits should be 11000.
;22034  ;       Error8 - Branch failed when the branch control bits were 00 and the
;22035  ;                OR bits should be 11100.
;22036  ;       Error9 - Branch failed when the branch control bits were 01 and the
;22037  ;                OR bits should be 00000.
;22038  ;       ErrorA - Branch failed when the branch control bits were 01 and the
;22039  ;                OR bits should be 00000.
;22040  ;       ErrorB - Branch failed when the branch control bits were 10 and the
;22041  ;                OR bits should be 10000 and the instruction bits were 00000.
;22042  ;       ErrorC - Branch failed when the branch control bits were 10 and the
;22043  ;                OR bits should be 10000 and the instruction bits were 11000.
;22044  ;       ErrorD - Branch failed when the branch control bits were 10 and the
;22045  ;                OR bits should be 00000 and the instruction bits were 10000.
;22046  ;       ErrorE - Branch failed when the branch control bits were 10 and the
;22047  ;                OR bits should be 00000 and the instruction bits were 01000.
;22048  ;       ErrorF - Branch failed when the branch control bits were 11 and the
;22049  ;                OR bits should be 11100.
;22050  ;       Error10 - Branch failed when the branch control bits were 11 and the
;22051  ;                 OR bits should be 00000.
;22052  ;
;22053  ; Debug:
;22054  ;
;22055  ;       Error1: If this error occurs then the OR inputs to the microsequencer
;22056  ;               should be checked for error. If the OR inputs are not in error
;22057  ;               then the EXT BRANCH PAL should be checked for error. If the
;22058  ;               PAL is not the cause of the error then EXT BRANCH and EXT CLOCK
;22059  ;               should be checked to be asserted and UBCTL bits and the branch
;22060  ;               conditions should be checked for error. If EXT BRANCH is in
;22061  ;               error then the CS Latch that it comes from should be checked
;22062  ;               for error. If the latch is not in error then the gate before it
;22063  ;               should be checked for error.
;22064  ;       Error2: Same as error 1.
;22065  ;       Error3: Same as error 1.
;22066  ;       Error4: Same as error 1.
;22067  ;       Error5: Same as error 1.
;22068  ;       Error6: Same as error 1.
```

```
                ;22069  ;       Error7: Same as error 1.
                ;22070  ;       Error8: Same as error 1.
                ;22071  ;       Error9: Same as error 1.
                ;22072  ;       ErrorA: Same as error 1.
                ;22073  ;       ErrorB: Same as error 1.
                ;22074  ;       ErrorC: Same as error 1.
                ;22075  ;       ErrorD: Same as error 1.
                ;22076  ;       ErrorE: Same as error 1.
                ;22077  ;       ErrorF: Same as error 1.
                ;22078  ;       Error10: Same as error 1.
                ;22079  ;
                ;22080  ;
                ;22081  ;--
                ;22082  T.40:
U 22DA, B65E,15 ;22083          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;22084                                          ; STATUS WORD
U 22DB, 3E80,15 ;22085          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;22086                                          ; BIT 15 INDICATES START OF TEST TO
                ;22087                                          ; CONSOLE
U 22DC, 10E0,15 ;22088          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;22089  WAIT.T40.0:
U 22DD, 8A2D,D4 ;22090          JMP [WAIT.T40.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;22091                                          ; RESPOND
U 22DE, 887E,EC ;22092          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;22093                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 22DF, 3798,15 ;22094          MOV LS[T40.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 22E0, BE12,15 ;22095          MOV WR[0] TO LS[T9]
U 22E1, B72E,15 ;22096          MOV LS[#2A7] TO WR[0]           ;Get address of LOAD ET0
U 22E2, BE14,15 ;22097          MOV WR[0] TO LS[T10]
U 22E3, 8870,3C ;22098          JSR [L0]                        ;Get EXT BRNCH on INSTR BITS 2-0
U 22E4, 3E10,15 ;22099          MOV WR[0] TO LS[T8]
U 22E5, 8870,3C ;22100          JSR [L0]                        ;Get EXT BRNCH on SIZE BITS
U 22E6, 3E16,15 ;22101          MOV WR[0] TO LS[T11]
U 22E7, 8870,3C ;22102          JSR [L0]                        ;Get EXT BRNCH on DOUB OPER
U 22E8, BE13,15 ;22103          MOV WR[0] TO LS[T12]
U 22E9, 8870,3C ;22104          JSR [L0]                        ;Get EXT BRNCH on INSTR BITS 2-0
U 22EA, 3E1A,15 ;22105          MOV WR[0] TO LS[T13]
U 22EB, 8870,3C ;22106          JSR [L0]                        ;Get address of MOV ET0 TO ET2
U 22EC, 3E1C,15 ;22107          MOV WR[0] TO LS[T14]
U 22ED, 8870,3C ;22108          JSR [L0]                        ;Get address of ADD ET0 TO ET2
U 22EE, BE1E,15 ;22109          MOV WR[0] TO LS[T15]
U 22EF, 8870,3C ;22110          JSR [L0]                        ;Get address of MOV FWR[FT0] TO FQ
U 22F0, 3EAE,15 ;22111          MOV WR[0] TO LS[T57]
U 22F1, 8870,3C ;22112          JSR [L0]                        ;Get address of Shift left FQ
U 22F2, 3EB0,15 ;22113          MOV WR[0] TO LS[T58]
U 22F3, B718,15 ;22114  T40.1:  MOV LS[#00040300] TO WR[0]      ;Set the instruction and size bits
U 22F4, 90F6,15 ;22115          MISC2 [TRAP.ACC] WR[0]          ; to 0
U 22F5, E5A2,15 ;22116          CLR LS[DATA.1]
U 22F6, 8877,DC ;22117          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 22F7, 0878,AC ;22118          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 22F8, 361E,15 ;22119          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 22F9, 3610,95 ;22120          MOV LS[T8] TO WR[1]             ;Setup to force EXT BRANCH on INSTR BIT
U 22FA, 90F6,15 ;22121          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 22FB, 10F6,95 ;22122          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 22FC, 10F8,15 ;22123          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
```

```
U 22FD, 3B08,15 ;22124          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 22FE, 3708,15 ;22125          MOV LS[T84] TO WR[0]            ;Set up received data
U 22FF, B716,95 ;22126          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2300, 0869,3C ;22127          JSR [CHECK.RESULT]              ;Check for error
U 2301, 8A2F,34 ;22128          JMP [T40.1]                     ;Loop on error
U 2302, FF82,15 ;22129          INC LS[ERROR.NUMBER]            ;Error2
U 2303, B718,15 ;22130  T40.2:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 00001
U 2304, C754,15 ;22131          BIS LS[BIT10] TO WR[0]
U 2305, 90F6,15 ;22132          MISC2 [TRAP.ACC] WR[0]          ; and the size bits to 0
U 2306, E5A2,15 ;22133          CLR LS[DATA.1]
U 2307, 8877,DC ;22134          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 2308, 0878,AC ;22135          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 2309, 361E,15 ;22136          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 230A, 3610,95 ;22137          MOV LS[T8] TO WR[1]             ;Setup to force EXT BRANCH on INSTR BIT
U 230B, 90F6,15 ;22138          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 230C, 10F6,95 ;22139          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 230D, 10F8,15 ;22140          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 230E, 3B08,15 ;22141          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 230F, 3708,15 ;22142          MOV LS[T84] TO WR[0]            ;Set up received data
U 2310, B716,95 ;22143          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2311, C744,95 ;22144          BIS LS[#4] TO WR[1]
U 2312, 0869,3C ;22145          JSR [CHECK.RESULT]              ;Check for error
U 2313, 0A30,34 ;22146          JMP [T40.2]                     ;Loop on error
U 2314, FF82,15 ;22147          INC LS[ERROR.NUMBER]            ;Error3
U 2315, B718,15 ;22148  T40.3:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 00010
U 2316, 4756,15 ;22149          BIS LS[BIT11] TO WR[0]
U 2317, 90F6,15 ;22150          MISC2 [TRAP.ACC] WR[0]          ; and the size bits to 0
U 2318, E5A2,15 ;22151          CLR LS[DATA.1]
U 2319, 8877,DC ;22152          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 231A, 0878,AC ;22153          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 231B, 361E,15 ;22154          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 231C, 3610,95 ;22155          MOV LS[T8] TO WR[1]             ;Setup to force EXT BRANCH on INSTR BIT
U 231D, 90F6,15 ;22156          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 231E, 10F6,95 ;22157          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 231F, 10F8,15 ;22158          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2320, 3B08,15 ;22159          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 2321, 3708,15 ;22160          MOV LS[T84] TO WR[0]            ;Set up received data
U 2322, B716,95 ;22161          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2323, 4746,95 ;22162          BIS LS[BIT3] TO WR[1]
U 2324, 0869,3C ;22163          JSR [CHECK.RESULT]              ;Check for error
U 2325, 8A31,54 ;22164          JMP [T40.3]                     ;Loop on error
U 2326, FF82,15 ;22165          INC LS[ERROR.NUMBER]            ;Error4
U 2327, B718,15 ;22166  T40.4:  MOV LS[#0004030C] TO WR[0]      ;Set the instruction bits to 00011
U 2328, C754,15 ;22167          BIS LS[BIT10] TO WR[0]
U 2329, 4756,15 ;22168          BIS LS[BIT11] TO WR[0]
U 232A, 90F6,15 ;22169          MISC2 [TRAP.ACC] WR[0]          ; and the size bits to 0
U 232B, E5A2,15 ;22170          CLR LS[DATA.1]
U 232C, 8877,DC ;22171          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 232D, 0878,AC ;22172          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 232E, 361E,15 ;22173          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 232F, 3610,95 ;22174          MOV LS[T8] TO WR[1]             ;Setup to force EXT BRANCH on INSTR BIT
U 2330, 90F6,15 ;22175          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 2331, 10F6,95 ;22176          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 2332, 10F8,15 ;22177          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2333, 3B08,15 ;22178          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
```

```
U 2334, 3708.15 ;22179          MOV LS[T84] TO WR[0]          ;Set up received data
U 2335, B716.95 ;22180          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2336, C744.95 ;22181          BIS LS[BIT2] TO WR[1]
U 2337, 4746.95 ;22182          BIS LS[BIT3] TO WR[1]
U 2338, 0869.3C ;22183          JSR [CHECK.RESULT]            ;Check for error
U 2339, 0A32.74 ;22184          JMP [T40.4]                   ;Loop on error
U 233A, FF82.15 ;22185          INC LS[ERROR.NUMBER]          ;Error5
U 233B, B718.15 ;22186  T40.5:  MOV LS[#00040300] TO WR[0]    ;Set the instruction bits to 00100
U 233C, C758.15 ;22187          BIS LS[BIT12] TO WR[0]        ; and the size bits to 0
U 233D, 90F6.15 ;22188          MISC2 [TRAP.ACC] WR[0]
U 233E, E5A2.15 ;22189          CLR LS[DATA.1]
U 233F, 8877.DC ;22190          JSR [LOAD.DATA]               ;Load ET0 with zeros
U 2340, 0878.AC ;22191          JSR [MOV.DATA]                ;Load ET2 with zeros
U 2341, 361E.15 ;22192          MOV LS[T15] TO WR[0]          ;Set up for ADD ET0 TO ET2
U 2342, 3610.95 ;22193          MOV LS[T8] TO WR[1]           ;Setup to force EXT BRANCH on INSTR BIT
U 2343, 90F6.15 ;22194          MISC2 [TRAP.ACC] WR[0]        ;Get address of ADD ET0 TO ET2
U 2344, 10F6.95 ;22195          MISC2 [TRAP.ACC] WR[1]        ;Force EXT BRANCH on INSTR BITS
U 2345, 10F8.15 ;22196          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 2346, 3B08.15 ;22197          MOV FPA TO LS[T84]            ;Enable the FPA BUS onto the Y BUS
U 2347, 3708.15 ;22198          MOV LS[T84] TO WR[0]          ;Set up received data
U 2348, B716.95 ;22199          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2349, C748.95 ;22200          BIS LS[BIT4] TO WR[1]
U 234A, 0869.3C ;22201          JSR [CHECK.RESULT]            ;Check for error
U 234B, 8A33.B4 ;22202          JMP [T40.5]                   ;Loop on error
U 234C, FF82.15 ;22203          INC LS[ERROR.NUMBER]          ;Error6
U 234D, B718.15 ;22204  T40.6:  MOV LS[#00040300] TO WR[0]    ;Set the instruction bits to 00101
U 234E, C758.15 ;22205          BIS LS[BIT12] TO WR[0]        ; and the size bits to 0
U 234F, C754.15 ;22206          BIS LS[BIT10] TO WR[0]
U 2350, 90F6.15 ;22207          MISC2 [TRAP.ACC] WR[0]
U 2351, E5A2.15 ;22208          CLR LS[DATA.1]
U 2352, 8877.DC ;22209          JSR [LOAD.DATA]               ;Load ET0 with zeros
U 2353, 0878.AC ;22210          JSR [MOV.DATA]                ;Load ET2 with zeros
U 2354, 361E.15 ;22211          MOV LS[T15] TO WR[0]          ;Set up for ADD ET0 TO ET2
U 2355, 3610.95 ;22212          MOV LS[T8] TO WR[1]           ;Setup to force EXT BRANCH on INSTR BIT
U 2356, 90F6.15 ;22213          MISC2 [TRAP.ACC] WR[0]        ;Get address of ADD ET0 TO ET2
U 2357, 10F6.95 ;22214          MISC2 [TRAP.ACC] WR[1]        ;Force EXT BRANCH on INSTR BITS
U 2358, 10F8.15 ;22215          MISC2 [READ.ACC.UPC]          ;Enable the NUA BUS onto the FPA BUS
U 2359, 3B08.15 ;22216          MOV FPA TO LS[T84]            ;Enable the FPA BUS onto the Y BUS
U 235A, 3708.15 ;22217          MOV LS[T84] TO WR[0]          ;Set up received data
U 235B, B716.95 ;22218          MOV LS[#300] TO WR[1]         ;Set up expected data
U 235C, C748.95 ;22219          BIS LS[BIT4] TO WR[1]
U 235D, C744.95 ;22220          BIS LS[BIT2] TO WR[1]
U 235E, 0869.3C ;22221          JSR [CHECK.RESULT]            ;Check for error
U 235F, 0A34.D4 ;22222          JMP [T40.6]                   ;Loop on error
U 2360, FF82.15 ;22223          INC LS[ERROR.NUMBER]          ;Error7
U 2361, B718.15 ;22224  T40.7:  MOV LS[#00040300] TO WR[0]    ;Set the instruction bits to 00110
U 2362, C758.15 ;22225          BIS LS[BIT12] TO WR[0]        ; and the size bits to 0
U 2363, 4756.15 ;22226          BIS LS[BIT11] TO WR[0]
U 2364, 90F6.15 ;22227          MISC2 [TRAP.ACC] WR[0]
U 2365, E5A2.15 ;22228          CLR LS[DATA.1]
U 2366, 8877.DC ;22229          JSR [LOAD.DATA]               ;Load ET0 with zeros
U 2367, 0878.AC ;22230          JSR [MOV.DATA]                ;Load ET2 with zeros
U 2368, 361E.15 ;22231          MOV LS[T15] TO WR[0]          ;Set up for ADD ET0 TO ET2
U 2369, 3610.95 ;22232          MOV LS[T8] TO WR[1]           ;Setup to force EXT BRANCH on INSTR BIT
U 236A, 90F6.15 ;22233          MISC2 [TRAP.ACC] WR[0]        ;Get address of ADD ET0 TO ET2
```

```
U 236B, 10F6,95 ;22234          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 236C, 10F8,15 ;22235          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 236D, 3B08,15 ;22236          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 236E, 3708,15 ;22237          MOV LS[T84] TO WR[0]            ;Set up received data
U 236F, B716,95 ;22238          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2370, 4746,95 ;22239          BIS LS[BIT3] TO WR[1]
U 2371, C748,95 ;22240          BIS LS[BIT4] TO WR[1]
U 2372, 0869,3C ;22241          JSR [CHECK.RESULT]              ;Check for error
U 2373, 8A36,14 ;22242          JMP [T40.7]                     ;Loop on error
U 2374, FF82,15 ;22243          INC LS[ERROR.NUMBER]            ;Error8
U 2375, B718,15 ;22244  T40.8:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 00111
U 2376, C758,15 ;22245          BIS LS[BIT12] TO WR[0]          ; and the size bits to 0
U 2377, C754,15 ;22246          BIS LS[BIT10] TO WR[0]
U 2378, 4756,15 ;22247          BIS LS[BIT11] TO WR[0]
U 2379, 90F6,15 ;22248          MISC2 [TRAP.ACC] WR[0]
U 237A, E5A2,15 ;22249          CLR LS[DATA.1]
U 237B, 8877,DC ;22250          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 237C, 0878,AC ;22251          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 237D, 361E,15 ;22252          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 237E, 3610,95 ;22253          MOV LS[T8] TO WR[1]             ;Setup to force EXT BRANCH on INSTR BIT
U 237F, 90F6,15 ;22254          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 2380, 10F6,95 ;22255          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 2381, 10F8,15 ;22256          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2382, 3B08,15 ;22257          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 2383, 3708,15 ;22258          MOV LS[T84] TO WR[0]            ;Set up received data
U 2384, B716,95 ;22259          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2385, C744,95 ;22260          BIS LS[BIT2] TO WR[1]
U 2386, 4746,95 ;22261          BIS LS[BIT3] TO WR[1]
U 2387, C748,95 ;22262          BIS LS[BIT4] TO WR[1]
U 2388, 0869,3C ;22263          JSR [CHECK.RESULT]              ;Check for error
U 2389, 8A37,54 ;22264          JMP [T40.8]                     ;Loop on error
U 238A, FF82,15 ;22265          INC LS[ERROR.NUMBER]            ;Error9
U 238B, B718,15 ;22266  T40.9:  MOV LS[#00040300] TO WR[0]      ;Set the instruction and size bits to 0
U 238C, 90F6,15 ;22267          MISC2 [TRAP.ACC] WR[0]
U 238D, E5A2,15 ;22268          CLR LS[DATA.1]
U 238E, 8877,DC ;22269          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 238F, 0878,AC ;22270          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 2390, 361E,15 ;22271          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 2391, 3616,95 ;22272          MOV LS[T11] TO WR[1]            ;Setup to force EXT BRANCH on INSTR BIT
U 2392, 90F6,15 ;22273          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 2393, 10F6,95 ;22274          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 2394, 10F8,15 ;22275          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2395, 3B08,15 ;22276          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 2396, 3708,15 ;22277          MOV LS[T84] TO WR[0]            ;Set received data
U 2397, B716,95 ;22278          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2398, 0869,3C ;22279          JSR [CHECK.RESULT]              ;Check for error
U 2399, 0A38,B4 ;22280          JMP [T40.9]                     ;Loop on error
U 239A, FF82,15 ;22281          INC LS[ERROR.NUMBER]            ;ErrorA
U 239B, B748,15 ;22282  T40.A:  MOV LS[#70300] TO WR[0]         ;Set the instruction bits to 0 and the
U 239C, 90F6,15 ;22283          MISC2 [TRAP.ACC] WR[0]          ; and the size bits to 11
U 239D, E5A2,15 ;22284          CLR LS[DATA.1]
U 239E, 8877,DC ;22285          JSR [LOAD.DATA]                 ;Load ET0 with zeros
U 239F, 0878,AC ;22286          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 23A0, 361E,15 ;22287          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 23A1, 3616,95 ;22288          MOV LS[T11] TO WR[1]            ;Setup to force EXT BRANCH on INSTR BIT
```

```
U 23A2, 90F6,15 ;22289          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 23A3, 10F6,95 ;22290          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 23A4, 10F8,15 ;22291          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 23A5, 3B08,15 ;22292          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 23A6, 3708,15 ;22293          MOV LS[T84] TO WR[0]            ;Set received data
U 23A7, B716,95 ;22294          MOV LS[#300] TO WR[1]           ;Set up expected data
U 23A8, 4746,95 ;22295          BIS LS[BIT3] TO WR[1]
U 23A9, C748,95 ;22296          BIS LS[BIT4] TO WR[1]
U 23AA, 0869,3C ;22297          JSR [CHECK.RESULT]              ;Check for error
U 23AB, 8A39,B4 ;22298          JMP [T40.A]                     ;Loop on error
U 23AC, FF82,15 ;22299          INC LS[ERROR.NUMBER]            ;ErrorB
U 23AD, B718,15 ;22300  T40.B:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 00000
U 23AE, 90F6,15 ;22301          MISC2 [TRAP.ACC] WR[0]
U 23AF, E5A2,15 ;22302          CLR LS[DATA.1]                  ;Load FWR[FT0] with zeros
U 23B0, 8877,DC ;22303          JSR [LOAD.DATA]
U 23B1, B6AE,15 ;22304          MOV LS[T57] TO WR[0]            ;MOV FWR[FT0] TO FQ
U 23B2, 90F6,15 ;22305          MISC2 [TRAP.ACC] WR[0]          ;Force the MOV
U 23B3, B6B0,15 ;22306          MOV LS[T58] TO WR[0]            ;Set up to force the shift
U 23B4, B618,95 ;22307          MOV LS[T12] TO WR[1]            ;Set up to force the branch
U 23B5, 90F6,15 ;22308          MISC2 [TRAP.ACC] WR[0]          ;Force the shift
U 23B6, 10F6,95 ;22309          MISC2 [TRAP.ACC] WR[1]          ;Force the branch
U 23B7, 10F8,15 ;22310          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 23B8, 3B08,15 ;22311          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 23B9, 3708,15 ;22312          MOV LS[T84] TO WR[0]            ;Set received data
U 23BA, B716,95 ;22313          MOV LS[#300] TO WR[1]           ;Set up expected data
U 23BB, C748,95 ;22314          BIS LS[BIT4] TO WR[1]
U 23BC, 0869,3C ;22315          JSR [CHECK.RESULT]              ;Check for error
U 23BD, 8A3A,D4 ;22316          JMP [T40.B]                     ;Loop on error
U 23BE, FF82,15 ;22317          INC LS[ERROR.NUMBER]            ;ErrorC
U 23BF, B718,15 ;22318  T40.C:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 11000
U 23C0, 475C,15 ;22319          BIS LS[BIT14] TO WR[0]          ; and the size bits to 0
U 23C1, 475A,15 ;22320          BIS LS[BIT13] TO WR[0]
U 23C2, 90F6,15 ;22321          MISC2 [TRAP.ACC] WR[0]
U 23C3, E5A2,15 ;22322          CLR LS[DATA.1]                  ;Load FWR[FT0] with zeros
U 23C4, 8877,DC ;22323          JSR [LOAD.DATA]
U 23C5, B6AE,15 ;22324          MOV LS[T57] TO WR[0]            ;MOV FWR[FT0] TO FQ
U 23C6, 90F6,15 ;22325          MISC2 [TRAP.ACC] WR[0]          ;Force the MOV
U 23C7, B6B0,15 ;22326          MOV LS[T58] TO WR[0]            ;Set up to force the shift
U 23C8, B618,95 ;22327          MOV LS[T12] TO WR[1]            ;Set up to force the branch
U 23C9, 90F6,15 ;22328          MISC2 [TRAP.ACC] WR[0]          ;Force the shift
U 23CA, 10F6,95 ;22329          MISC2 [TRAP.ACC] WR[1]          ;Force the branch
U 23CB, 10F8,15 ;22330          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 23CC, 3B08,15 ;22331          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 23CD, 3708,15 ;22332          MOV LS[T84] TO WR[0]            ;Set received data
U 23CE, B716,95 ;22333          MOV LS[#300] TO WR[1]           ;Set up expected data
U 23CF, C748,95 ;22334          BIS LS[BIT4] TO WR[1]
U 23D0, 0869,3C ;22335          JSR [CHECK.RESULT]              ;Check for error
U 23D1, 8A3B,F4 ;22336          JMP [T40.C]                     ;Loop on error
U 23D2, FF82,15 ;22337          INC LS[ERROR.NUMBER]            ;ErrorD
U 23D3, B718,15 ;22338  T40.D:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 10000
U 23D4, 475C,15 ;22339          BIS LS[BIT14] TO WR[0]          ; and the size bits to 0
U 23D5, 90F6,15 ;22340          MISC2 [TRAP.ACC] WR[0]
U 23D6, E5A2,15 ;22341          CLR LS[DATA.1]                  ;Load FWR[FT0] with zeros
U 23D7, 8877,DC ;22342          JSR [LOAD.DATA]
U 23D8, B6AE,15 ;22343          MOV LS[T57] TO WR[0]            ;MOV FWR[FT0] TO FQ
```

```
U 23D9, 90F6,15 ;22344          MISC2 [TRAP.ACC] WR[0]          ;Force the MOV
U 23DA, B6B0,15 ;22345          MOV LS[T58] TO WR[0]            ;Set up to force the shift
U 23DB, B618,95 ;22346          MOV LS[T12] TO WR[1]            ;Set up to force the branch
U 23DC, 90F6,15 ;22347          MISC2 [TRAP.ACC] WR[0]          ;Force the shift
U 23DD, 10F6,95 ;22348          MISC2 [TRAP.ACC] WR[1]          ;Force the branch
U 23DE, 10F8,15 ;22349          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 23DF, 3B08,15 ;22350          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 23E0, 3708,15 ;22351          MOV LS[T84] TO WR[0]            ;Set received data
U 23E1, B716,95 ;22352          MOV LS[#300] TO WR[1]           ;Set up expected data
U 23E2, 0869,3C ;22353          JSR [CHECK.RESULT]              ;Check for error
U 23E3, 8A3D,34 ;22354          JMP [T40.D]                     ;Loop on error
U 23E4, FF82,15 ;22355          INC LS[ERROR.NUMBER]            ;ErrorE
U 23E5, B718,15 ;22356  T40.E:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 01000
U 23E6, 475A,15 ;22357          BIS LS[BIT13] TO WR[0]
U 23E7, 90F6,15 ;22358          MISC2 [TRAP.ACC] WR[0]
U 23E8, E5A2,15 ;22359          CLR LS[DATA.1]                  ;Load FWR[FT0] with zeros
U 23E9, 8877,DC ;22360          JSR [LOAD.DATA]
U 23EA, B6AE,15 ;22361          MOV LS[T57] TO WR[0]            ;MOV FWR[FT0] TO FQ
U 23EB, 90F6,15 ;22362          MISC2 [TRAP.ACC] WR[0]          ;Force the MOV
U 23EC, B6B0,15 ;22363          MOV LS[T58] TO WR[0]            ;Set up to force the shift
U 23ED, B618,95 ;22364          MOV LS[T12] TO WR[1]            ;Set up to force the branch
U 23EE, 90F6,15 ;22365          MISC2 [TRAP.ACC] WR[0]          ;Force the shift
U 23EF, 10F6,95 ;22366          MISC2 [TRAP.ACC] WR[1]          ;Force the branch
U 23F0, 10F8,15 ;22367          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 23F1, 3B08,15 ;22368          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 23F2, 3708,15 ;22369          MOV LS[T84] TO WR[0]            ;Set received data
U 23F3, B716,95 ;22370          MOV LS[#300] TO WR[1]           ;Set up expected data
U 23F4, 0869,3C ;22371          JSR [CHECK.RESULT]              ;Check for error
U 23F5, 8A3E,54 ;22372          JMP [T40.E]                     ;Loop on error
U 23F6, FF82,15 ;22373          INC LS[ERROR.NUMBER]            ;ErrorF
U 23F7, B718,15 ;22374  T40.F:  MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 00000
U 23F8, 90F6,15 ;22375          MISC2 [TRAP.ACC] WR[0]          ; and the size bits to 0
U 23F9, B654,15 ;22376          MOV LS[BIT10] TO WR[0]
U 23FA, 3EA2,15 ;22377          MOV WR[0] TO LS[DATA.1]
U 23FB, 8877,DC ;22378          JSR [LOAD.DATA]                 ;Load ET0 with one
U 23FC, 0878,AC ;22379          JSR [MOV.DATA]                  ;Load ET2 with one
U 23FD, 361E,15 ;22380          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 23FE, 361A,95 ;22381          MOV LS[T13] TO WR[1]            ;Setup to force EXT BRANCH on INSTR BIT
U 23FF, 90F6,15 ;22382          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 2400, 10F6,95 ;22383          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 2401, 10F8,15 ;22384          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2402, 3B08,15 ;22385          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 2403, 3708,15 ;22386          MOV LS[T84] TO WR[0]            ;Set received data
U 2404, B716,95 ;22387          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2405, 0869,3C ;22388          JSR [CHECK.RESULT]              ;Check for error
U 2406, 8A3F,74 ;22389          JMP [T40.F]                     ;Loop on error
U 2407, FF82,15 ;22390          INC LS[ERROR.NUMBER]            ;Error10
U 2408, B718,15 ;22391  T40.10: MOV LS[#00040300] TO WR[0]      ;Set the instruction bits to 11100
U 2409, C758,15 ;22392          BIS LS[BIT12] TO WR[0]          ; and the size bits to 0
U 240A, 4756,15 ;22393          BIS LS[BIT11] TO WR[0]
U 240B, C754,15 ;22394          BIS LS[BIT10] TO WR[0]
U 240C, 90F6,15 ;22395          MISC2 [TRAP.ACC] WR[0]
U 240D, B654,15 ;22396          MOV LS[BIT10] TO WR[0]
U 240E, 3EA2,15 ;22397          MOV WR[0] TO LS[DATA.1]
U 240F, 8877,DC ;22398          JSR [LOAD.DATA]                 ;Load ET0 with zeros
```

```
U 2410, 0878,AC ;22399          JSR [MOV.DATA]                  ;Load ET2 with zeros
U 2411, 361E,15 ;22400          MOV LS[T15] TO WR[0]            ;Set up for ADD ET0 TO ET2
U 2412, 361A,95 ;22401          MOV LS[T13] TO WR[1]            ;Setup to force EXT BRANCH on INSTR BIT
U 2413, 90F6,15 ;22402          MISC2 [TRAP.ACC] WR[0]          ;Get address of ADD ET0 TO ET2
U 2414, 10F6,95 ;22403          MISC2 [TRAP.ACC] WR[1]          ;Force EXT BRANCH on INSTR BITS
U 2415, 10F8,15 ;22404          MISC2 [READ.ACC.UPC]            ;Enable the NUA BUS onto the FPA BUS
U 2416, 3B08,15 ;22405          MOV FPA TO LS[T84]              ;Enable the FPA BUS onto the Y BUS
U 2417, 3708,15 ;22406          MOV LS[T84] TO WR[0]            ;Set received data
U 2418, B716,95 ;22407          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2419, C748,95 ;22408          BIS LS[BIT4] TO WR[1]
U 241A, 4746,95 ;22409          BIS LS[BIT3] TO WR[1]
U 241B, C744,95 ;22410          BIS LS[BIT2] TO WR[1]
U 241C, 0869,3C ;22411          JSR [CHECK.RESULT]              ;Check for error
U 241D, 0A40,84 ;22412          JMP [T40.10]                    ;Loop on error
                ;22413  T40.END:
```

```
;22414  .PAGE
;22415  .TOC    "TEST 41 MUL I1 AND FRAC55 Q3 BRANCH TEST(M8389 FPA MODULE)"
;22416  ;++
;22417  ;
;22418  ;
;22419  ; Test Description:
;22420  ;
;22421  ;       The purpose of this test is to use the MUL branch condition to check
;22422  ;       some of the multiply logic with the branch condition and the branch
;22423  ;       condition itself. This will be done by setting up the size, the clock,
;22424  ;       the instruction bits, and the shift out of the Q register. There are
;22425  ;       9 of these cases that will be tested. Then there will be a case to
;22426  ;       check the other branch condition.
;22427  ;
;22428  ; Assumptions:
;22429  ;
;22430  ;       It is assumed that all of the 2901's have been tested and are working.
;22431  ;
;22432  ; Test Steps:
;22433  ;
;22434  ;       1) Set the size bits and the instruction bits to 0. Load a one into Q32
;22435  ;          shift FQ right and branch. If there isn't any branch then issue
;22436  ;          error 1. Repeat this procedure with a zero in Q32. If there is any
;22437  ;          branch then issue error 1.
;22438  ;       2) Load a one into Q16 and a zero into Q32. TOGGLE Q16 DEFAULT, shift
;22439  ;          FQ right and branch. If there isn't any branch then issue error 2.
;22440  ;          Repeat the procedure with Q16 set to zero and Q32 set to one. Tf
;22441  ;          there is any branch then issue error 2.
;22442  ;       3) Set the instruction bits to integer. Load a zero into Q32 and a one
;22443  ;          into Q16, shift FQ right, and branch. If there isn't a branch then
;22444  ;          issue error 3. Repeat this with a one in Q32 and a zero in Q16. If
;22445  ;          there is a branch then issue error 3.
;22446  ;       4) Set the instruction bits to emod. Load a zero into bit Q32 and a one
;22447  ;          into Q16, shift right and branch. If there is a branch then issue
;22448  ;          error 4. Repeat the procedure with a one in bit Q32 and a zero in
;22449  ;       5) Set the size bits to double. Load a zero into bit Q00 and a one
;22450  ;          into Q16, shift right, and branch. If there is a branch then issue
;22451  ;          error 5. Repeat the procedure with a zero in bit Q16 and a one in
;22452  ;          bit Q00. If there isn't a branch then issue error 5.
;22453  ;       6) Load a zero into Q00 and a one into Q16. Toggle Q16 default, sh
;22454  ;          Repeat the procedure with a one in Q00 and a zero in Q16. If there
;22455  ;          is a branch then issue error 6.
;22456  ;       7) Set the size bits to grand. Load a zero into bit Q00 and a one into
;22457  ;          bit Q16, shift right, and branch. If there is a branch then issue
;22458  ;          error 7. Repeat the procedure with a zero in Q16 and a one in Q00.
;22459  ;          If there isn't a branch then issue error 7.
;22460  ;       8) Load a zero into bit Q00 and a one into bit Q16. Toggle Q16 default,
;22461  ;          shift right, and branch. If there isn't any branch then issue
;22462  ;          error 8. Repeat the procedure with a one in bit Q00 and a zero in
;22463  ;          Q16. If there is a branch then issue error 8.
;22464  ;       9) Set the size bits to huge. Load a zero into bit EXTQ00 and a one
;22465  ;          into bit Q16, shift right, and branch. If there is a branch then
;22466  ;          issue error 9. Repeat the procedure with a zero in Q16 and a one in
;22467  ;          EXTQ00. If there isn't a branch then issue error 9.
;22468  ;      10) Load a zero into bit EXTQ00 and a one into bit Q16. Toggle Q16
```

```
;22469  ;          default,shift right, and branch. If there isn't any branch then
;22470  ;          issue error 10. Repeat the procedure with a one in bit EXTQ00 and a
;22471  ;          zero in Q16. If there is a branch then issue error 10.
;22472  ;       11) Load a one into bit Q55, shift left, and branch. If there isn't a
;22473  ;          branch then issue error 11.
;22474  ;
;22475  ; Error:
;22476  ;
;22477  ;       Error1 - MULI1 branch failed when the size bit were floating and Q16
;22478  ;                DEFAULT, INTEGER, and EMOD were unasserted.
;22479  ;       Error2 - MULI1 branch failed when the size btis were floating and Q16
;22480  ;                DEFAULT was asserted and INTEGER and EMOD were unasserted.
;22481  ;       Error3 - MULI1 branch failed when then size bits were floating and
;22482  ;                INTEGER was asserted and Q16 DEFAULT and EMOD were unasserted.
;22483  ;       Error4 - MULI1 branch failed when the size bits were floating and Q16
;22484  ;                DEAFULT and INTEGER were unasserted and EMOD were asserted.
;22485  ;       Error5 - MULI1 branch failed when the size bits were double and Q16
;22486  ;                DEFAULT is unasserted.
;22487  ;       Error6 - MULI1 branch failed when the size bits were double and Q16
;22488  ;                DEFAULT is asserted.
;22489  ;       Error7 - MULI1 branch failed when the size bits were grand and Q16
;22490  ;                DEFAULT is unasserted.
;22491  ;       Error8 - MULI1 branch failed when the size bits were grand and Q16
;22492  ;                DEFAULT is asserted.
;22493  ;       Error9 - MULI1 branch failed when the size bits were huge and Q16
;22494  ;                DEFAULT is unasserted.
;22495  ;       ErrorA - MULI1 branch failed when the size bits were huge and Q16
;22496  ;                 DEFAULT is asserted.
;22497  ;       ErrorB - FRAC55 Q3 failed when Q55 was asserted.
;22498  ;
;22499  ; Debug:
;22500  ;
;22501  ;       Error1: If this error occurs then the BRANCH 2 PAL, the MUL/DIV PAL
;22502  ;               or the EXT FUNC CTL PAL are in error. If all errors are
;22503  ;               failing then it's likely that the BRANCH 2 PAL is the cause
;22504  ;               of error. If Q16 appears to be stuck either high or low
;22505  ;               causing every other error to occur then the EXT FUNC PAL
;22506  ;               is probably the cause of error. Any other errors are probably
;22507  ;               the result of the MUL/DIV PAL. If not mentioned in the error
;22508  ;               description the integer and emod are unasserted.
;22509  ;       Error2: Same as error 1.
;22510  ;       Error3: Same as error 1.
;22511  ;       Error4: Same as error 1.
;22512  ;       Error5: Same as error 1.
;22513  ;       Error6: Same as error 1.
;22514  ;       Error7: Same as error 1.
;22515  ;       Error8: Same as error 1.
;22516  ;       Error9: Same as error 1.
;22517  ;       ErrorA: Same as error 1.
;22518  ;       ErrorB: If this error occurs then it's likely that the BRANCH 1 PAL
;22519  ;                is the cause of error. FPAC FRAC55 Q3 the input to the pal
;22520  ;                and FPAB BRANCH 0 the output of the pal should both be
;22521  ;                asserted.
;22522  ;
;22523  ;
```

```
                ;22524  ;--
                ;22525  T.41:
U 241E, B65E,15 ;22526          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;22527                                          ; STATUS WORD
U 241F, 3E80,15 ;22528          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;22529                                          ; BIT 15 INDICATES START OF TEST TO
                ;22530                                          ; CONSOLE
U 2420, 10E0,15 ;22531          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;22532  WAIT.T41.0:
U 2421, 8A42,14 ;22533          JMP [WAIT.T41.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;22534                                          ; RESPOND
U 2422, 887E,EC ;22535          JSR [SETUP.10]                  ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;22536                                          ; AND SIZE BITS. CHECK LOWER 10 BITS
U 2423, B7A6,15 ;22537          MOV LS[T41.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2424, BE12,15 ;22538          MOV WR[0] TO LS[T9]
U 2425, 8870,3C ;22539          JSR [LO]                        ;Get address of TOGGLE Q16 DEFAULT
U 2426, BE14,15 ;22540          MOV WR[0] TO LS[T10]
U 2427, 8870,3C ;22541          JSR [LO]                        ;Get address of SHIFT LEFT FQ
U 2428, 3E16,15 ;22542          MOV WR[0] TO LS[T11]
U 2429, 8870,3C ;22543          JSR [LO]                        ;Get address of Branch
U 242A, BE18,15 ;22544          MOV WR[0] TO LS[T12]
U 242B, 8870,3C ;22545          JSR [LO]                        ;Get address of MOV FTO TO FQ
U 242C, 3E1C,15 ;22546          MOV WR[0] TO LS[T14]
U 242D, 8870,3C ;22547          JSR [LO]                        ;Get address of SHIFT FQ RIGHT
U 242E, BE1E,15 ;22548          MOV WR[0] TO LS[T15]
U 242F, 3716,15 ;22549          MOV LS[#300] TO WR[0]           ;Set the instruction and size bits
U 2430, C764,15 ;22550          BIS LS[BIT18] TO WR[0]          ; to zero
U 2431, 90F6,15 ;22551          MISC2 [TRAP.ACC] WR[0]
U 2432, E5A2,15 ;22552  T41.1:  CLR LS[DATA.1]                  ;Load zeros into data.1
U 2433, B640,15 ;22553          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 2434, 3EA4,15 ;22554          MOV WR[0] TO LS[DATA.2]
U 2435, 65A6,15 ;22555          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2436, 0878,FC ;22556          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 2437, 0878,AC ;22557          JSR [MOV.DATA]                  ;MOV FTO TO FQ
U 2438, 361E,15 ;22558          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2439, B618,95 ;22559          MOV LS[T12] TO WR[1]            ;Set up to branch
U 243A, 90F6,15 ;22560          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 243B, 10F6,95 ;22561          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 243C, 10F8,15 ;22562          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 243D, BA1A,15 ;22563          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 243E, B61A,15 ;22564          MOV LS[T13] TO WR[0]            ;Set up received data
U 243F, B716,95 ;22565          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2440, 0869,3C ;22566          JSR [CHECK.RESULT]              ;Check for error
U 2441, 0A43,24 ;22567          JMP [T41.1]                     ;Loop on error
U 2442, 3660,15 ;22568  T41.2:  MOV LS[BIT16] TO WR[0]          ;Set bit 16 in data.1
U 2443, 3EA2,15 ;22569          MOV WR[0] TO LS[DATA.1]
U 2444, E5A4,15 ;22570          CLR LS[DATA.2]                  ;Load zeros into data.2
U 2445, 65A6,15 ;22571          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2446, 0878,FC ;22572          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 2447, 0878,AC ;22573          JSR [MOV.DATA]                  ;MOV FTO TO FQ
U 2448, 361E,15 ;22574          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2449, B618,95 ;22575          MOV LS[T12] TO WR[1]            ;Set up to branch
U 244A, 90F6,15 ;22576          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 244B, 10F6,95 ;22577          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 244C, 10F8,15 ;22578          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
```

```
U 244D, BA1A,15 ;22579          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 244E, B61A,15 ;22580          MOV LS[T13] TO WR[0]            ;Set up received data
U 244F, B740,95 ;22581          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2450, 0869,3C ;22582          JSR [CHECK.RESULT]              ;Check for error
U 2451, 8A44,24 ;22583          JMP [T41.2]                     ;Loop on error
U 2452, FF82,15 ;22584          INC LS[ERROR.NUMBER]            ;Go onto error 2
U 2453, E5A2,15 ;22585  T41.3:  CLR LS[DATA.1]                  ;Load zeros into data.1
U 2454, B640,15 ;22586          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 2455, 3EA4,15 ;22587          MOV WR[0] TO LS[DATA.2]
U 2456, 65A6,15 ;22588          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2457, 0878,FC ;22589          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2458, 0878,AC ;22590          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 2459, 3614,15 ;22591          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 245A, B618,95 ;22592          MOV LS[T12] TO WR[1]            ;Set up to branch
U 245B, 90F6,15 ;22593          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 245C, DB00,15 ;22594          NOP
U 245D, 10F6,95 ;22595          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 245E, 10F8,15 ;22596          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 245F, BA1A,15 ;22597          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2460, B61A,15 ;22598          MOV LS[T13] TO WR[0]            ;Set up received data
U 2461, B740,95 ;22599          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2462, 0869,3C ;22600          JSR [CHECK.RESULT]              ;Check for error
U 2463, 8A45,34 ;22601          JMP [T41.3]                     ;Loop on error
U 2464, 3660,15 ;22602  T41.4:  MOV LS[BIT16] TO WR[0]          ;Set bit 16 in data.1
U 2465, 3EA2,15 ;22603          MOV WR[0] TO LS[DATA.1]
U 2466, E5A4,15 ;22604          CLR LS[DATA.2]                  ;Load zeros into data.2
U 2467, 65A6,15 ;22605          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2468, 0878,FC ;22606          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2469, 0878,AC ;22607          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 246A, 3614,15 ;22608          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 246B, B618,95 ;22609          MOV LS[T12] TO WR[1]            ;Set up to branch
U 246C, 90F6,15 ;22610          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 246D, DB00,15 ;22611          NOP
U 246E, 10F6,95 ;22612          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 246F, 10F8,15 ;22613          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2470, BA1A,15 ;22614          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2471, B61A,15 ;22615          MOV LS[T13] TO WR[0]            ;Set up received data
U 2472, B716,95 ;22616          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2473, 0869,3C ;22617          JSR [CHECK.RESULT]              ;Check for error
U 2474, 0A46,44 ;22618          JMP [T41.4]                     ;Loop on error
U 2475, FF82,15 ;22619          INC LS[ERROR.NUMBER]            ;Go onto error 3
U 2476, 3716,15 ;22620          MOV LS[#300] TO WR[0]           ;Set the instruction bit to integer
U 2477, C764,15 ;22621          BIS LS[BIT18] TO WR[0]
U 2478, 475A,15 ;22622          BIS LS[BIT13] TO WR[0]
U 2479, 475C,15 ;22623          BIS LS[BIT14] TO WR[0]
U 247A, 90F6,15 ;22624          MISC2 [TRAP.ACC] WR[0]
U 247B, E5A2,15 ;22625  T41.5:  CLR LS[DATA.1]                  ;Load zeros into data.1
U 247C, B640,15 ;22626          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 247D, 3EA4,15 ;22627          MOV WR[0] TO LS[DATA.2]
U 247E, 65A6,15 ;22628          CLR LS[DATA.3]                  ;Load zeros into data.3
U 247F, 0878,FC ;22629          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2480, 0878,AC ;22630          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 2481, 361E,15 ;22631          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2482, B618,95 ;22632          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2483, 90F6,15 ;22633          MISC2 [TRAP.ACC] WR[0]          ;Shift right
```

```
U 2484, 10F6,95 ;22634          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2485, 10F8,15 ;22635          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2486, BA1A,15 ;22636          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2487, B61A,15 ;22637          MOV LS[T13] TO WR[0]            ;Set up received data
U 2488, B740,95 ;22638          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2489, 0869,3C ;22639          JSR [CHECK.RESULT]              ;Check for error
U 248A, 8A47,B4 ;22640          JMP [T41.5]                     ;Loop on error
U 248B, 3660,15 ;22641  T41.6:  MOV LS[BIT16] TO WR[0]          ;Set bit 16 in data.1
U 248C, 3EA2,15 ;22642          MOV WR[0] TO LS[DATA.1]
U 248D, E5A4,15 ;22643          CLR LS[DATA.2]                  ;Load zeros into data.2
U 248E, 65A6,15 ;22644          CLR LS[DATA.3]                  ;Load zeros into data.3
U 248F, 0878,FC ;22645          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2490, 0878,AC ;22646          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 2491, 361E,15 ;22647          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2492, B618,95 ;22648          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2493, 90F6,15 ;22649          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2494, 10F6,95 ;22650          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2495, 10F8,15 ;22651          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2496, BA1A,15 ;22652          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2497, B61A,15 ;22653          MOV LS[T13] TO WR[0]            ;Set up received data
U 2498, B716,95 ;22654          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2499, 0869,3C ;22655          JSR [CHECK.RESULT]              ;Check for error
U 249A, 8A48,B4 ;22656          JMP [T41.6]                     ;Loop on error
U 249B, FF82,15 ;22657          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 249C, 3716,15 ;22658          MOV LS[#300] TO WR[0]           ;Set instruction bits to EMOD
U 249D, C764,15 ;22659          BIS LS[BIT18] TO WR[0]
U 249E, C754,15 ;22660          BIS LS[BIT10] TO WR[0]
U 249F, 4756,15 ;22661          BIS LS[BIT11] TO WR[0]
U 24A0, C758,15 ;22662          BIS LS[BIT12] TO WR[0]
U 24A1, 90F6,15 ;22663          MISC2 [TRAP.ACC] WR[0]
U 24A2, E5A2,15 ;22664  T41.7:  CLR LS[DATA.1]                  ;Load zeros into data.1
U 24A3, B640,15 ;22665          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 24A4, 3EA4,15 ;22666          MOV WR[0] TO LS[DATA.2]
U 24A5, 65A6,15 ;22667          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24A6, 0878,FC ;22668          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24A7, 0878,AC ;22669          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24A8, 361E,15 ;22670          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 24A9, B618,95 ;22671          MOV LS[T12] TO WR[1]            ;Set up to branch
U 24AA, 90F6,15 ;22672          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 24AB, 10F6,95 ;22673          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 24AC, 10F8,15 ;22674          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 24AD, BA1A,15 ;22675          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 24AE, B61A,15 ;22676          MOV LS[T13] TO WR[0]            ;Set up received data
U 24AF, B716,95 ;22677          MOV LS[#300] TO WR[1]           ;Set up expected data
U 24B0, 0869,3C ;22678          JSR [CHECK.RESULT]              ;Check for error
U 24B1, CA4A,24 ;22679          JMP [T41.7]                     ;Loop on error
U 24B2, 3660,15 ;22680  T41.8:  MOV LS[BIT16] TO WR[0]          ;Set bit 16 in data.1
U 24B3, 3EA2,15 ;22681          MOV WR[0] TO LS[DATA.1]
U 24B4, E5A4,15 ;22682          CLR LS[DATA.2]                  ;Load zeros into data.2
U 24B5, 65A6,15 ;22683          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24B6, 0878,FC ;22684          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24B7, 0878,AC ;22685          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24B8, 361E,15 ;22686          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 24B9, B618,95 ;22687          MOV LS[T12] TO WR[1]            ;Set up to branch
U 24BA, 90F6,15 ;22688          MISC2 [TRAP.ACC] WR[0]          ;Shift right
```

```
U 24BB, 10F6,95 ;22689          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 24BC, 10F8,15 ;22690          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 24BD, BA1A,15 ;22691          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 24BE, B61A,15 ;22692          MOV LS[T13] TO WR[0]            ;Set up received data
U 24BF, B740,95 ;22693          MOV LS[#302] TO WR[1]           ;Set up expected data
U 24C0, 0869,3C ;22694          JSR [CHECK.RESULT]              ;Check for error
U 24C1, 8A4B,24 ;22695          JMP [T41.8]                     ;Loop on error
U 24C2, FF82,15 ;22696          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 24C3, 3716,15 ;22697          MOV LS[#300] TO WR[0]           ;Set the size bits to double
U 24C4, C764,15 ;22698          BIS LS[BIT18] TO WR[0]
U 24C5, 4760,15 ;22699          BIS LS[BIT16] TO WR[0]
U 24C6, 90F6,15 ;22700          MISC2 [TRAP.ACC] WR[0]
U 24C7, E5A2,15 ;22701  T41.9:  CLR LS[DATA.1]                  ;Load zeros into data.1
U 24C8, B640,15 ;22702          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 24C9, 3EA4,15 ;22703          MOV WR[0] TO LS[DATA.2]
U 24CA, 65A6,15 ;22704          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24CB, 0878,FC ;22705          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24CC, 0878,AC ;22706          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24CD, 361E,15 ;22707          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 24CE, B618,95 ;22708          MOV LS[T12] TO WR[1]            ;Set up to branch
U 24CF, 90F6,15 ;22709          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 24D0, 10F6,95 ;22710          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 24D1, 10F8,15 ;22711          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 24D2, BA1A,15 ;22712          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 24D3, B61A,15 ;22713          MOV LS[T13] TO WR[0]            ;Set up received data
U 24D4, B716,95 ;22714          MOV LS[#300] TO WR[1]           ;Set up expected data
U 24D5, 0869,3C ;22715          JSR [CHECK.RESULT]              ;Check for error
U 24D6, 0A4C,74 ;22716          JMP [T41.9]                     ;Loop on error
U 24D7, E5A2,15 ;22717  T41.10: CLR LS[DATA.1]                  ;Load zeros into data.1
U 24D8, 3660,15 ;22718          MOV LS[BIT16] TO WR[0]          ;Set bit 16 of data.2
U 24D9, 3EA4,15 ;22719          MOV WR[0] TO LS[DATA.2]
U 24DA, 65A6,15 ;22720          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24DB, 0878,FC ;22721          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24DC, 0878,AC ;22722          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24DD, 361E,15 ;22723          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 24DE, B618,95 ;22724          MOV LS[T12] TO WR[1]            ;Set up to branch
U 24DF, 90F6,15 ;22725          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 24E0, 10F6,95 ;22726          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 24E1, 10F8,15 ;22727          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 24E2, BA1A,15 ;22728          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 24E3, B61A,15 ;22729          MOV LS[T13] TO WR[0]            ;Set up received data
U 24E4, B740,95 ;22730          MOV LS[#302] TO WR[1]           ;Set up expected data
U 24E5, 0869,3C ;22731          JSR [CHECK.RESULT]              ;Check for error
U 24E6, 8A4D,74 ;22732          JMP [T41.10]                    ;Loop on error
U 24E7, FF82,15 ;22733          INC LS[ERROR.NUMBER]            ;Go onto error 6
U 24E8, E5A2,15 ;22734  T41.11: CLR LS[DATA.1]                  ;Load zeros into data.1
U 24E9, B640,15 ;22735          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 24EA, 3EA4,15 ;22736          MOV WR[0] TO LS[DATA.2]
U 24EB, 65A6,15 ;22737          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24EC, 0878,FC ;22738          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24ED, 0878,AC ;22739          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24EE, 3614,15 ;22740          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 24EF, B618,95 ;22741          MOV LS[T12] TO WR[1]            ;Set up to branch
U 24F0, 90F6,15 ;22742          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 24F1, DB00,15 ;22743          NOP
```

```
U 24F2, 10F6,95 ;22744          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 24F3, 10F8,15 ;22745          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 24F4, BA1A,15 ;22746          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 24F5, B61A,15 ;22747          MOV LS[T13] TO WR[0]            ;Set up received data
U 24F6, B740,95 ;22748          MOV LS[#302] TO WR[1]           ;Set up expected data
U 24F7, 0869,3C ;22749          JSR [CHECK.RESULT]              ;Check for error
U 24F8, 8A4E,84 ;22750          JMP [T41.11]                    ;Loop on error
U 24F9, E5A2,15 ;22751  T41.12: CLR LS[DATA.1]                  ;Load zeros into data.1
U 24FA, 3660,15 ;22752          MOV LS[BIT16] TO WR[0]          ;Set bit 16 of data.2
U 24FB, 3EA4,15 ;22753          MOV WR[0] TO LS[DATA.2]
U 24FC, 65A6,15 ;22754          CLR LS[DATA.3]                  ;Load zeros into data.3
U 24FD, 0878,FC ;22755          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 24FE, 0878,AC ;22756          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 24FF, 3614,15 ;22757          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 2500, B618,95 ;22758          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2501, 90F6,15 ;22759          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2502, DB00,15 ;22760          NOP
U 2503, 10F6,95 ;22761          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2504, 10F8,15 ;22762          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2505, BA1A,15 ;22763          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2506, B61A,15 ;22764          MOV LS[T13] TO WR[0]            ;Set up received data
U 2507, B716,95 ;22765          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2508, 0869,3C ;22766          JSR [CHECK.RESULT]              ;Check for error
U 2509, 8A4F,94 ;22767          JMP [T41.12]                    ;Loop on error
U 250A, FF82,15 ;22768          INC LS[ERROR.NUMBER]            ;Go onto error 7
U 250B, 3716,15 ;22769          MOV LS[#300] TO WR[0]           ;Set the size bits to grand
U 250C, C764,15 ;22770          BIS LS[BIT18] TO WR[0]
U 250D, C762,15 ;22771          BIS LS[BIT17] TO WR[0]
U 250E, 90F6,15 ;22772          MISC2 [TRAP.ACC] WR[0]
U 250F, E5A2,15 ;22773  T41.13: CLR LS[DATA.1]                  ;Load zeros into data.1
U 2510, B640,15 ;22774          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 2511, 3EA4,15 ;22775          MOV WR[0] TO LS[DATA.2]
U 2512, 65A6,15 ;22776          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2513, 0878,FC ;22777          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2514, 0878,AC ;22778          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 2515, 361E,15 ;22779          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2516, B618,95 ;22780          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2517, 90F6,15 ;22781          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2518, 10F6,95 ;22782          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2519, 10F8,15 ;22783          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 251A, BA1A,15 ;22784          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 251B, B61A,15 ;22785          MOV LS[T13] TO WR[0]            ;Set up received data
U 251C, B716,95 ;22786          MOV LS[#300] TO WR[1]           ;Set up expected data
U 251D, 0869,3C ;22787          JSR [CHECK.RESULT]              ;Check for error
U 251E, 0A50,F4 ;22788          JMP [T41.13]                    ;Loop on error
U 251F, E5A2,15 ;22789  T41.14: CLR LS[DATA.1]                  ;Load zeros into data.1
U 2520, 3660,15 ;22790          MOV LS[BIT16] TO WR[0]          ;Set bit 16 of data.2
U 2521, 3EA4,15 ;22791          MOV WR[0] TO LS[DATA.2]
U 2522, 65A6,15 ;22792          CLR LS[DATA.3]                  ;Load zeros into data.3
U 2523, 0878,FC ;22793          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2524, 0878,AC ;22794          JSR [MOV.DATA]                  ;MOV FT0 TO FQ
U 2525, 361E,15 ;22795          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 2526, B618,95 ;22796          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2527, 90F6,15 ;22797          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2528, 10F6,95 ;22798          MISC2 [TRAP.ACC] WR[1]          ;Branch
```

```
U 2529, 10F8.15 ;22799          MISC2 [READ.ACC.UPC]          ;Enable NUA onto FPA
U 252A, BA1A.15 ;22800          MOV FPA TO LS[T13]            ;Enable FPA onto Y
U 252B, B61A.15 ;22801          MOV LS[T13] TO WR[0]          ;Set up received data
U 252C, B740.95 ;22802          MOV LS[#302] TO WR[1]         ;Set up expected data
U 252D, 0869.3C ;22803          JSR [CHECK.RESULT]            ;Check for error
U 252E, 8A51.F4 ;22804          JMP [T41.14]                  ;Loop on error
U 252F, FF82.15 ;22805          INC LS[ERROR.NUMBER]          ;Go onto error 8
U 2530, E5A2.15 ;22806  T41.15: CLR LS[DATA.1]                ;Load zeros into data.1
U 2531, B640.15 ;22807          MOV LS[BIT0] TO WR[0]         ;Set bit 0 of data.2
U 2532, 3EA4.15 ;22808          MOV WR[0] TO LS[DATA.2]
U 2533, 65A6.15 ;22809          CLR LS[DATA.3]                ;Load zeros into data.3
U 2534, 0878.FC ;22810          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2535, 0878.AC ;22811          JSR [MOV.DATA]                ;MOV FT0 TO FQ
U 2536, 3614.15 ;22812          MOV LS[T10] TO WR[0]          ;Set up to shift right
U 2537, B618.95 ;22813          MOV LS[T12] TO WR[1]          ;Set up to branch
U 2538, 90F6.15 ;22814          MISC2 [TRAP.ACC] WR[0]        ;Shift right
U 2539, DB00.15 ;22815          NOP
U 253A, 10F6.95 ;22816          MISC2 [TRAP.ACC] WR[1]        ;Branch
U 253B, 10F8.15 ;22817          MISC2 [READ.ACC.UPC]          ;Enable NUA onto FPA
U 253C, BA1A.15 ;22818          MOV FPA TO LS[T13]            ;Enable FPA onto Y
U 253D, B61A.15 ;22819          MOV LS[T13] TO WR[0]          ;Set up received data
U 253E, B740.95 ;22820          MOV LS[#302] TO WR[1]         ;Set up expected data
U 253F, 0869.3C ;22821          JSR [CHECK.RESULT]            ;Check for error
U 2540, 0A53.04 ;22822          JMP [T41.15]                  ;Loop on error
U 2541, E5A2.15 ;22823  T41.16: CLR LS[DATA.1]                ;Load zeros into data.1
U 2542, 3660.15 ;22824          MOV LS[BIT16] TO WR[0]        ;Set bit 16 of data.2
U 2543, 3EA4.15 ;22825          MOV WR[0] TO LS[DATA.2]
U 2544, 65A6.15 ;22826          CLR LS[DATA.3]                ;Load zeros into data.3
U 2545, 0878.FC ;22827          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2546, 0878.AC ;22828          JSR [MOV.DATA]                ;MOV FT0 TO FQ
U 2547, 3614.15 ;22829          MOV LS[T10] TO WR[0]          ;Set up to shift right
U 2548, B618.95 ;22830          MOV LS[T12] TO WR[1]          ;Set up to branch
U 2549, 90F6.15 ;22831          MISC2 [TRAP.ACC] WR[0]        ;Shift right
U 254A, DB00.15 ;22832          NOP
U 254B, 10F6.95 ;22833          MISC2 [TRAP.ACC] WR[1]        ;Branch
U 254C, 10F8.15 ;22834          MISC2 [READ.ACC.UPC]          ;Enable NUA onto FPA
U 254D, BA1A.15 ;22835          MOV FPA TO LS[T13]            ;Enable FPA onto Y
U 254E, B61A.15 ;22836          MOV LS[T13] TO WR[0]          ;Set up received data
U 254F, B716.95 ;22837          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2550, 0869.3C ;22838          JSR [CHECK.RESULT]            ;Check for error
U 2551, 0A54.14 ;22839          JMP [T41.16]                  ;Loop on error
U 2552, FF82.15 ;22840          INC LS[ERROR.NUMBER]          ;Go onto error 9
U 2553, 3716.15 ;22841          MOV LS[#300] TO WR[0]         ;Set the size bits to huge
U 2554, C764.15 ;22842          BIS LS[BIT18] TO WR[0]
U 2555, 4760.15 ;22843          BIS LS[BIT16] TO WR[0]
U 2556, C762.15 ;22844          BIS LS[BIT17] TO WR[0]
U 2557, 90F6.15 ;22845          MISC2 [TRAP.ACC] WR[0]
U 2558, E5A2.15 ;22846  T41.17: CLR LS[DATA.1]                ;Load zeros into data.1
U 2559, B640.15 ;22847          MOV LS[BIT0] TO WR[0]         ;Set bit 0 of data.2
U 255A, 3EA4.15 ;22848          MOV WR[0] TO LS[DATA.2]
U 255B, 65A6.15 ;22849          CLR LS[DATA.3]                ;Load zeros into data.3
U 255C, 0878.FC ;22850          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 255D, 0878.AC ;22851          JSR [MOV.DATA]                ;MOV FT0 TO FQ
U 255E, 361E.15 ;22852          MOV LS[T15] TO WR[0]          ;Set up to shift right
U 255F, B618.95 ;22853          MOV LS[T12] TO WR[1]          ;Set up to branch
```

```
U 2560, 90F6,15 ;22854          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2561, 10F6,95 ;22855          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2562, 10F8,15 ;22856          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2563, BA1A,15 ;22857          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2564, B61A,15 ;22858          MOV LS[T13] TO WR[0]            ;Set up received data
U 2565, B716,95 ;22859          MOV LS[#300] TO WR[1]           ;Set up expected data
U 2566, 0869,3C ;22860          JSR [CHECK.RESULT]              ;Check for error
U 2567, 8A55,84 ;22861          JMP [T41.17]                    ;Loop on error
U 2568, E5A2,15 ;22862  T41.18: CLR LS[DATA.1]                  ;Load zeros into data.1
U 2569, E5A4,15 ;22863          CLR LS[DATA.2]                  ;Load zeros into data.2
U 256A, 3650,15 ;22864          MOV LS[BIT8] TO WR[0]           ;Set bit 8 in data.3
U 256B, BEA6,15 ;22865          MOV WR[0] TO LS[DATA.3]
U 256C, 0878,FC ;22866          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 256D, 0878,AC ;22867          JSR [MOV.DATA]                  ;MOV FTO TO FQ
U 256E, 351E,15 ;22868          MOV LS[T15] TO WR[0]            ;Set up to shift right
U 256F, B618,95 ;22869          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2570, 90F6,15 ;22870          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2571, 10F6,95 ;22871          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2572, 10F8,15 ;22872          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2573, BA1A,15 ;22873          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2574, B61A,15 ;22874          MOV LS[T13] TO WR[0]            ;Set up received data
U 2575, B740,95 ;22875          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2576, 0869,3C ;22876          JSR [CHECK.RESULT]              ;Check for error
U 2577, 8A56,84 ;22877          JMP [T41.18]                    ;Loop on error
U 2578, FF82,15 ;22878          INC LS[ERROR.NUMBER]            ;Go onto error 10
U 2579, E5A2,15 ;22879  T41.19: CLR LS[DATA.1]                  ;Load zeros into data.1
U 257A, B640,15 ;22880          MOV LS[BIT0] TO WR[0]           ;Set bit 0 of data.2
U 257B, 3EA4,15 ;22881          MOV WR[0] TO LS[DATA.2]
U 257C, 65A6,15 ;22882          CLR LS[DATA.3]                  ;Load zeros into data.3
U 257D, 0878,FC ;22883          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 257E, 0878,AC ;22884          JSR [MOV.DATA]                  ;MOV FTO TO FQ
U 257F, 3614,15 ;22885          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 2580, B618,95 ;22886          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2581, 90F6,15 ;22887          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2582, DB00,15 ;22888          NOP
U 2583, 10F6,95 ;22889          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2584, 10F8,15 ;22890          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2585, BA1A,15 ;22891          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 2586, B61A,15 ;22892          MOV LS[T13] TO WR[0]            ;Set up received data
U 2587, B740,95 ;22893          MOV LS[#302] TO WR[1]           ;Set up expected data
U 2588, 0869,3C ;22894          JSR [CHECK.RESULT]              ;Check for error
U 2589, 8A57,94 ;22895          JMP [T41.19]                    ;Loop on error
U 258A, E5A2,15 ;22896  T41.1A: CLR LS[DATA.1]                  ;Load zeros into data.1
U 258B, E5A4,15 ;22897          CLR LS[DATA.2]                  ;Load zeros into data.2
U 258C, 3650,15 ;22898          MOV LS[BIT8] TO WR[0]           ;Set bit 8 in data.3
U 258D, BEA6,15 ;22899          MOV WR[0] TO LS[DATA.3]
U 258E, 0878,FC ;22900          JSR [LOAD.TRIPLE]               ;Load data into FTO
U 258F, 0878,AC ;22901          JSR [MOV.DATA]                  ;MOV FTO TO FQ
U 2590, 3614,15 ;22902          MOV LS[T10] TO WR[0]            ;Set up to shift right
U 2591, B618,95 ;22903          MOV LS[T12] TO WR[1]            ;Set up to branch
U 2592, 90F6,15 ;22904          MISC2 [TRAP.ACC] WR[0]          ;Shift right
U 2593, DB00,15 ;22905          NOP
U 2594, 10F6,95 ;22906          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 2595, 10F8,15 ;22907          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 2596, BA1A,15 ;22908          MOV FPA TO LS[T13]              ;Enable FPA onto Y
```

```
U 2597, B61A,15 ;22909          MOV LS[T13] TO WR[0]          ;Set up received data
U 2598, B716,95 ;22910          MOV LS[#300] TO WR[1]         ;Set up expected data
U 2599, 0869,3C ;22911          JSR [CHECK.RESULT]            ;Check for error
U 259A, 8A58,A4 ;22912          JMP [T41.1A]                  ;Loop on error
U 259B, FF82,15 ;22913          INC LS[ERROR.NUMBER]          ;Go onto error 11
U 259C, 3716,15 ;22914          MOV LS[#300] TO WR[0]         ;Set size bits to floating
U 259D, C764,15 ;22915          BIS LS[BIT18] TO WR[0]
U 259E, 90F6,15 ;22916          MISC2 [TRAP.ACC] WR[0]
U 259F, 087C,1C ;22917  T41.1B: JSR [CLR.FT0]                 ;LOAD ZEROS INTO FT0
U 25A0, 0878,AC ;22918          JSR [MOV.DATA]                ;MOV FT0 TO FQ
U 25A1, B616,15 ;22919          MOV LS[T11] TO WR[0]          ;Setup to Shift FQ left
U 25A2, B618,95 ;22920          MOV LS[T12] TO WR[1]          ;Setup to branch
U 25A3, 90F6,15 ;22921          MISC2 [TRAP.ACC] WR[0]        ;SHIFT FQ LEFT
U 25A4, 10F6,95 ;22922          MISC2 [TRAP.ACC] WR[1]        ;Branch
U 25A5, 10F8,15 ;22923          MISC2 [READ.ACC.UPC]          ;Enable NUA onto FPA
U 25A6, BA1A,15 ;22924          MOV FPA TO LS[T13]            ;Enable FPA onto Y
U 25A7, B61A,15 ;22925          MOV LS[T13] TO WR[0]          ;Setup received data
U 25A8, B73E,95 ;22926          MOV LS[#301] TO WR[1]         ;Setup expected data
U 25A9, 0869,3C ;22927          JSR [CHECK.RESULT]            ;Check for error
U 25AA, 0A59,F4 ;22928          JMP [T41.1B]                  ;Loop on error
                ;22929  T41.END:
```

```
;22930  .PAGE
;22931  .TOC    "TEST 42 FRAC<55:32>=0 AND DIV I3 BRANCH TEST(M8389 FPA MODULE)"
;22932  ;++
;22933  ;
;22934  ;
;22935  ; Test Description:
;22936  ;
;22937  ;       The purpose of this test is make sure the FRAC<55:32>=0 and the DIV I3
;22938  ;       branch condition and the logic associated with those signals are
;22939  ;       working. DIV I3 is a function of the signals SIZE 1, SIZE 0, INTEGER,
;22940  ;       FRAC55 R3 SAVE, FRAC I3, HUGE R3 SV, and FRAC47 F3. When the size bits
;22941  ;       are not huge and integer is unasserted DIV I3 equals the exclusive nor
;22942  ;       of FRAC I3, FRAC55 R3 SAVE, and FCOUT. When INTEGER is asserted then
;22943  ;       DIV I3 equals FRAC47 F3. When the size btis are huge DIV I3 becomes a
;22944  ;       function of HUGE R3 SV. Since huge requires that TOGGLE HUGE R3 SV be
;22945  ;       executed and a force clears TOGGLE HUGE R3 SV the case of DIV I3 when
;22946  ;       the size bits equal huge won't be tested here but will be tested in the
;22947  ;       MUL/DIV TEST.
;22948  ;
;22949  ; Assumptions:
;22950  ;
;22951  ;       All other branch tested have been run successfully by this point.
;22952  ;
;22953  ; Test Steps:
;22954  ;
;22955  ;       1) Set the size bits to floating. Load FT3 and FT0 with zero and FT1
;22956  ;          with zeros and bit 55 set. Force ADD SHFL FWR[FT1] TO FWR[FT3]
;22957  ;          and then ADD SHFL FWR[FT0] TO FWR[FT2] + FCOUT, then branch. If
;22958  ;          there is a branch then issue error 1.
;22959  ;       2) Repeat step one except load FT1 with zeros. If there isn't a branch
;22960  ;          then issue error 2.
;22961  ;       3) Set the size bits to double. Load FT3 and FT0 with zero and FT1
;22962  ;          with zeros and bit 55 set. Force ADD SHFL FWR[FT1] TO FWR[FT3]
;22963  ;          and then ADD SHFL FWR[FT0] TO FWR[FT2] + FCOUT, then branch. If
;22964  ;          there is a branch then issue error 3.
;22965  ;       4) Repeat step 3 except load FT1 with zeros. If there isn't a branch
;22966  ;          then issue error 4.
;22967  ;       5) Set the size bits to grand. Load FT3 and FT0 with zero and FT1
;22968  ;          with zeros and bit 55 set. Force ADD SHFL FWR[FT1] TO FWR[FT3]
;22969  ;          and then ADD SHFL FWR[FT0] TO FWR[FT2] + FCOUT, then branch. If
;22970  ;          there is a branch then issue error 5.
;22971  ;       6) Repeat step 5 except load FT1 with zeros. If there isn't a branch
;22972  ;          then issue error 6.
;22973  ;       7) Set the size bits to floating. Load FT3 and FT1 with zeros and FT0
;22974  ;          and FT2 with bit 55 set and zeros everywhere else. execute the same
;22975  ;          two instructions followed by a branch. If there is a branch then
;22976  ;          issue error 7.
;22977  ;       8) Load FT3 with zeros and FT0, FT1, and FT2 with bit 55 set and zeros
;22978  ;          in the rest. execute the two instruction followed by a branch. If
;22979  ;          there is no branch then issue error 8.
;22980  ;       9) Load FT0, FT1, and FT3 with zeros and load FT2 with bit 55 set.
;22981  ;          Force SUB SHFL FWR[FT1] FROM FWR[FT3], then SUB SHFL FWR[FT0] FROM
;22982  ;          FWR[FT2] + FCOUT, then branch. If there's branch then issue error
;22983  ;          9.
;22984  ;      10) Load FT0 and FT1 with zeros and FT2 and FT3 with ones.Force SUB SHFL
```

```
;22985  ;          FWR[FT1] FROM FWR[FT3], then SUB SHFL FWR[Ft0] FROM FWR[FT2] + FCOUT
;22986  ;          then branch. If there isn't any branch then issue error A.
;22987  ;      11) Load FT0, FT1, FT2, and FT3 with zeros. Force SUB SHFL FWR[FT1] FROM
;22988  ;          FWR[FT3], then SUB SHFL FWR[FT0] FROM FWR[FT2] + FCOUT, then branch.
;22989  ;          If there isn't any branch then issue error B.
;22990  ;      12) Load FT0, FT1, and FT2 with zeros and load FT3 with bit 55 set.
;22991  ;          Force SUB SHFL FWR[FT1] FROM FWR[FT3], then SUB SHFL FWR[FT0] FROM
;22992  ;          FWR[FT2] + FCOUT, then branch. If there is any branch then issue
;22993  ;          error C.
;22994  ;      13) Set the instruction bits to integer. Load FT0 with zeros and bit 47
;22995  ;          with a one. Move FT0 TO FQ then branch. If there is no branch then
;22996  ;          issue error D.
;22997  ;      14) Repeat Step 14 with bit 47 set to zero. If there is any branch then
;22998  ;          issue error E.
;22999  ;      15) Load FT0 with bit 55 set. MOV FT0 TO FQ then branch. If there isn't
;23000  ;          a branch then issue error F.
;23001  ;
;23002  ; Error:
;23003  ;
;23004  ;       Error1 - DIV I3 branch failed when the size bits were floating, the
;23005  ;                instruction bits were null, and FRAC55 R3 SAVE was asserted.
;23006  ;       Error2 - DIV I3 branch occured when it was not supposed to and the size
;23007  ;                bit were floating, the instruction bits were null, and
;23008  ;                FRAC55 R3 SAVE was unasserted.
;23009  ;       Error3 - Same as error 1 except that the size bits are double.
;23010  ;       Error4 - Same as error 2 except that the size bits are double.
;23011  ;       Error5 - Same as error 1 except that the size bits are grand.
;23012  ;       Error6 - Same as error 2 except that the size bits are grand.
;23013  ;       Error7 - DIV I3 branch failed when the size bits were floating, the
;23014  ;                instruction bits were null, FCOUT was asserted, and FRAC I3
;23015  ;                and FRAC55 R3 SAVE were unasserted.
;23016  ;       Error8 - DIV I3 branch failed when the size bits were floating, the
;23017  ;                instruction bits were null, FCOUT and FRAC55 R3 SAVE are
;23018  ;                asserted and FRAC I3 is unasserted.
;23019  ;       Error9 - DIV I3 branch failed when the size bits were floating, the
;23020  ;                instruction bits were null, FRAC I3 was asserted and FCOUT
;23021  ;                and FRAC55 R3 SAVE were unasserted.
;23022  ;       ErrorA - DIV I3 branch failed when the size bits were floating, the
;23023  ;                instruction bits were null, FRAC I3 and FRAC55 R3 SAVE
;23024  ;                were asserted and FCOUT was unasserted.
;23025  ;       ErrorB - DIV I3 branch failed when the size bits were floating, the
;23026  ;                instruction bits were null, FRAC I3 and FCOUT were asserted
;23027  ;                and FRAC55 R3 SAVE was unasserted.
;23028  ;       ErrorC - DIV I3 branch occured when the size bits were floating, the
;23029  ;                instruction bits were null, and FRAC I3, FCOUT, and FRAC55
;23030  ;                R3 SAVE were asserted.
;23031  ;       ErrorD - DIV I3 or FRAC<55:32>=0 branch failed when the size bits were
;23032  ;                floating, the instruction bits were integer, and BIT 47 was
;23033  ;                asserted.
;23034  ;       ErrorE - DIV I3 or FRAC<55:32>=0 branch occured when the size bits were
;23035  ;                floating, the instruction bits were integer, and BIT 47 was
;23036  ;                unasserted.
;23037  ;       ErrorF - FRAC<55:32>=0 branch failed when bit 55 was set.
;23038  ;
;23039  ; Debug:
```

```
                ;23040  ;
                ;23041  ;       Error1: If this error occurs and the DIV I3 branch is the branch
                ;23042  ;               that failed then the logic that is likely to be at fault
                ;23043  ;               is the BRANCH 1 PAL, MUL/DIV PAL, the MUL/DIV MUX, or the
                ;23044  ;               ENB DIV or ENB MUL AND, NOT, and NAND gates. The most likely
                ;23045  ;               source of all the errors is the MUL/DIV PAL, but if the errors
                ;23046  ;               where FRAC I3 is supposed to be asserted particularly errors 9
                ;23047  ;               or 10 then the inputs and outputs to the MUX should be checked.
                ;23048  ;               If FRAC I3 is supposed to be asserted and is not then the FRAC
                ;23049  ;               CTL 3 input should be asserted. If the FRAC<55:32>=0 branch
                ;23050  ;               logic is failing then the BRANCH 2 PAL should be checked for
                ;23051  ;               error.
                ;23052  ;
                ;23053  ;       Error2 - ErrorF: Same as error 1.
                ;23054  ;
                ;23055  ;--
                ;23056  T.42:
U 25AB, B65E,15 ;23057          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;23058                                          ; STATUS WORD
U 25AC, 3E80,15 ;23059          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;23060                                          ; BIT 15 INDICATES START OF TEST TO
                ;23061                                          ; CONSOLE
U 25AD, 10E0,15 ;23062          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;23063  WAIT.T42.0:
U 25AE, 8A5A,E4 ;23064          JMP [WAIT.T42.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;23065                                          ; RESPOND
U 25AF, 887E,EC ;23066          JSR [SETUP.10]                  ;SETUP MODULE CODE, ERROR MASK, ETC.,
                ;23067                                          ; CLEAR INST AND SIZE BITS. CHECK
                ;23068                                          ; 10 BITS ONLY
U 25B0, 37A8,15 ;23069          MOV LS[T42.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 25B1, BE12,15 ;23070          MOV WR[0] TO LS[T9]
U 25B2, 8870,3C ;23071          JSR [LO]                        ;Get address of ADD SHFL FWR[FT1] TO
U 25B3, 3E10,15 ;23072          MOV WR[0] TO LS[T8]              ; FWR[FT3]
U 25B4, 8870,3C ;23073          JSR [LO]                        ;Get address of ADD SHFL FWR[FT0] TO
U 25B5, BE14,15 ;23074          MOV WR[0] TO LS[T10]             ; FWR[FT2] + FCOUT
U 25B6, 8870,3C ;23075          JSR [LO]                        ;Get address of SUB SHFL FWR[FT1] FROM
U 25B7, BE1E,15 ;23076          MOV WR[0] TO LS[T15]             ; FWR[FT3]
U 25B8, 8870,3C ;23077          JSR [LO]                        ;Get address of SUB SHFL FWR[FT0] FROM
U 25B9, BE18,15 ;23078          MOV WR[0] TO LS[T12]             ; FWR[FT2] + FCOUT
U 25BA, 8870,3C ;23079          JSR [LO]                        ;Get address of SHIFT LEFT FT0
U 25BB, 3E16,15 ;23080          MOV WR[0] TO LS[T11]
U 25BC, 8870,3C ;23081          JSR [LO]                        ;Get address of MOV FT0 TO FQ
U 25BD, 3E1C,15 ;23082          MOV WR[0] TO LS[T14]
U 25BE, 8870,3C ;23083          JSR [LO]                        ;Get address of MOV FT0 TO FT1
U 25BF, 3EAE,15 ;23084          MOV WR[0] TO LS[T57]
U 25C0, 8870,3C ;23085          JSR [LO]                        ;Get address of MOV FT0 TO FT2
U 25C1, 3EB0,15 ;23086          MOV WR[0] TO LS[T58]
U 25C2, 8870,3C ;23087          JSR [LO]                        ;Get address of MOV FT0 TO FT3
U 25C3, BEB2,15 ;23088          MOV WR[0] TO LS[T59]
U 25C4, 8870,3C ;23089          JSR [LO]                        ;Get address of branch
U 25C5, BF08,15 ;23090          MOV WR[0] TO LS[T84]
                ;23091
U 25C6, 0874,3C ;23092          JSR [CLR.INST.AND.SIZE.BITS]    ;SET INST TO NULL AND SIZE TO FLOATING
U 25C7, 8A68,EC ;23093          JSR [T42.1.3]                   ;CHECK DIV I3 BRANCH WITH INSTRUCTION
                ;23094                                          ; BITS NULL, FCOUT NOT, AND FRAC 55 R3
```

```
                ;23095                                  ; SAVE FIRST ASSERTED THEN UNASSERTED
                ;23096                                  ; SIZE=FLOATING
                ;23097
U 25C8, 3716,15 ;23098          MOV LS[#300] TO WR[0]   ;Set the size bits to double
U 25C9, C764,15 ;23099          BIS LS[BIT18] TO WR[0]
U 25CA, 4760,15 ;23100          BIS LS[BIT16] TO WR[0]
U 25CB, 90F6,15 ;23101          MISC2 [TRAP.ACC] WR[0]
U 25CC, 8A68,EC ;23102          JSR [T42.1.3]           ;CHECK DIV I3 BRANCH WITH INSTRUCTION
                ;23103                                  ; BITS NULL, FCOUT NOT, AND FRAC 55 R3
                ;23104                                  ; SAVE FIRST ASSERTED THEN UNASSERTED
                ;23105                                  ; SIZE=DOUBLE
                ;23106
U 25CD, 3716,15 ;23107          MOV LS[#300] TO WR[0]   ;Set size bits to grand
U 25CE, C764,15 ;23108          BIS LS[BIT18] TO WR[0]
U 25CF, C762,15 ;23109          BIS LS[BIT17] TO WR[0]
U 25D0, 90F6,15 ;23110          MISC2 [TRAP.ACC] WR[0]
U 25D1, 087C,1C ;23111  T42.5:  JSR [CLR.FT0]           ;LOAD FT0 WITH ZEROS
U 25D2, B616,15 ;23112          MOV LS[T11] TO WR[0]    ;Setup to SHIFT LEFT FT0
U 25D3, B716,95 ;23113          MOV LS[#300] TO WR[1]   ;Setup to loop on self
U 25D4, 90F6,15 ;23114          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25D5, 10F6,95 ;23115          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25D6, 90F6,15 ;23116          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25D7, 10F6,95 ;23117          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25D8, 90F6,15 ;23118          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25D9, 10F6,95 ;23119          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25DA, B6AE,15 ;23120          MOV LS[T57] TO WR[0]    ;Setup MOV FT0 TO FT1
U 25DB, 90F6,15 ;23121          MISC2 [TRAP.ACC] WR[0]  ;MOV FT0 TO FT1
U 25DC, 10F6,95 ;23122          MISC2 [TRAP.ACC] WR[1]  ;LOOP ON SELF
U 25DD, B616,15 ;23123          MOV LS[T11] TO WR[0]    ;Setup to SHIFT LEFT FT0
U 25DE, 90F6,15 ;23124          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25DF, 10F6,95 ;23125          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25E0, 8A6B,BC ;23126          JSR [T42.5.6]           ;GO DO MOVES, SHIFTS AND BRANCH
U 25E1, 0869,3C ;23127          JSR [CHECK.RESULT]      ;Check for error
U 25E2, 0A5D,14 ;23128          JMP [T42.5]             ;Loop on error
                ;23129
U 25E3, FF82,15 ;23130          INC LS[ERROR.NUMBER]    ;Go onto error 6
U 25E4, 087C,1C ;23131  T42.6:  JSR [CLR.FT0]           ;LOAD FT0 WITH ZEROS
U 25E5, B616,15 ;23132          MOV LS[T11] TO WR[0]    ;Setup to SHIFT LEFT FT0
U 25E6, B716,95 ;23133          MOV LS[#300] TO WR[1]   ;Setup to loop on self
U 25E7, 90F6,15 ;23134          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25E8, 10F6,95 ;23135          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25E9, 90F6,15 ;23136          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25EA, 10F6,95 ;23137          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25EB, 90F6,15 ;23138          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25EC, 10F6,95 ;23139          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25ED, 90F6,15 ;23140          MISC2 [TRAP.ACC] WR[0]  ;SHIFT LEFT FT0
U 25EE, 10F6,95 ;23141          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25EF, B6AE,15 ;23142          MOV LS[T57] TO WR[0]    ;Setup MOV FT0 TO FT1
U 25F0, 90F6,15 ;23143          MISC2 [TRAP.ACC] WR[0]  ;MOV FT0 TO FT1
U 25F1, 10F6,95 ;23144          MISC2 [TRAP.ACC] WR[1]  ;Loop on self
U 25F2, 8A6B,BC ;23145          JSR [T42.5.6]           ;GO DO MOVES, SHIFTS AND BRANCH
U 25F3, 2042,95 ;23146          INC WR[1]
U 25F4, 0869,3C ;23147          JSR [CHECK.RESULT]      ;Check for error
U 25F5, 0A5E,44 ;23148          JMP [T42.6]             ;Loop on error
                ;23149
```

```
U 25F6, FF82,15 ;23150          INC LS[ERROR.NUMBER]              ;Go onto error 7
U 25F7, 0874,3C ;23151          JSR [CLR.INST.AND.SIZE.BITS]      ;SET INST TO NULL AND SIZE TO FLOATING
U 25F8, 087C,1C ;23152  T42.7:  JSR [CLR.FT0]                     ;LOAD FT0 WITH ZEROS
U 25F9, B616,15 ;23153          MOV LS[T11] TO WR[0]              ;Setup to SHIFT LEFT FT0
U 25FA, B716,95 ;23154          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 25FB, 90F6,15 ;23155          MISC2 [TRAP.ACC] WR[0]            ;SHIFT LEFT FT0
U 25FC, 10F6,95 ;23156          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 25FD, B6AE,15 ;23157          MOV LS[T57] TO WR[0]              ;Setup MOV FT0 TO FT1
U 25FE, 90F6,15 ;23158          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT1
U 25FF, 10F6,95 ;23159          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2600, 36B2,15 ;23160          MOV LS[T59] TO WR[0]              ;Setup to MOV FT0 TO FT3
U 2601, 90F6,15 ;23161          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT3
U 2602, 10F6,95 ;23162          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2603, 8879,AC ;23163          JSR [LOAD.DOUBLE]                 ;Load data
U 2604, B716,95 ;23164          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 2605, 0A6C,EC ;23165          JSR [T42.7.8]                     ;GO DO MOVE, ADD SHFL AND BRANCH
U 2606, 0869,3C ;23166          JSR [CHECK.RESULT]                ;Check for error
U 2607, 8A5F,84 ;23167          JMP [T42.7]                       ;Loop on error
                ;23168
U 2608, FF82,15 ;23169          INC LS[ERROR.NUMBER]              ;Go onto error 8
U 2609, 087C,1C ;23170  T42.8:  JSR [CLR.FT0]                     ;LOAD FT0 WITH ZEROS
U 260A, B616,15 ;23171          MOV LS[T11] TO WR[0]              ;Setup to SHIFT LEFT FT0
U 260B, B716,95 ;23172          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 260C, 90F6,15 ;23173          MISC2 [TRAP.ACC] WR[0]            ;SHIFT LEFT FT0
U 260D, 10F6,95 ;23174          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 260E, 36B2,15 ;23175          MOV LS[T59] TO WR[0]              ;Setup to MOV FT0 TO FT3
U 260F, 90F6,15 ;23176          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT3
U 2610, 10F6,95 ;23177          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2611, 0878,FC ;23178          JSR [LOAD.TRIPLE]                 ;Load data
U 2612, B6AE,15 ;23179          MOV LS[T57] TO WR[0]              ;Setup MOV FT0 TO FT1
U 2613, B716,95 ;23180          MOV LS[#300] TO WR[1]
U 2614, 90F6,15 ;23181          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT1
U 2615, 10F6,95 ;23182          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2616, 0A6C,EC ;23183          JSR [T42.7.8]                     ;GO DO MOVE, ADD SHFL AND BRANCH
U 2617, 2042,95 ;23184          INC WR[1]
U 2618, 0869,3C ;23185          JSR [CHECK.RESULT]                ;Check for error
U 2619, 0A60,94 ;23186          JMP [T42.8]                       ;Loop on error
                ;23187
U 261A, FF82,15 ;23188          INC LS[ERROR.NUMBER]              ;Go onto error 9
U 261B, 087C,1C ;23189  T42.9:  JSR [CLR.FT0]                     ;LOAD FT0 WITH ZEROS
U 261C, B616,15 ;23190          MOV LS[T11] TO WR[0]              ;Setup to SHIFT LEFT FT0
U 261D, B716,95 ;23191          MOV LS[#300] TO WR[1]             ;Setup to loop on self
U 261E, 90F6,15 ;23192          MISC2 [TRAP.ACC] WR[0]            ;SHIFT LEFT FT0
U 261F, 10F6,95 ;23193          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2620, B6AE,15 ;23194          MOV LS[T57] TO WR[0]              ;Setup MOV FT0 TO FT1
U 2621, 90F6,15 ;23195          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT1
U 2622, 10F6,95 ;23196          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2623, B6B0,15 ;23197          MOV LS[T58] TO WR[0]              ;Setup to MOV FT0 TO FT2
U 2624, 90F6,15 ;23198          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT2
U 2625, 10F6,95 ;23199          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2626, 36B2,15 ;23200          MOV LS[T59] TO WR[0]              ;Setup to MOV FT0 TO FT3
U 2627, 90F6,15 ;23201          MISC2 [TRAP.ACC] WR[0]            ;MOV FT0 TO FT3
U 2628, 10F6,95 ;23202          MISC2 [TRAP.ACC] WR[1]            ;Loop on self
U 2629, 8879,AC ;23203          JSR [LOAD.DOUBLE]                 ;Load FT0 with BIT 55 set
U 262A, 8A6D,EC ;23204          JSR [T42.9.A.B.C]                 ;DO SUB SHL AND BRANCH
```

```
U 262B, 0869,3C ;23205          JSR [CHECK.RESULT]              ;Check for error
U 262C, 0A61,B4 ;23206          JMP [T42.9]                     ;Loop on error
                ;23207
U 262D, FF82,15 ;23208          INC LS[ERROR.NUMBER]            ;Go onto error A
U 262E, 087C,1C ;23209  T42.A:  JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 262F, B616,15 ;23210          MOV LS[T11] TO WR[0]            ;Setup to SHIFT LEFT FT0
U 2630, B716,95 ;23211          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2631, 90F6,15 ;23212          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0
U 2632, 10F6,95 ;23213          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2633, B6AE,15 ;23214          MOV LS[T57] TO WR[0]            ;Setup MOV FT0 TO FT1
U 2634, 90F6,15 ;23215          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 2635, 10F6,95 ;23216          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2636, B6B0,15 ;23217          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2637, 90F6,15 ;23218          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2638, 10F6,95 ;23219          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2639, 8879,AC ;23220          JSR [LOAD.DOUBLE]               ;Load data
U 263A, 36B2,15 ;23221          MOV LS[T59] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 263B, B716,95 ;23222          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 263C, 90F6,15 ;23223          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 263D, 10F6,95 ;23224          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 263E, 8A6D,EC ;23225          JSR [T42.9.A.B.C]               ;DO SUB SHL AND BRANCH
U 263F, 2042,95 ;23226          INC WR[1]
U 2640, 0869,3C ;23227          JSR [CHECK.RESULT]              ;Check for error
U 2641, 0A62,E4 ;23228          JMP [T42.A]                     ;Loop on error
                ;23229
U 2642, FF82,15 ;23230          INC LS[ERROR.NUMBER]            ;Go onto error B
U 2643, 087C,1C ;23231  T42.B:  JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 2644, B616,15 ;23232          MOV LS[T11] TO WR[0]            ;Setup to SHIFT LEFT FT0
U 2645, B716,95 ;23233          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2646, 90F6,15 ;23234          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0
U 2647, 10F6,95 ;23235          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2648, B6AE,15 ;23236          MOV LS[T57] TO WR[0]            ;Setup MOV FT0 TO FT1
U 2649, 90F6,15 ;23237          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 264A, 10F6,95 ;23238          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 264B, B6B0,15 ;23239          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 264C, 90F6,15 ;23240          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 264D, 10F6,95 ;23241          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 264E, 36B2,15 ;23242          MOV LS[T59] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 264F, 90F6,15 ;23243          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 2650, 10F6,95 ;23244          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2651, 8A6D,EC ;23245          JSR [T42.9.A.B.C]               ;DO SUB SHL AND BRANCH
U 2652, 2042,95 ;23246          INC WR[1]
U 2653, 0869,3C ;23247          JSR [CHECK.RESULT]              ;Check for error
U 2654, 8A64,34 ;23248          JMP [T42.B]                     ;Loop on error
                ;23249
U 2655, FF82,15 ;23250          INC LS[ERROR.NUMBER]            ;Go onto error C
U 2656, 087C,1C ;23251  T42.C:  JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 2657, 36B2,15 ;23252          MOV LS[T59] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 2658, B716,95 ;23253          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2659, 90F6,15 ;23254          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 265A, 10F6,95 ;23255          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 265B, 8879,AC ;23256          JSR [LOAD.DOUBLE]               ;Load FT0 with zeros
U 265C, B616,15 ;23257          MOV LS[T11] TO WR[0]            ;Setup to SHIFT LEFT FT0
U 265D, B716,95 ;23258          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 265E, 90F6,15 ;23259          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0
```

```
U 265F, 10F6,95 ;23260          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2660, B6AE,15 ;23261          MOV LS[T57] TO WR[0]            ;Setup MOV FT0 TO FT1
U 2661, 90F6,15 ;23262          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 2662, 10F6,95 ;23263          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2663, B6B0,15 ;23264          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2664, 90F6,15 ;23265          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2665, 10F6,95 ;23266          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2666, 8A6D,EC ;23267          JSR [T42.9.A.B.C]               ;DO SUB SHL AND BRANCH
U 2667, 0869,3C ;23268          JSR [CHECK.RESULT]              ;Check for error
U 2668, 0A65,64 ;23269          JMP [T42.C]                     ;Loop on error
                ;23270
U 2669, FF82,15 ;23271          INC LS[ERROR.NUMBER]            ;Go onto error D
U 266A, 0874,3C ;23272  T42.D:  JSR [CLR.INST.AND.SIZE.BITS]    ;SET INST TO NULL AND SIZE TO FLOATING
U 266B, E5A4,15 ;23273          CLR LS[DATA.2]                  ;Load data.2 with zeros
U 266C, B67C,15 ;23274          MOV LS[BIT30] TO WR[0]
U 266D, 3EA2,15 ;23275          MOV WR[0] TO LS[DATA.1]         ;Load data.1 with bit 31 set
U 266E, 65A6,15 ;23276          CLR LS[DATA.3]                  ;Load data.3 with zeros
U 266F, 8879,AC ;23277          JSR [LOAD.DOUBLE]               ;Load data into FT0
U 2670, 8A6F,3C ;23278          JSR [SET.INST.INTEGER]          ;SET INSTRUCTION BITS TO INTEGER
U 2671, B616,15 ;23279          MOV LS[T11] TO WR[0]            ;Setup to shift left FT0
U 2672, B716,95 ;23280          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2673, 90F6,15 ;23281          MISC2 [TRAP.ACC] WR[0]          ;Shift Left FT0
U 2674, 10F6,95 ;23282          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2675, 8A6E,BC ;23283          JSR [T42.D.E.F]                 ;GO DO MOVES AND BRANCH
U 2676, B716,95 ;23284          MOV LS[#300] TO WR[1]           ;Setup expected data
U 2677, 0869,3C ;23285          JSR [CHECK.RESULT]              ;Check for error
U 2678, 0A66,A4 ;23286          JMP [T42.D]                     ;Loop on error
                ;23287
U 2679, FF82,15 ;23288          INC LS[ERROR.NUMBER]            ;Go onto error E
U 267A, 0874,3C ;23289  T42.E:  JSR [CLR.INST.AND.SIZE.BITS]    ;SET INST TO NULL AND SIZE TO FLOATING
U 267B, 087C,1C ;23290          JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 267C, 8A6F,3C ;23291          JSR [SET.INST.INTEGER]          ;SET INSTRUCTION BITS TO INTEGER
U 267D, B616,15 ;23292          MOV LS[T11] TO WR[0]            ;Setup to shift left FT0
U 267E, B716,95 ;23293          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 267F, 90F6,15 ;23294          MISC2 [TRAP.ACC] WR[0]          ;Shift Left FT0
U 2680, 10F6,95 ;23295          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2681, 8A6E,BC ;23296          JSR [T42.D.E.F]                 ;GO DO MOVES AND BRANCH
U 2682, 3742,95 ;23297          MOV LS[#303] TO WR[1]           ;Setup expected data
U 2683, 0869,3C ;23298          JSR [CHECK.RESULT]              ;Check for error
U 2684, 8A67,A4 ;23299          JMP [T42.E]                     ;Loop on error
                ;23300
U 2685, FF82,15 ;23301          INC LS[ERROR.NUMBER]            ;Go onto error F
U 2686, 0874,3C ;23302  T42.F:  JSR [CLR.INST.AND.SIZE.BITS]    ;SET INST TO NULL AND SIZE TO FLOATING
U 2687, 087C,1C ;23303          JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 2688, 8A6F,3C ;23304          JSR [SET.INST.INTEGER]          ;SET INSTRUCTION BITS TO INTEGER
U 2689, 8A6E,BC ;23305          JSR [T42.D.E.F]                 ;GO DO MOVES AND BRANCH
U 268A, B73E,95 ;23306          MOV LS[#301] TO WR[1]           ;Setup expected data
U 268B, 0869,3C ;23307          JSR [CHECK.RESULT]              ;Check for error
U 268C, 8A68,64 ;23308          JMP [T42.F]                     ;Loop on error
                ;23309
U 268D, 0A6F,94 ;23310          JMP [T42.END]                   ;DONE
                ;23311
                ;23312  ;
                ;23313  ; These 2 subroutines used for errors 1,2,3 and 4.
                ;23314  ;
```

```
                ;23315  T42.1.3:
U 268E, 087C,1C ;23316          JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 268F, B6AE,15 ;23317          MOV LS[T57] TO WR[0]            ;Setup MOV FT0 TO FT1
U 2690, B716,95 ;23318          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2691, 90F6,15 ;23319          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 2692, 10F6,95 ;23320          MISC2 [TRAP.ACC] WR[1]
U 2693, B616,15 ;23321          MOV LS[T11] TO WR[0]            ;Setup to SHIFT LEFT FT0
U 2694, 90F6,15 ;23322          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0
U 2695, 10F6,95 ;23323          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2696, 0A6A,8C ;23324          JSR [T42.1.2.3.4]               ;GO SETUP FRAC 55 R3
                ;23325
U 2697, 0869,3C ;23326          JSR [CHECK.RESULT]              ;Check for error
U 2698, 0A68,E4 ;23327          JMP [T42.1.3]                   ;Loop on error
U 2699, FF82,15 ;23328          INC LS[ERROR.NUMBER]            ;Go onto error 2 OR 4
                ;23329  T42.2.4:
U 269A, 087C,1C ;23330          JSR [CLR.FT0]                   ;LOAD FT0 WITH ZEROS
U 269B, B616,15 ;23331          MOV LS[T11] TO WR[0]            ;Setup to SHIFT LEFT FT0
U 269C, B716,95 ;23332          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 269D, 90F6,15 ;23333          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0
U 269E, 10F6,95 ;23334          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 269F, B6AE,15 ;23335          MOV LS[T57] TO WR[0]            ;Setup MOV FT0 TO FT1
U 26A0, 90F6,15 ;23336          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 26A1, 10F6,95 ;23337          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26A2, 0A6A,8C ;23338          JSR [T42.1.2.3.4]               ;GO SETUP FRAC 55 R3
                ;23339
U 26A3, 2042,95 ;23340          INC WR[1]
U 26A4, 0869,3C ;23341          JSR [CHECK.RESULT]              ;Check for error
U 26A5, 0A69,A4 ;23342          JMP [T42.2.4]                   ;Loop on error
U 26A6, FF82,15 ;23343          INC LS[ERROR.NUMBER]            ;Go onto next error
                ;23344
U 26A7, 5B00,14 ;23345          RETURN
                ;23346
                ;23347  T42.1.2.3.4:
U 26A8, B6B0,15 ;23348          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 26A9, 90F6,15 ;23349          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 26AA, 10F6,95 ;23350          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26AB, 36B2,15 ;23351          MOV LS[T59] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 26AC, 90F6,15 ;23352          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 26AD, 10F6,95 ;23353          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26AE, B610,15 ;23354          MOV LS[T8] TO WR[0]             ;Setup to ADD SHFL
U 26AF, B614,95 ;23355          MOV LS[T10] TO WR[1]            ;Set up to ADD SHFL and branch
U 26B0, 90F6,15 ;23356          MISC2 [TRAP.ACC] WR[0]          ;ADD SHFL
U 26B1, 10F6,95 ;23357          MISC2 [TRAP.ACC] WR[1]          ;ADD SHFL
U 26B2, 10F8,15 ;23358          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 26B3, BA1A,15 ;23359          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 26B4, 3716,15 ;23360          MOV LS[#300] TO WR[0]           ;Setup to loop on self
U 26B5, 90F6,15 ;23361          MISC2 [TRAP.ACC] WR[0]          ;Loop on self
U 26B6, B61A,15 ;23362          MOV LS[T13] TO WR[0]            ;Setup received data
U 26B7, B650,95 ;23363          MOV LS[#100] TO WR[1]           ;Setup expected data
U 26B8, 474C,95 ;23364          BIS LS[#40] TO WR[1]
U 26B9, C742,95 ;23365          BIS LS[#2] TO WR[1]
U 26BA, 5B00,14 ;23366          RETURN                          ;BACK TO CHECK RESULT
                ;23367
                ;23368  ;
                ;23369  ; This subroutine used for errors 5 and 6.
```

```
                ;23370  ;
                ;23371  T42.5.6:
U 26BB, B6B0,15 ;23372          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 26BC, 90F6,15 ;23373          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 26BD, 10F6,95 ;23374          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26BE, 36B2,15 ;23375          MOV LS[T59] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 26BF, 90F6,15 ;23376          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 26C0, 10F6,95 ;23377          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26C1, B610,15 ;23378          MOV LS[T8] TO WR[0]             ;Setup to ADD SHFL
U 26C2, B614,95 ;23379          MOV LS[T10] TO WR[1]            ;Set up to ADD SHFL and branch
U 26C3, 90F6,15 ;23380          MISC2 [TRAP.ACC] WR[0]          ;ADD SHFL
U 26C4, 10F6,95 ;23381          MISC2 [TRAP.ACC] WR[1]          ;ADD SHFL
U 26C5, 10F8,15 ;23382          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 26C6, BA1A,15 ;23383          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 26C7, 3716,15 ;23384          MOV LS[#300] TO WR[0]           ;Setup to loop on self
U 26C8, 90F6,15 ;23385          MISC2 [TRAP.ACC] WR[0]          ;Loop on self
U 26C9, B61A,15 ;23386          MOV LS[T13] TO WR[0]            ;Setup received data
U 26CA, B650,95 ;23387          MOV LS[#100] TO WR[1]           ;Setup expected data
U 26CB, 474C,95 ;23388          BIS LS[#40] TO WR[1]
U 26CC, C742,95 ;23389          BIS LS[#2] TO WR[1]
U 26CD, 5B00,14 ;23390          RETURN                          ;GO CHECK RESULT
                ;23391
                ;23392  ;
                ;23393  ; This subroutine used for errors 7 and 8.
                ;23394  ;
                ;23395  T42.7.8:
U 26CE, B6B0,15 ;23396          MOV LS[T58] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 26CF, 90F6,15 ;23397          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 26D0, 10F6,95 ;23398          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 26D1, B610,15 ;23399          MOV LS[T8] TO WR[0]             ;Setup to ADD SHFL
U 26D2, B614,95 ;23400          MOV LS[T10] TO WR[1]            ;Set up to ADD SHFL and branch
U 26D3, 90F6,15 ;23401          MISC2 [TRAP.ACC] WR[0]          ;ADD SHFL
U 26D4, 10F6,95 ;23402          MISC2 [TRAP.ACC] WR[1]          ;ADD SHFL
U 26D5, 10F8,15 ;23403          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 26D6, BA1A,15 ;23404          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 26D7, 3716,15 ;23405          MOV LS[#300] TO WR[0]           ;Setup to loop on self
U 26D8, 90F6,15 ;23406          MISC2 [TRAP.ACC] WR[0]          ;Loop on self
U 26D9, B61A,15 ;23407          MOV LS[T13] TO WR[0]            ;Setup received data
U 26DA, B650,95 ;23408          MOV LS[#100] TO WR[1]           ;Setup expected data
U 26DB, 474C,95 ;23409          BIS LS[#40] TO WR[1]
U 26DC, C742,95 ;23410          BIS LS[#2] TO WR[1]
U 26DD, 5B00,14 ;23411          RETURN                          ;GO CHECK RESULT
                ;23412
                ;23413  ;
                ;23414  ; This subroutine used for errors 9, A, B and C.
                ;23415  ;
                ;23416  T42.9.A.B.C:
U 26DE, 361E,15 ;23417          MOV LS[T15] TO WR[0]            ;Setup to SUB SHFL
U 26DF, B618,95 ;23418          MOV LS[T12] TO WR[1]            ;Set up to SUB SHFL and branch
U 26E0, 90F6,15 ;23419          MISC2 [TRAP.ACC] WR[0]          ;SUB SHFL
U 26E1, 10F6,95 ;23420          MISC2 [TRAP.ACC] WR[1]          ;SUB SHFL
U 26E2, 10F8,15 ;23421          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 26E3, BA1A,15 ;23422          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 26E4, 3716,15 ;23423          MOV LS[#300] TO WR[0]           ;Setup to loop on self
U 26E5, 90F6,15 ;23424          MISC2 [TRAP.ACC] WR[0]          ;Loop on self
```

```
U 26E6, B61A,15 ;23425          MOV LS[T13] TO WR[0]            ;Setup received data
U 26E7, B650,95 ;23426          MOV LS[#100] TO WR[1]           ;Setup expected data
U 26E8, 474C,95 ;23427          BIS LS[#40] TO WR[1]
U 26E9, C742,95 ;23428          BIS LS[#2] TO WR[1]
U 26EA, 5B00,14 ;23429          RETURN                          ;CHECK RESULT
                ;23430
                ;23431  ;
                ;23432  ; This subroutine used for errors D, E and F.
                ;23433  ;
                ;23434  T42.D.E.F:
U 26EB, B61C,15 ;23435          MOV LS[T14] TO WR[0]            ;Setup to MOV FTO TO FQ
U 26EC, B708,95 ;23436          MOV LS[T84] TO WR[1]            ;Setup to branch
U 26ED, 90F6,15 ;23437          MISC2 [TRAP.ACC] WR[0]          ;MOV FTO TO FQ
U 26EE, 10F6,95 ;23438          MISC2 [TRAP.ACC] WR[1]          ;Branch
U 26EF, 10F8,15 ;23439          MISC2 [READ.ACC.UPC]            ;Enable NUA onto FPA
U 26F0, BA1A,15 ;23440          MOV FPA TO LS[T13]              ;Enable FPA onto Y
U 26F1, B61A,15 ;23441          MOV LS[T13] TO WR[0]            ;Setup received data
U 26F2, 5B00,14 ;23442          RETURN                          ;GO CHECK RESULT
                ;23443
                ;23444
                ;23445  SET.INST.INTEGER:
U 26F3, 3716,15 ;23446          MOV LS[#300] TO WR[0]           ;Set instruction bits to integer
U 26F4, C764,15 ;23447          BIS LS[BIT18] TO WR[0]
U 26F5, 475A,15 ;23448          BIS LS[BIT13] TO WR[0]
U 26F6, 475C,15 ;23449          BIS LS[BIT14] TO WR[0]
U 26F7, 90F6,15 ;23450          MISC2 [TRAP.ACC] WR[0]
U 26F8, 5B00,14 ;23451          RETURN                          ;DONE
                ;23452
                ;23453  T42.END:
```

```
;23454  .PAGE
;23455  .TOC    "TEST 43 MUL AND DIV LOGIC TEST(M8389 FPA MODULE)"
;23456  ;++
;23457  ;
;23458  ;
;23459  ; Test Description:
;23460  ;
;23461  ;       The purpose of this test is to check the MUL/DIV logic associated with
;23462  ;       the FPA. The simplest way of testing this logic is to perform a divide
;23463  ;       and perform a multiply. This will be done for the five cases of
;23464  ;       multiply, Integer, Single, Double and Grand, EMOD, EMOD Default. There
;23465  ;       are four cases of divide, DIVF, DIVG and DIVD, DIVH, and DIV L. This
;23466  ;       will mean that four different divides will be performed to test the DIV
;23467  ;       logic. If there is a failure then error can be isolated down to either
;23468  ;       the MUL/DIV MUX or the MUL/DIV PAL.
;23469  ;
;23470  ; Assumptions:
;23471  ;
;23472  ;       It is assumed that all of the 2901 logic has been tested is working.
;23473  ;
;23474  ; Test Steps:
;23475  ;
;23476  ;       1) Load FQ with bit 32 set, FT2 and FT4 with bit 55 set and the size
;23477  ;          and instruction bits set to single and poly respectively. Force
;23478  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23479  ;          and check for error.
;23480  ;       2) Repeat step 1 with FT4 loaded with zeros.
;23481  ;       3) Repeat step 1 with FQ set to zero.
;23482  ;       4) Load FQ with bit 00 set, FT2 and FT4 with bit 55 set and the size
;23483  ;          and instruction bits set to double and poly respectively. Force
;23484  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23485  ;          and check for error.
;23486  ;       5) Repeat step 4 with FT4 loaded with zeros.
;23487  ;       6) Repeat step 4 with FQ loaded with zeros.
;23488  ;       7) Load FQ with bit 00 set, FT2 and FT4 with bit 55 set and the size
;23489  ;          and instruction bits set to grand and poly respectively. Force
;23490  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23491  ;          and check for error.
;23492  ;       8) Repeat step 7 with FT4 loaded with zeros.
;23493  ;       9) Repeat step 7 with FQ loaded with zeros.
;23494  ;      10) Load FQ with bit EXT00 set, FT2 and FT4 with bit 55 set and the size
;23495  ;          and instruction bits set to huge and poly respectively. Force
;23496  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23497  ;          and check for error.
;23498  ;      11) Repeat step 10 with FT4 loaded with zeros.
;23499  ;      12) Repeat step 10 with FQ loaded with zeros.
;23500  ;      13) Load FQ with bit 32 set, FT2 and FT4 with bit 55 set and the size
;23501  ;          and instruction bits set to single and mul respectively. Force
;23502  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23503  ;          and check for error.
;23504  ;      14) Repeat step 13 with FT4 loaded with zeros.
;23505  ;      15) Repeat step 13 with FQ loaded with zeros.
;23506  ;      16) Load FQ with bit 32 set, FT2 and FT4 with bit 55 set and the size
;23507  ;          and instruction bits set to single and emod respectively. Force
;23508  ;          a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
```

```
;23509  ;          and check for error.
;23510  ;       17) Repeat step 16 with FT4 loaded with zeros.
;23511  ;       18) Repeat step 16 with FQ loaded with zeros.
;23512  ;       19) Load FQ with bit 32 set, FT2 and FT4 with bit 55 set and the size
;23513  ;           and instruction bits set to single and add respectively. Force
;23514  ;           a MUL ADD FWR[FT2] TO FWR[FT4], then store the contents of FT4
;23515  ;           and check for error.
;23516  ;       20) Repeat step 19 with FT4 loaded with zeros.
;23517  ;       21) Repeat step 19 with FQ loaded with zeros.
;23518  ;       22) Load FQ with zeros, set the instruction and size bits to div and
;23519  ;           single, load FT2 with .1 and FT0 with .11. Then force the condi-
;23520  ;           tional divide. Store the contents of FT2 and checkc for error.
;23521  ;       23) Repeat step 22 with FT2 loaded with .11 and FT0 loaded with .1.
;23522  ;       24) Repeat step 22 with FT2 loaded with .1 and FT0 loaded with .1.
;23523  ;       25) Repeat step 22 with FT2 loaded with .11 and FT0 loaded with .01.
;23524  ;       26) Repeat steps 22 - 25 with the size bits set to double.
;23525  ;       27) Repeat steps 22 - 25 with the size btis set to grand.
;23526  ;       28) Set the instruction and size bits to divl and single, FQ to zero,
;23527  ;           FT2 to 010 and FT0 to 010. The force the conditional divide for
;23528  ;           integer. Store the contents of FT2 and check for error.
;23529  ;       29) Repeat step 28 with FT2 loaded with 011 and FT0 loaded with 001.
;23530  ;       30) Repeat step 28 with FT2 loaded with 010 and FT0 loaded with 011.
;23531  ;       31) For the following 6 cases FT1, FT3, and FQ should be loaded with
;23532  ;           zeros. Load FT0 and FT2 with .11 set the instruction bits to div
;23533  ;           and the soze bits to huge. Force the ADD.LOOP code in the FPA and
;23534  ;           wait 6 cycles, then loop self. Store the data in FQ to the CPU and
;23535  ;           check for error.
;23536  ;       32) Repeat step 31 except force the SUB.LOOP instead of ADD.LOOP.
;23537  ;       33) Load FT0 with .11, FT2 with .101, and set the instruction bits to
;23538  ;           div and the size bits to huge. Force the ADD.LOOP code in the FPA
;23539  ;           and wait 6 cycles, then loop self. Store the data in FQ to the CPU
;23540  ;           and check for error.
;23541  ;       34) Repeat step 33 except force SUB.LOOP instead of ADD.LOOP.
;23542  ;       35) Load FT0 with .0101, FT2 with .101, and set the instruction bits to
;23543  ;           div and the size bits to huge. Force the ADD.LOOP code in the FPA
;23544  ;           and wait 6 cycles, then loop self. Store the data in FQ to the CPU
;23545  ;           and check for error.
;23546  ;       36) Load FT0 and FT2 with .01 set the instruction bits to div and the
;23547  ;           size bits to huge. Force the ADD.LOOP code in the FPA and  wait 6
;23548  ;           cycles, then loop self. Store the data in FQ to the CPU and check
;23549  ;           for error.
;23550  ;
;23551  ; Error:
;23552  ;
;23553  ;       Error1 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23554  ;                FT2 and FT4 had .1 in them and the instruction bits were poly
;23555  ;       Error2 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23556  ;                and FT2 had .1 and FT4 had .01 and the instruction bits were
;23557  ;                poly.
;23558  ;       Error3 - MUL ADD failed when size bits were single, FQ had zeros in it
;23559  ;                and FT2 and FT4 had .1 in it and the instruction bits were
;23560  ;                poly.
;23561  ;       Error4 - MUL ADD failed when size bits were double, FQ had bit 00 set
;23562  ;                FT2 and FT4 had .1 in them and the instruction bits were poly.
;23563  ;       Error5 - MUL ADD failed when size bits were double, FQ had bit 00 set
```

```
;23564  ;                and FT2 had .1 and FT4 had .01 and the instruction bits were
;23565  ;                poly.
;23566  ;       Error6 - MUL ADD failed when size bits were double, FQ had zeros in it
;23567  ;                and FT2 and FT4 had .1 in it and the instruction bits were
;23568  ;                poly,
;23569  ;       Error7 - MUL ADD failed when size bits were grand, FQ had bit 00 set
;23570  ;                FT2 and FT4 had .1 in them and the instruction bits were poly.
;23571  ;       Error8 - MUL ADD failed when size bits were grand, FQ had bit 00 set
;23572  ;                and FT2 had .1 and FT4 had .01 and the instruction bits were
;23573  ;                poly.
;23574  ;       Error9 - MUL ADD failed when size bits were grand, FQ had zeros in it
;23575  ;                and FT2 and FT4 had .1 in it and the instruction bits were
;23576  ;                poly.
;23577  ;       ErrorA - MUL ADD failed when size bits were huge, FQ had bit 32 set
;23578  ;                FT2 and FT4 had .1 in them and the instruction bits were
;23579  ;                poly.
;23580  ;       ErrorB - MUL ADD failed when size bits were huge, FQ had bit 32 set
;23581  ;                and FT2 had .1 and FT4 had .01 and the instruction bits were
;23582  ;                poly.
;23583  ;       ErrorC - MUL ADD failed when size bits were huge, FQ had zeros in it
;23584  ;                and FT2 and FT4 had .1 in it and the instruction bits were
;23585  ;                poly.
;23586  ;       ErrorD - MUL ADD failed when size bits were single, FQ had bit 32 set
;23587  ;                FT2 and FT4 had .1 in them and the instruction bits were mul.
;23588  ;       ErrorE - MUL ADD failed when size bits were single, FQ had bit 32 set
;23589  ;                and FT2 had .1 and FT4 had .01 and the instruction bits were
;23590  ;                mul.
;23591  ;       ErrorF - MUL ADD failed when size bits were single, FQ had zeros in it
;23592  ;                and FT2 and FT4 had .1 in it and the instruction bits were
;23593  ;                mul.
;23594  ;       Error10 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23595  ;                 FT2 and FT4 had .1 in them and the instruction bits were emod
;23596  ;       Error11 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23597  ;                 and FT2 had .1 and FT4 had .01 and the instruction bits were
;23598  ;                 emod.
;23599  ;       Error12 - MUL ADD failed when size bits were single, FQ had zeros in it
;23600  ;                 and FT2 and FT4 had .1 in it and the instruction bits were
;23601  ;                 emod.
;23602  ;       Error13 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23603  ;                 FT2 and FT4 had .1 in them and the instruction bits were add.
;23604  ;       Error14 - MUL ADD failed when size bits were single, FQ had bit 32 set
;23605  ;                 and FT2 had .1 and FT4 had .01 and the instruction bits were
;23606  ;                 add.
;23607  ;       Error15 - MUL ADD failed when size bits were single, FQ had zeros in it
;23608  ;                 and FT2 and FT4 had .1 in it and the instruction bits were
;23609  ;                 add.
;23610  ;       Error16 - MUL ADD failed when the instruction bits were mull and the
;23611  ;                 size bits were single and FT2 and FT4 had .1 in them.
;23612  ;       Error17 - MUL ADD failed when the instruction bits were mull and the
;23613  ;                 size bits were single and FT2 and FT4 had .11 in them.
;23614  ;       Error18 - MUL ADD failed when the instruciton bits were mull and the
;23615  ;                 size bits were single and FT2 and FT4 had .01 in them.
;23616  ;       Error19 - MUL ADD failed when the instruciton bits were mull and the
;23617  ;                 size bits were single and FT2 had .11 and FT4 had .01 in them
;23618  ;       Error1A - Conditional divide step failed when FT2 was .1, FT0 was .11,
```

```
;23619  ;                the instruction bits were div, and the size bits were single.
;23620  ;        Error1B - Conditional divide step failed when FT2 was .11, FT0 was .1,
;23621  ;                the instruction bits were div, and the size bits were single.
;23622  ;        Error1C - Conditional divide step failed when FT2 was .1, FT0 was .1,
;23623  ;                the instruction bits were div, and the size bits were single.
;23624  ;        Error1D - Conditional divide step failed when FT2 was .11, FT0 was .01,
;23625  ;                the instruction bits were div, and the size bits were single.
;23626  ;        Error1E - Conditional divide step failed when FT2 was .1, FT0 was .11,
;23627  ;                the instruction bits were div, and the size bits were double.
;23628  ;        Error1F - Conditional divide step failed when FT2 was .11, FT0 was .1,
;23629  ;                the instruction bits were div, and the size bits were double.
;23630  ;        Error20 - Conditional divide step failed when FT2 was .1, FT0 was .1,
;23631  ;                the instruction bits were div, and the size bits were double.
;23632  ;        Error21 - Conditional divide step failed when FT2 was .11, FT0 was .01,
;23633  ;                the instruction bits were div, and the size bits were double.
;23634  ;        Error22 - Conditional divide step failed when FT2 was .1, FT0 was .11,
;23635  ;                the instruction bits were div, and the size bits were grand.
;23636  ;        Error23 - Conditional divide step failed when FT2 was .11, FT0 was .1,
;23637  ;                the instruction bits were div, and the size bits were grand.
;23638  ;        Error24 - Conditional divide step failed when FT2 was .1, FT0 was .1,
;23639  ;                the instruction bits were div, and the size bits were grand.
;23640  ;        Error25 - Conditional divide step failed when FT2 was .11, FT0 was .01,
;23641  ;                the instruction bits were div, and the size bits were grand.
;23642  ;        Error26 - Conditional divide failed when the instriction bits were divl
;23643  ;                the size bits were single, FT2 was 01, and FT0 was 01.
;23644  ;        Error27 - Conditional divide failed when the instruction bits were divl
;23645  ;                the size bits were single, FT2 was 011, and FT0 was 001.
;23646  ;        Error28 - Conditional divide failed when the instruction bits were divl
;23647  ;                the size bits were single, FT2 was 010, and FT0 was 011.
;23648  ;        Error29 - HUGE DIV failed when FT2 and FT0 were .11 and ADD.LOOP was
;23649  ;                forced.
;23650  ;        Error2A - HUGE DIV failed when FT2 and FT0 were .11 and SUB.LOOP was
;23651  ;                forced.
;23652  ;        Error2B - HUGE DIV failed when FT2 was .101 and FT0 was .11 and ADD
;23653  ;                LOOP was forced.
;23654  ;        Error2C - HUGE DIV failed when FT2 was .101 and FT0 was .11 and SUB
;23655  ;                LOOP was forced.
;23656  ;        Error2D - HUGE DIV failed when FT2 was .101 and FT0 was .0101 and
;23657  ;                ADD.LOOP was forced.
;23658  ;        Error2E - HUGE DIV failed when FT2 and FT0 were .01 and ADD.LOOP was
;23659  ;                forced.
;23660  ;
;23661  ; Debug:
;23662  ;
;23663  ;        Error1: If this error occurs then there are two likely causes of error.
;23664  ;                The two likely sources of error are either the MUL/DIV MUX or
;23665  ;                the MUL/DIV PAL. If all of the multiply tests or all of the
;23666  ;                divide tests are failing it's likely that the MUX is the cause
;23667  ;                of the error. If a sigle test is failing it's likely that the
;23668  ;                PAL is the cause of the error.
;23669  ;        Error2 - Error2E: Same as error 1.
;23670  ;
;23671  ;
;23672  ;--
;23673  T.43:
```

```
U 26F9, B65E,15 ;23674          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND
                ;23675                                          ; STATUS WORD
U 26FA, 3E80,15 ;23676          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;23677                                          ; BIT 15 INDICATES START OF TEST TO
                ;23678                                          ; CONSOLE
U 26FB, 10E0,15 ;23679          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;23680  WAIT.T43.0:
U 26FC, 0A6F,C4 ;23681          JMP [WAIT.T43.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;23682                                          ; RESPOND
U 26FD, 087E,6C ;23683          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;23684                                          ; AND SIZE BITS
U 26FE, B670,15 ;23685          MOV LS[OTHER.DATA] TO WR[0]     ;set up the other data field
U 26FF, 3E80,15 ;23686          MOV WR[0] TO LS[CONTROL.STATUS]
U 2700, 365F,15 ;23687          MOV LS[BIT15] TO WR[2]          ;Setup first float mask
U 2701, B725,95 ;23688          MOV LS[#FFFF00FF] TO WR[3]      ;Setup third float mask
U 2702, B7AA,15 ;23689          MOV LS[T43.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2703, BE12,15 ;23690          MOV WR[0] TO LS[T9]
U 2704, 8870,3C ;23691          JSR [L0]                        ;Get address of MOV FT0 TO FT2
U 2705, BE18,15 ;23692          MOV WR[0] TO LS[T12]
U 2706, 8870,3C ;23693          JSR [L0]                        ;Set address of MOV FT0 TO FT4
U 2707, 3E1C,15 ;23694          MOV WR[0] TO LS[T14]
U 2708, 8870,3C ;23695          JSR [L0]                        ;Get address of MOV FT0 TO FQ
U 2709, 3E10,15 ;23696          MOV WR[0] TO LS[T8]
U 270A, 8870,3C ;23697          JSR [L0]                        ;Get address of MUL ADD FT2 TO FT4
U 270B, BE1E,15 ;23698          MOV WR[0] TO LS[T15]
U 270C, 8870,3C ;23699          JSR [L0]                        ;Get address of Store FF FT4
U 270D, 3EAE,15 ;23700          MOV WR[0] TO LS[T57]
U 270E, 8870,3C ;23701          JSR [L0]                        ;Get address of Store SF FT4
U 270F, 3EB0,15 ;23702          MOV WR[0] TO LS[T58]
U 2710, 8870,3C ;23703          JSR [L0]                        ;Get address of Store TF FT4
U 2711, BEB2,15 ;23704          MOV WR[0] TO LS[T59]
U 2712, 8870,3C ;23705          JSR [L0]                        ;Get address of Shift left FT0
U 2713, BE14,15 ;23706          MOV WR[0] TO LS[T10]
U 2714, 8870,3C ;23707          JSR [L0]                        ;Get address of Shift right FT4
U 2715, BF08,15 ;23708          MOV WR[0] TO LS[T84]
U 2716, 8870,3C ;23709          JSR [L0]                        ;Get address of CLR FT0
U 2717, BF8C,15 ;23710          MOV WR[0] TO LS[TC5]
U 2718, 8870,3C ;23711          JSR [L0]                        ;Get address of SHIFT FT0 LEFT IN[ONE]
U 2719, 3F90,15 ;23712          MOV WR[0] TO LS[TC8]
U 271A, 3716,15 ;23713          MOV LS[#300] TO WR[0]           ;Set the size and instruction bits
U 271B, C758,15 ;23714          BIS LS[BIT12] TO WR[0]          ; to single and poly
U 271C, C764,15 ;23715          BIS LS[BIT18] TO WR[0]
U 271D, 90F6,15 ;23716          MISC2 [TRAP.ACC] WR[0]
U 271E, E5A2,15 ;23717  T43.1:  CLR LS[DATA.1]                  ;Setup data for load
U 271F, B65E,15 ;23718          MOV LS[BIT15] TO WR[0]
U 2720, 3EA4,15 ;23719          MOV WR[0] TO LS[DATA.2]
U 2721, 65A6,15 ;23720          CLR LS[DATA.3]
U 2722, 0878,FC ;23721          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2723, 3614,15 ;23722          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 2724, 90F6,15 ;23723          MISC2 [TRAP.ACC] WR[0]
U 2725, B610,15 ;23724          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2726, B716,95 ;23725          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2727, 90F6,15 ;23726          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2728, 10F6,95 ;23727          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2729, E5A4,15 ;23728          CLR LS[DATA.2]                  ;Set up data for load
```

```
U 272A, 0878,FC ;23729          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 272B, 0878,AC ;23730          JSR [MOV.DATA]                 ;MOV FT0 TO FT4
U 272C, 3618,15 ;23731          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT4
U 272D, B716,95 ;23732          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 272E, 90F6,15 ;23733          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT4
U 272F, 10F6,95 ;23734          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2730, 0AD1,1C ;23735          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 2731, 3708,15 ;23736          MOV LS[T84] TO WR[0]           ;Setup to shift right FT4
U 2732, 90F6,15 ;23737          MISC2 [TRAP.ACC] WR[0]         ;Shift right FT4
U 2733, 10F6,95 ;23738          MISC2 [TRAP.ACC] WR[1]
U 2734, 087A,BC ;23739          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 2735, 364C,95 ;23740          MOV LS[BIT6] TO WR[1]          ;Setup expected data
U 2736, BEA2,95 ;23741          MOV WR[1] TO LS[DATA.1]
U 2737, E5A4,15 ;23742          CLR LS[DATA.2]
U 2738, 65A6,15 ;23743          CLR LS[DATA.3]
U 2739, 887C,6C ;23744          JSR [CHECK.SUB]
U 273A, 8A71,E4 ;23745          JMP [T43.1]                    ;Loop on error
U 273B, FF82,15 ;23746          INC LS[ERROR.NUMBER]           ;Go onto error 2
U 273C, E5A2,15 ;23747  T43.2:  CLR LS[DATA.1]                 ;Setup data for load
U 273D, B65E,15 ;23748          MOV LS[BIT15] TO WR[0]
U 273E, 3EA4,15 ;23749          MOV WR[0] TO LS[DATA.2]
U 273F, 65A6,15 ;23750          CLR LS[DATA.3]
U 2740, 0878,FC ;23751          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2741, 3614,15 ;23752          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 2742, 90F6,15 ;23753          MISC2 [TRAP.ACC] WR[0]
U 2743, B610,15 ;23754          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 2744, B716,95 ;23755          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 2745, 90F6,15 ;23756          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2746, 10F6,95 ;23757          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2747, E5A4,15 ;23758          CLR LS[DATA.2]                 ;Set up data for load
U 2748, 0878,FC ;23759          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2749, 3618,15 ;23760          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 274A, B716,95 ;23761          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 274B, 90F6,15 ;23762          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 274C, 10F6,95 ;23763          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 274D, 378C,15 ;23764          MOV LS[TC5] TO WR[0]           ;Setup to CLR FT0
U 274E, 90F6,15 ;23765          MISC2 [TRAP.ACC] WR[0]
U 274F, 10F6,95 ;23766          MISC2 [TRAP.ACC] WR[1]
U 2750, 0878,AC ;23767          JSR [MOV.DATA]
U 2751, 0AD1,1C ;23768          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 2752, 087A,BC ;23769          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 2753, 364C,95 ;23770          MOV LS[BIT6] TO WR[1]          ;Setup expected data
U 2754, BEA2,95 ;23771          MOV WR[1] TO LS[DATA.1]
U 2755, E5A4,15 ;23772          CLR LS[DATA.2]
U 2756, 65A6,15 ;23773          CLR LS[DATA.3]
U 2757, 887C,6C ;23774          JSR [CHECK.SUB]
U 2758, 8A73,C4 ;23775          JMP [T43.2]                    ;Loop on error
U 2759, FF82,15 ;23776          INC LS[ERROR.NUMBER]           ;Go on to error 3
U 275A, 087C,1C ;23777  T43.3:  JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 275B, 3614,15 ;23778          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 275C, 90F6,15 ;23779          MISC2 [TRAP.ACC] WR[0]
U 275D, B610,15 ;23780          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 275E, B716,95 ;23781          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 275F, 90F6,15 ;23782          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2760, 10F6,95 ;23783          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
```

```
U 2761, E5A4,15 ;23784          CLR LS[DATA.2]                  ;Set up data for load
U 2762, 0878,FC ;23785          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2763, 0878,AC ;23786          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2764, 3618,15 ;23787          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 2765, B716,95 ;23788          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2766, 90F6,15 ;23789          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 2767, 10F6,95 ;23790          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2768, 0AD1,1C ;23791          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2769, 087A,BC ;23792          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 276A, 364C,95 ;23793          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 276B, BEA2,95 ;23794          MOV WR[1] TO LS[DATA.1]
U 276C, E5A4,15 ;23795          CLR LS[DATA.2]
U 276D, 65A6,15 ;23796          CLR LS[DATA.3]
U 276E, 887C,6C ;23797          JSR [CHECK.SUB]
U 276F, 8A75,A4 ;23798          JMP [T43.3]                     ;Loop on error
U 2770, FF82,15 ;23799          INC LS[ERROR.NUMBER]            ;Go onto error 4
U 2771, 3716,15 ;23800          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2772, 4760,15 ;23801          BIS LS[BIT16] TO WR[0]          ; poly and double
U 2773, C764,15 ;23802          BIS LS[BIT18] TO WR[0]
U 2774, C758,15 ;23803          BIS LS[BIT12] TO WR[0]
U 2775, 90F6,15 ;23804          MISC2 [TRAP.ACC] WR[0]
U 2776, E5A2,15 ;23805 T43.4:   CLR LS[DATA.1]                  ;Setup data for load
U 2777, E5A4,15 ;23806          CLR LS[DATA.2]
U 2778, B65E,15 ;23807          MOV LS[BIT15] TO WR[0]
U 2779, BEA6,15 ;23808          MOV WR[0] TO LS[DATA.3]
U 277A, 0878,FC ;23809          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 277B, 3614,15 ;23810          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 277C, 90F6,15 ;23811          MISC2 [TRAP.ACC] WR[0]
U 277D, B610,15 ;23812          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 277E, B716,95 ;23813          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 277F, 90F6,15 ;23814          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2780, 10F6,95 ;23815          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2781, 65A6,15 ;23816          CLR LS[DATA.3]                  ;Set up data for load
U 2782, 0878,FC ;23817          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2783, 0878,AC ;23818          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2784, 3618,15 ;23819          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 2785, B716,95 ;23820          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2786, 90F6,15 ;23821          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 2787, 10F6,95 ;23822          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2788, 0AD1,1C ;23823          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2789, 3708,15 ;23824          MOV LS[T84] TO WR[0]            ;Setup to shift FT4 right
U 278A, 90F6,15 ;23825          MISC2 [TRAP.ACC] WR[0]          ;Shift right FT4
U 278B, 10F6,95 ;23826          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 278C, 087A,BC ;23827          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 278D, 364C,95 ;23828          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 278E, BEA2,95 ;23829          MOV WR[1] TO LS[DATA.1]
U 278F, E5A4,15 ;23830          CLR LS[DATA.2]
U 2790, 65A6,15 ;23831          CLR LS[DATA.3]
U 2791, 887C,6C ;23832          JSR [CHECK.SUB]
U 2792, 0A77,64 ;23833          JMP [T43.4]                     ;Loop on error
U 2793, FF82,15 ;23834          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 2794, E5A2,15 ;23835 T43.5:   CLR LS[DATA.1]                  ;Setup data for load
U 2795, E5A4,15 ;23836          CLR LS[DATA.2]
U 2796, B65E,15 ;23837          MOV LS[BIT15] TO WR[0]
U 2797, BEA6,15 ;23838          MOV WR[0] TO LS[DATA.3]
```

```
U 2798, 0878,FC ;23839          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2799, 3614,15 ;23840          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 279A, 90F6,15 ;23841          MISC2 [TRAP.ACC] WR[0]
U 279B, B610,15 ;23842          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 279C, B716,95 ;23843          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 279D, 90F6,15 ;23844          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 279E, 10F6,95 ;23845          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 279F, 65A6,15 ;23846          CLR LS[DATA.3]                 ;Set up data for load
U 27A0, 0878,FC ;23847          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 27A1, 3618,15 ;23848          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 27A2, B716,95 ;23849          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 27A3, 90F6,15 ;23850          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 27A4, 10F6,95 ;23851          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 27A5, 378C,15 ;23852          MOV LS[TC5] TO WR[0]           ;Setup to CLR FT4
U 27A6, 90F6,15 ;23853          MISC2 [TRAP.ACC] WR[0]         ;CLR FT4
U 27A7, 10F6,95 ;23854          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 27A8, 0878,AC ;23855          JSR [MOV.DATA]                 ;MOV FT0 TO FT4
U 27A9, 0AD1,1C ;23856          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 27AA, 087A,BC ;23857          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 27AB, 364C,95 ;23858          MOV LS[BIT6] TO WR[1]          ;Setup expected data
U 27AC, BEA2,95 ;23859          MOV WR[1] TO LS[DATA.1]
U 27AD, E5A4,15 ;23860          CLR LS[DATA.2]
U 27AE, 65A6,15 ;23861          CLR LS[DATA.3]
U 27AF, 887C,6C ;23862          JSR [CHECK.SUB]
U 27B0, 0A79,44 ;23863          JMP [T43.5]                    ;Loop on error
U 27B1, FF82,15 ;23864          INC LS[ERROR.NUMBER]           ;Go on to error 6
U 27B2, 087C,1C ;23865  T43.6:  JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 27B3, 3614,15 ;23866          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 27B4, 90F6,15 ;23867          MISC2 [TRAP.ACC] WR[0]
U 27B5, B610,15 ;23868          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 27B6, B716,95 ;23869          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 27B7, 90F6,15 ;23870          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 27B8, 10F6,95 ;23871          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 27B9, E5A4,15 ;23872          CLR LS[DATA.2]                 ;Set up data for load
U 27BA, 0878,FC ;23873          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 27BB, 0878,AC ;23874          JSR [MOV.DATA]                 ;MOV FT0 TO FT4
U 27BC, 3618,15 ;23875          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT4
U 27BD, B716,95 ;23876          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 27BE, 90F6,15 ;23877          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT4
U 27BF, 10F6,95 ;23878          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 27C0, 0AD1,1C ;23879          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 27C1, 087A,BC ;23880          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 27C2, 364C,95 ;23881          MOV LS[BIT6] TO WR[1]          ;Setup expected data
U 27C3, BEA2,95 ;23882          MOV WR[1] TO LS[DATA.1]
U 27C4, E5A4,15 ;23883          CLR LS[DATA.2]
U 27C5, 65A6,15 ;23884          CLR LS[DATA.3]
U 27C6, 887C,6C ;23885          JSR [CHECK.SUB]
U 27C7, 8A7B,24 ;23886          JMP [T43.6]                    ;Loop on error
U 27C8, FF82,15 ;23887          INC LS[ERROR.NUMBER]           ;Go onto error 7
U 27C9, 3716,15 ;23888          MOV LS[#300] TO WR[0]          ;Set instruction and size bits to
U 27CA, C764,15 ;23889          BIS LS[BIT18] TO WR[0]         ; to poly and single
U 27CB, C758,15 ;23890          BIS LS[BIT12] TO WR[0]
U 27CC, 90F6,15 ;23891          MISC2 [TRAP.ACC] WR[0]
U 27CD, E5A2,15 ;23892  T43.7:  CLR LS[DATA.1]                 ;Setup data for load
U 27CE, E5A4,15 ;23893          CLR LS[DATA.2]
```

```
U 27CF, B65E,15 ;23894          MOV LS[BIT15] TO WR[0]
U 27D0, BEA6,15 ;23895          MOV WR[0] TO LS[DATA.3]
U 27D1, 0878,FC ;23896          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 27D2, 3614,15 ;23897          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 27D3, 90F6,15 ;23898          MISC2 [TRAP.ACC] WR[0]
U 27D4, B610,15 ;23899          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 27D5, B716,95 ;23900          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 27D6, 90F6,15 ;23901          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 27D7, 10F6,95 ;23902          MISC2 [TRAP.ACC] WR[1]
U 27D8, 65A6,15 ;23903          CLR LS[DATA.3]                  ;Setup data
U 27D9, 0878,FC ;23904          JSR [LOAD.TRIPLE]               ;Load data
U 27DA, 0878,AC ;23905          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 27DB, 3618,15 ;23906          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 27DC, B716,95 ;23907          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 27DD, 90F6,15 ;23908          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 27DE, 10F6,95 ;23909          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 27DF, 3716,15 ;23910          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 27E0, C764,15 ;23911          BIS LS[BIT18] TO WR[0]          ; to poly and grand
U 27E1, C762,15 ;23912          BIS LS[BIT17] TO WR[0]
U 27E2, C758,15 ;23913          BIS LS[BIT12] TO WR[0]
U 27E3, 90F6,15 ;23914          MISC2 [TRAP.ACC] WR[0]
U 27E4, 0AD1,1C ;23915          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 27E5, 3716,15 ;23916          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 27E6, C764,15 ;23917          BIS LS[BIT18] TO WR[0]          ; to poly and single
U 27E7, C758,15 ;23918          BIS LS[BIT12] TO WR[0]
U 27E8, 90F6,15 ;23919          MISC2 [TRAP.ACC] WR[0]
U 27E9, 3708,15 ;23920          MOV LS[T84] TO WR[0]            ;Setup to shift FT4 right
U 27EA, 90F6,15 ;23921          MISC2 [TRAP.ACC] WR[0]          ;Shift right FT4
U 27EB, 10F6,95 ;23922          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 27EC, 087A,BC ;23923          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 27ED, 364C,95 ;23924          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 27EE, BEA2,95 ;23925          MOV WR[1] TO LS[DATA.1]
U 27EF, E5A4,15 ;23926          CLR LS[DATA.2]
U 27F0, 65A6,15 ;23927          CLR LS[DATA.3]
U 27F1, 887C,6C ;23928          JSR [CHECK.SUB]
U 27F2, 0A7C,D4 ;23929          JMP [T43.7]                     ;Loop on error
U 27F3, FF82,15 ;23930          INC LS[ERROR.NUMBER]            ;Go onto error 8
U 27F4, E5A2,15 ;23931  T43.8:  CLR LS[DATA.1]                  ;Setup data for load
U 27F5, E5A4,15 ;23932          CLR LS[DATA.2]
U 27F6, B65E,15 ;23933          MOV LS[BIT15] TO WR[0]
U 27F7, BEA6,15 ;23934          MOV WR[0] TO LS[DATA.3]
U 27F8, 0878,FC ;23935          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 27F9, 3614,15 ;23936          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 27FA, 90F6,15 ;23937          MISC2 [TRAP.ACC] WR[0]
U 27FB, B610,15 ;23938          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 27FC, B716,95 ;23939          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 27FD, 90F6,15 ;23940          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 27FE, 10F6,95 ;23941          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 27FF, 65A6,15 ;23942          CLR LS[DATA.3]
U 2800, 0878,FC ;23943          JSR [LOAD.TRIPLE]
U 2801, 3618,15 ;23944          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2802, B716,95 ;23945          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2803, 90F6,15 ;23946          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2804, 10F6,95 ;23947          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2805, 378C,15 ;23948          MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
```

```
U 2806, 90F6,15 ;23949          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2807, 0878,AC ;23950          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2808, 3716,15 ;23951          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2809, C764,15 ;23952          BIS LS[BIT18] TO WR[0]          ; to poly and grand
U 280A, C762,15 ;23953          BIS LS[BIT17] TO WR[0]
U 280B, C758,15 ;23954          BIS LS[BIT12] TO WR[0]
U 280C, 90F6,15 ;23955          MISC2 [TRAP.ACC] WR[0]
U 280D, 0AD1,1C ;23956          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 280E, 3716,15 ;23957          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 280F, C764,15 ;23958          BIS LS[BIT18] TO WR[0]          ; to poly and single
U 2810, C758,15 ;23959          BIS LS[BIT12] TO WR[0]
U 2811, 90F6,15 ;23960          MISC2 [TRAP.ACC] WR[0]
U 2812, 087A,BC ;23961          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2813, 364C,95 ;23962          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2814, BEA2,95 ;23963          MOV WR[1] TO LS[DATA.1]
U 2815, E5A4,15 ;23964          CLR LS[DATA.2]
U 2816, 65A6,15 ;23965          CLR LS[DATA.3]
U 2817, 887C,6C ;23966          JSR [CHECK.SUB]
U 2818, 0A7F,44 ;23967          JMP [T43.8]                     ;Loop on error
U 2819, FF82,15 ;23968          INC LS[ERROR.NUMBER]            ;Go on to error 9
U 281A, 378C,15 ;23969  T43.9:  MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 281B, B716,95 ;23970          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 281C, 90F6,15 ;23971          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 281D, 10F6,95 ;23972          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 281E, B610,15 ;23973          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 281F, B716,95 ;23974          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2820, 90F6,15 ;23975          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2821, 10F6,95 ;23976          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2822, 087C,1C ;23977          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 2823, 0878,AC ;23978          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2824, 3618,15 ;23979          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 2825, B716,95 ;23980          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2826, 90F6,15 ;23981          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 2827, 10F6,95 ;23982          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2828, 3716,15 ;23983          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2829, C764,15 ;23984          BIS LS[BIT18] TO WR[0]          ; to poly and grand
U 282A, C762,15 ;23985          BIS LS[BIT17] TO WR[0]
U 282B, C758,15 ;23986          BIS LS[BIT12] TO WR[0]
U 282C, 90F6,15 ;23987          MISC2 [TRAP.ACC] WR[0]
U 282D, 0AD1,1C ;23988          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 282E, 3716,15 ;23989          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 282F, C764,15 ;23990          BIS LS[BIT18] TO WR[0]          ; to poly and single
U 2830, C758,15 ;23991          BIS LS[BIT12] TO WR[0]
U 2831, 90F6,15 ;23992          MISC2 [TRAP.ACC] WR[0]
U 2832, 087A,BC ;23993          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2833, 364C,95 ;23994          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2834, BEA2,95 ;23995          MOV WR[1] TO LS[DATA.1]
U 2835, E5A4,15 ;23996          CLR LS[DATA.2]
U 2836, 65A6,15 ;23997          CLR LS[DATA.3]
U 2837, 887C,6C ;23998          JSR [CHECK.SUB]
U 2838, 0A81,A4 ;23999          JMP [T43.9]                     ;Loop on error
U 2839, FF82,15 ;24000          INC LS[ERROR.NUMBER]            ;Go onto error A
U 283A, E5A2,15 ;24001  T43.A:  CLR LS[DATA.1]                  ;Setup data for load
U 283B, E5A4,15 ;24002          CLR LS[DATA.2]
U 283C, 65A6,15 ;24003          CLR LS[DATA.3]
```

```
U 283D, 378C.15 ;24004          MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 283E, B716.95 ;24005          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 283F, 90F6.15 ;24006          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2840, 10F6.95 ;24007          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2841, B790.15 ;24008          MOV LS[TC8] TO WR[0]          ;Setup to shift left FT0 IN[ONE]
U 2842, 90F6.15 ;24009          MISC2 [TRAP.ACC] WR[0]
U 2843, 10F6.95 ;24010          MISC2 [TRAP.ACC] WR[1]
U 2844, B610.15 ;24011          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 2845, 90F6.15 ;24012          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 2846, 10F6.95 ;24013          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2847, 0878.FC ;24014          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2848, 0878.AC ;24015          JSR [MOV.DATA]                ;MOV FT0 TO FT4
U 2849, 3618.15 ;24016          MOV LS[T12] TO WR[0]          ;Setup to MOV FT0 TO FT2
U 284A, B716.95 ;24017          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 284B, 90F6.15 ;24018          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT2
U 284C, 10F6.95 ;24019          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 284D, 3716.15 ;24020          MOV LS[#300] TO WR[0]         ;Set instruction and size bits to
U 284E, C764.15 ;24021          BIS LS[BIT18] TO WR[0]        ; poly and huge
U 284F, C762.15 ;24022          BIS LS[BIT17] TO WR[0]
U 2850, 4760.15 ;24023          BIS LS[BIT16] TO WR[0]
U 2851, C758.15 ;24024          BIS LS[BIT12] TO WR[0]
U 2852, 90F6.15 ;24025          MISC2 [TRAP.ACC] WR[0]
U 2853, 0AD1.1C ;24026          JSR [MUL.ADD.FT2.FT4]         ;MUL ADD FT2 TO FT4
U 2854, 3716.15 ;24027          MOV LS[#300] TO WR[0]         ;Set instruction and size bits to
U 2855, C764.15 ;24028          BIS LS[BIT18] TO WR[0]        ; poly and single
U 2856, C758.15 ;24029          BIS LS[BIT12] TO WR[0]
U 2857, 90F6.15 ;24030          MISC2 [TRAP.ACC] WR[0]
U 2858, 3708.15 ;24031          MOV LS[T84] TO WR[0]          ;Setup to shift FT4 right
U 2859, 90F6.15 ;24032          MISC2 [TRAP.ACC] WR[0]        ;Shift right FT4
U 285A, 10F6.95 ;24033          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 285B, 087A.BC ;24034          JSR [STORE.X.TRIPLE]          ;Store FT4 to the CPU
U 285C, 364C.95 ;24035          MOV LS[BIT6] TO WR[1]         ;Setup expected data
U 285D, BEA2.95 ;24036          MOV WR[1] TO LS[DATA.1]
U 285E, E5A4.15 ;24037          CLR LS[DATA.2]
U 285F, 65A6.15 ;24038          CLR LS[DATA.3]
U 2860, 887C.6C ;24039          JSR [CHECK.SUB]
U 2861, 8A83.A4 ;24040          JMP [T43.A]                   ;Loop on error
U 2862, FF82.15 ;24041          INC LS[ERROR.NUMBER]          ;Go onto error B
U 2863, E5A2.15 ;24042  T43.B:  CLR LS[DATA.1]                ;Setup data for load
U 2864, E5A4.15 ;24043          CLR LS[DATA.2]
U 2865, 65A6.15 ;24044          CLR LS[DATA.3]
U 2866, 378C.15 ;24045          MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 2867, 3716.15 ;24046          MOV LS[#300] TO WR[0]         ;Setup to loop on self
U 2868, 90F6.15 ;24047          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2869, 10F6.95 ;24048          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 286A, B790.15 ;24049          MOV LS[TC8] TO WR[0]          ;Setup to shift left FT0
U 286B, 90F6.15 ;24050          MISC2 [TRAP.ACC] WR[0]
U 286C, 10F6.95 ;24051          MISC2 [TRAP.ACC] WR[1]
U 286D, B610.15 ;24052          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 286E, 90F6.15 ;24053          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 286F, 10F6.95 ;24054          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2870, 0878.FC ;24055          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2871, 3618.15 ;24056          MOV LS[T12] TO WR[0]          ;Setup to MOV FT0 TO FT2
U 2872, B716.95 ;24057          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 2873, 90F6.15 ;24058          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT2
```

```
U 2874, 10F6,95 ;24059          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2875, 378C,15 ;24060          MOV LS[TC5] TO WR[0]            ;Set up to CLR FT0
U 2876, 90F6,15 ;24061          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2877, 10F6,95 ;24062          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2878, 0878,AC ;24063          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2879, 3716,15 ;24064          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 287A, C764,15 ;24065          BIS LS[BIT18] TO WR[0]          ; poly and huge
U 287B, C762,15 ;24066          BIS LS[BIT17] TO WR[0]
U 287C, 4760,15 ;24067          BIS LS[BIT16] TO WR[0]
U 287D, C758,15 ;24068          BIS LS[BIT12] TO WR[0]
U 287E, 90F6,15 ;24069          MISC2 [TRAP.ACC] WR[0]
U 287F, 0AD1,1C ;24070          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2880, 3716,15 ;24071          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2881, C764,15 ;24072          BIS LS[BIT18] TO WR[0]          ; poly and single
U 2882, C758,15 ;24073          BIS LS[BIT12] TO WR[0]
U 2883, 90F6,15 ;24074          MISC2 [TRAP.ACC] WR[0]
U 2884, 087A,BC ;24075          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2885, 364C,95 ;24076          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2886, BEA2,95 ;24077          MOV WR[1] TO LS[DATA.1]
U 2887, E5A4,15 ;24078          CLR LS[DATA.2]
U 2888, 65A6,15 ;24079          CLR LS[DATA.3]
U 2889, 887C,6C ;24080          JSR [CHECK.SUB]
U 288A, 8A86,34 ;24081          JMP [T43.B]                     ;Loop on error
U 288B, FF82,15 ;24082          INC LS[ERROR.NUMBER]            ;Go on to error C
U 288C, E5A2,15 ;24083  T43.C:  CLR LS[DATA.1]                  ;Setup data for load
U 288D, E5A4,15 ;24084          CLR LS[DATA.2]
U 288E, 65A6,15 ;24085          CLR LS[DATA.3]
U 288F, 378C,15 ;24086          MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2890, B716,95 ;24087          MOV LS[#300] TO WR[1]           ;Loop on self
U 2891, 90F6,15 ;24088          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2892, 10F6,95 ;24089          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2893, B610,15 ;24090          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2894, B716,95 ;24091          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2895, 90F6,15 ;24092          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2896, 10F6,95 ;24093          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2897, 0878,FC ;24094          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2898, 0878,AC ;24095          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2899, 3618,15 ;24096          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 289A, B716,95 ;24097          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 289B, 90F6,15 ;24098          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 289C, 10F6,95 ;24099          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 289D, 3716,15 ;24100          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 289E, C764,15 ;24101          BIS LS[BIT18] TO WR[0]          ; poly and huge
U 289F, C762,15 ;24102          BIS LS[BIT17] TO WR[0]
U 28A0, 4760,15 ;24103          BIS LS[BIT16] TO WR[0]
U 28A1, C758,15 ;24104          BIS LS[BIT12] TO WR[0]
U 28A2, 90F6,15 ;24105          MISC2 [TRAP.ACC] WR[0]
U 28A3, 0AD1,1C ;24106          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 28A4, 3716,15 ;24107          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 28A5, C764,15 ;24108          BIS LS[BIT18] TO WR[0]          ; poly and single
U 28A6, C758,15 ;24109          BIS LS[BIT12] TO WR[0]
U 28A7, 90F6,15 ;24110          MISC2 [TRAP.ACC] WR[0]
U 28A8, 087A,BC ;24111          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 28A9, 364C,95 ;24112          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 28AA, BEA2,95 ;24113          MOV WR[1] TO LS[DATA.1]
```

```
U 28AB, E5A4,15 ;24114          CLR LS[DATA.2]
U 28AC, 65A6,15 ;24115          CLR LS[DATA.3]
U 28AD, 887C,6C ;24116          JSR [CHECK.SUB]
U 28AE, 0A88,C4 ;24117          JMP [T43.C]                    ;Loop on error
U 28AF, FF82,15 ;24118          INC LS[ERROR.NUMBER]           ;Go onto error D
U 28B0, 3716,15 ;24119          MOV LS[#300] TO WR[0]          ;Set instruction and size bits
U 28B1, C764,15 ;24120          BIS LS[BIT18] TO WR[0]
U 28B2, C758,15 ;24121          BIS LS[BIT12] TO WR[0]
U 28B3, 4756,15 ;24122          BIS LS[BIT11] TO WR[0]
U 28B4, 90F6,15 ;24123          MISC2 [TRAP.ACC] WR[0]
U 28B5, E5A2,15 ;24124  T43.D:  CLR LS[DATA.1]                 ;Setup data for load
U 28B6, B65E,15 ;24125          MOV LS[BIT15] TO WR[0]
U 28B7, 3EA4,15 ;24126          MOV WR[0] TO LS[DATA.2]
U 28B8, 65A6,15 ;24127          CLR LS[DATA.3]
U 28B9, 0878,FC ;24128          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 28BA, 3614,15 ;24129          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 28BB, 90F6,15 ;24130          MISC2 [TRAP.ACC] WR[0]
U 28BC, B610,15 ;24131          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 28BD, B716,95 ;24132          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 28BE, 90F6,15 ;24133          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 28BF, 10F6,95 ;24134          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 28C0, E5A4,15 ;24135          CLR LS[DATA.2]                 ;Set up data for load
U 28C1, 0878,FC ;24136          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 28C2, 0878,AC ;24137          JSR [MOV.DATA]                 ;MOV FT0 TO FT4
U 28C3, 3618,15 ;24138          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT4
U 28C4, B716,95 ;24139          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 28C5, 90F6,15 ;24140          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT4
U 28C6, 10F6,95 ;24141          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 28C7, 0AD1,1C ;24142          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 28C8, 3708,15 ;24143          MOV LS[T84] TO WR[0]           ;Setup to shift right FT4
U 28C9, 90F6,15 ;24144          MISC2 [TRAP.ACC] WR[0]         ;Shift right FT4
U 28CA, 10F6,95 ;24145          MISC2 [TRAP.ACC] WR[1]
U 28CB, 087A,BC ;24146          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 28CC, 364C,95 ;24147          MOV LS[BIT6] TO WR[1]          ;Setup expected data
U 28CD, BEA2,95 ;24148          MOV WR[1] TO LS[DATA.1]
U 28CE, E5A4,15 ;24149          CLR LS[DATA.2]
U 28CF, 65A6,15 ;24150          CLR LS[DATA.3]
U 28D0, 887C,6C ;24151          JSR [CHECK.SUB]
U 28D1, 0A8B,54 ;24152          JMP [T43.D]                    ;Loop on error
U 28D2, FF82,15 ;24153          INC LS[ERROR.NUMBER]           ;Go onto error E
U 28D3, E5A2,15 ;24154  T43.E:  CLR LS[DATA.1]                 ;Setup data for load
U 28D4, B65E,15 ;24155          MOV LS[BIT15] TO WR[0]
U 28D5, 3EA4,15 ;24156          MOV WR[0] TO LS[DATA.2]
U 28D6, 65A6,15 ;24157          CLR LS[DATA.3]
U 28D7, 0878,FC ;24158          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 28D8, 3614,15 ;24159          MOV LS[T10] TO WR[0]           ;Setup to shift left FT0
U 28D9, 90F6,15 ;24160          MISC2 [TRAP.ACC] WR[0]
U 28DA, B610,15 ;24161          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 28DB, B716,95 ;24162          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 28DC, 90F6,15 ;24163          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 28DD, 10F6,95 ;24164          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 28DE, E5A4,15 ;24165          CLR LS[DATA.2]                 ;Set up data for load
U 28DF, 0878,FC ;24166          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 28E0, 3618,15 ;24167          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 28E1, B716,95 ;24168          MOV LS[#300] TO WR[1]          ;Setup to loop on self
```

```
U 28E2, 90F6,15 ;24169          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 28E3, 10F6,95 ;24170          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 28E4, 378C,15 ;24171          MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 28E5, 90F6,15 ;24172          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 28E6, 10F6,95 ;24173          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 28E7, 0878,AC ;24174          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 28E8, 0AD1,1C ;24175          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 28E9, 087A,BC ;24176          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 28EA, 364C,95 ;24177          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 28EB, BEA2,95 ;24178          MOV WR[1] TO LS[DATA.1]
U 28EC, E5A4,15 ;24179          CLR LS[DATA.2]
U 28ED, 65A6,15 ;24180          CLR LS[DATA.3]
U 28EE, 887C,6C ;24181          JSR [CHECK.SUB]
U 28EF, 0A8D,34 ;24182          JMP [T43.E]                     ;Loop on error
U 28F0, FF82,15 ;24183          INC LS[ERROR.NUMBER]            ;Go on to error F
U 28F1, 087C,1C ;24184  T43.F:  JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 28F2, 3614,15 ;24185          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 28F3, 90F6,15 ;24186          MISC2 [TRAP.ACC] WR[0]
U 28F4, B610,15 ;24187          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 28F5, B716,95 ;24188          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 28F6, 90F6,15 ;24189          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 28F7, 10F6,95 ;24190          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 28F8, E5A4,15 ;24191          CLR LS[DATA.2]                  ;Set up data for load
U 28F9, 0878,FC ;24192          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 28FA, 0878,AC ;24193          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 28FB, 3618,15 ;24194          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 28FC, B716,95 ;24195          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 28FD, 90F6,15 ;24196          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 28FE, 10F6,95 ;24197          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 28FF, 0AD1,1C ;24198          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2900, 087A,BC ;24199          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2901, 364C,95 ;24200          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2902, BEA2,95 ;24201          MOV WR[1] TO LS[DATA.1]
U 2903, E5A4,15 ;24202          CLR LS[DATA.2]
U 2904, 65A6,15 ;24203          CLR LS[DATA.3]
U 2905, 887C,6C ;24204          JSR [CHECK.SUB]
U 2906, 0A8F,14 ;24205          JMP [T43.F]                     ;Loop on error
U 2907, FF82,15 ;24206          INC LS[ERROR.NUMBER]            ;Go onto error 10
U 2908, 3716,15 ;24207          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2909, C764,15 ;24208          BIS LS[BIT18] TO WR[0]          ; emod and single
U 290A, C758,15 ;24209          BIS LS[BIT12] TO WR[0]
U 290B, 4756,15 ;24210          BIS LS[BIT11] TO WR[0]
U 290C, C754,15 ;24211          BIS LS[BIT10] TO WR[0]
U 290D, 90F6,15 ;24212          MISC2 [TRAP.ACC] WR[0]
U 290E, E5A2,15 ;24213  T43.10: CLR LS[DATA.1]                  ;Setup data for load
U 290F, B65E,15 ;24214          MOV LS[BIT15] TO WR[0]
U 2910, 3EA4,15 ;24215          MOV WR[0] TO LS[DATA.2]
U 2911, 65A6,15 ;24216          CLR LS[DATA.3]
U 2912, 0878,FC ;24217          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2913, 3614,15 ;24218          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 2914, 90F6,15 ;24219          MISC2 [TRAP.ACC] WR[0]
U 2915, B610,15 ;24220          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2916, B716,95 ;24221          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2917, 90F6,15 ;24222          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2918, 10F6,95 ;24223          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
```

```
U 2919, E5A4,15 ;24224          CLR LS[DATA.2]                  ;Set up data for load
U 291A, 0878,FC ;24225          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 291B, 0878,AC ;24226          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 291C, 3618,15 ;24227          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 291D, B716,95 ;24228          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 291E, 90F6,15 ;24229          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 291F, 10F6,95 ;24230          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2920, 0AD1,1C ;24231          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2921, 3708,15 ;24232          MOV LS[T84] TO WR[0]            ;Setup to shift right FT4
U 2922, 90F6,15 ;24233          MISC2 [TRAP.ACC] WR[0]          ;Shift right FT4
U 2923, 10F6,95 ;24234          MISC2 [TRAP.ACC] WR[1]
U 2924, 087A,BC ;24235          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2925, 364C,95 ;24236          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2926, BEA2,95 ;24237          MOV WR[1] TO LS[DATA.1]
U 2927, E5A4,15 ;24238          CLR LS[DATA.2]
U 2928, 65A6,15 ;24239          CLR LS[DATA.3]
U 2929, 887C,6C ;24240          JSR [CHECK.SUB]
U 292A, 8A90,E4 ;24241          JMP [T43.10]                    ;Loop on error
U 292B, FF82,15 ;24242          INC LS[ERROR.NUMBER]            ;Go onto error 11
U 292C, E5A2,15 ;24243  T43.11: CLR LS[DATA.1]                  ;Setup data for load
U 292D, B65E,15 ;24244          MOV LS[BIT15] TO WR[0]
U 292E, 3EA4,15 ;24245          MOV WR[0] TO LS[DATA.2]
U 292F, 65A6,15 ;24246          CLR LS[DATA.3]
U 2930, 0878,FC ;24247          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2931, 3614,15 ;24248          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 2932, 90F6,15 ;24249          MISC2 [TRAP.ACC] WR[0]
U 2933, B610,15 ;24250          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2934, B716,95 ;24251          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2935, 90F6,15 ;24252          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2936, 10F6,95 ;24253          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2937, E5A4,15 ;24254          CLR LS[DATA.2]                  ;Set up data for load
U 2938, 0878,FC ;24255          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2939, 3618,15 ;24256          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 293A, B716,95 ;24257          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 293B, 90F6,15 ;24258          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 293C, 10F6,95 ;24259          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 293D, 378C,15 ;24260          MOV LS[TC5] TO WR[0]            ;Setup CLR FT0
U 293E, 90F6,15 ;24261          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 293F, 10F6,95 ;24262          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2940, 0878,AC ;24263          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2941, 0AD1,1C ;24264          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2942, 087A,BC ;24265          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2943, 364C,95 ;24266          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 2944, BEA2,95 ;24267          MOV WR[1] TO LS[DATA.1]
U 2945, E5A4,15 ;24268          CLR LS[DATA.2]
U 2946, 65A6,15 ;24269          CLR LS[DATA.3]
U 2947, 887C,6C ;24270          JSR [CHECK.SUB]
U 2948, 8A92,C4 ;24271          JMP [T43.11]                    ;Loop on error
U 2949, FF82,15 ;24272          INC LS[ERROR.NUMBER]            ;Go on to error 12
U 294A, 087C,1C ;24273  T43.12: JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 294B, 3614,15 ;24274          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 294C, 90F6,15 ;24275          MISC2 [TRAP.ACC] WR[0]
U 294D, B610,15 ;24276          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 294E, B716,95 ;24277          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 294F, 90F6,15 ;24278          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
```

```
U 2950, 10F6,95 ;24279          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2951, E5A4,15 ;24280          CLR LS[DATA.2]                  ;Set up data for load
U 2952, 0878,FC ;24281          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2953, 0878,AC ;24282          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2954, 3618,15 ;24283          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 2955, B716,95 ;24284          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2956, 90F6,15 ;24285          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 2957, 10F6,95 ;24286          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2958, 0AD1,1C ;24287          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2959, 087A,BC ;24288          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 295A, 364C,95 ;24289          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 295B, BEA2,95 ;24290          MOV WR[1] TO LS[DATA.1]
U 295C, E5A4,15 ;24291          CLR LS[DATA.2]
U 295D, 65A6,15 ;24292          CLR LS[DATA.3]
U 295E, 887C,6C ;24293          JSR [CHECK.SUB]
U 295F, 8A94,A4 ;24294          JMP [T43.12]                    ;Loop on error
U 2960, FF82,15 ;24295          INC LS[ERROR.NUMBER]            ;Go onto error 13
U 2961, 3716,15 ;24296          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2962, C764,15 ;24297          BIS LS[BIT18] TO WR[0]          ; add and single
U 2963, 90F6,15 ;24298          MISC2 [TRAP.ACC] WR[0]
U 2964, E5A2,15 ;24299  T43.13: CLR LS[DATA.1]                  ;Setup data for load
U 2965, B65E,15 ;24300          MOV LS[BIT15] TO WR[0]
U 2966, 3EA4,15 ;24301          MOV WR[0] TO LS[DATA.2]
U 2967, 65A6,15 ;24302          CLR LS[DATA.3]
U 2968, 0878,FC ;24303          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2969, 3614,15 ;24304          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 296A, 90F6,15 ;24305          MISC2 [TRAP.ACC] WR[0]
U 296B, B610,15 ;24306          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 296C, B716,95 ;24307          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 296D, 90F6,15 ;24308          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 296E, 10F6,95 ;24309          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 296F, E5A4,15 ;24310          CLR LS[DATA.2]                  ;Set up data for load
U 2970, 0878,FC ;24311          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2971, 0878,AC ;24312          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2972, 3618,15 ;24313          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 2973, B716,95 ;24314          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2974, 90F6,15 ;24315          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 2975, 10F6,95 ;24316          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2976, 0AD1,1C ;24317          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2977, 3708,15 ;24318          MOV LS[T84] TO WR[0]            ;Setup to shift right FT4
U 2978, 90F6,15 ;24319          MISC2 [TRAP.ACC] WR[0]          ;Shift right FT4
U 2979, 10F6,95 ;24320          MISC2 [TRAP.ACC] WR[1]
U 297A, 087A,BC ;24321          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 297B, 364C,95 ;24322          MOV LS[BIT6] TO WR[1]
U 297C, BEA2,95 ;24323          MOV WR[1] TO LS[DATA.1]
U 297D, E5A4,15 ;24324          CLR LS[DATA.2]
U 297E, 65A6,15 ;24325          CLR LS[DATA.3]
U 297F, 887C,6C ;24326          JSR [CHECK.SUB]
U 2980, 8A96,44 ;24327          JMP [T43.13]                    ;Loop on error
U 2981, FF82,15 ;24328          INC LS[ERROR.NUMBER]            ;Go onto error 14
U 2982, E5A2,15 ;24329  T43.14: CLR LS[DATA.1]                  ;Setup data for load
U 2983, B65E,15 ;24330          MOV LS[BIT15] TO WR[0]
U 2984, 3EA4,15 ;24331          MOV WR[0] TO LS[DATA.2]
U 2985, 65A6,15 ;24332          CLR LS[DATA.3]
U 2986, 0878,FC ;24333          JSR [LOAD.TRIPLE]               ;Load data into FT0
```

```
U 2987, 3614,15 ;24334          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 2988, 90F6,15 ;24335          MISC2 [TRAP.ACC] WR[0]
U 2989, B610,15 ;24336          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 298A, B716,95 ;24337          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 298B, 90F6,15 ;24338          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 298C, 10F6,95 ;24339          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 298D, E5A4,15 ;24340          CLR LS[DATA.2]                  ;Set up data for load
U 298E, 0878,FC ;24341          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 298F, 3618,15 ;24342          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2990, B716,95 ;24343          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2991, 90F6,15 ;24344          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2992, 10F6,95 ;24345          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2993, 378C,15 ;24346          MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2994, 90F6,15 ;24347          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2995, 10F6,95 ;24348          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2996, 0878,AC ;24349          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2997, 0AD1,1C ;24350          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2998, 087A,BC ;24351          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2999, 364C,95 ;24352          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 299A, BEA2,95 ;24353          MOV WR[1] TO LS[DATA.1]
U 299B, E5A4,15 ;24354          CLR LS[DATA.2]
U 299C, 65A6,15 ;24355          CLR LS[DATA.3]
U 299D, 887C,6C ;24356          JSR [CHECK.SUB]
U 299E, 0A98,24 ;24357          JMP [T43.14]                    ;Loop on error
U 299F, FF82,15 ;24358          INC LS[ERROR.NUMBER]            ;Go on to error 15
U 29A0, 087C,1C ;24359  T43.15: JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 29A1, 3614,15 ;24360          MOV LS[T10] TO WR[0]            ;Setup to shift left FT0
U 29A2, 90F6,15 ;24361          MISC2 [TRAP.ACC] WR[0]
U 29A3, B610,15 ;24362          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 29A4, B716,95 ;24363          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 29A5, 90F6,15 ;24364          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 29A6, 10F6,95 ;24365          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 29A7, E5A4,15 ;24366          CLR LS[DATA.2]                  ;Set up data for load
U 29A8, 0878,FC ;24367          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 29A9, 0878,AC ;24368          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 29AA, 3618,15 ;24369          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT4
U 29AB, B716,95 ;24370          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 29AC, 90F6,15 ;24371          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT4
U 29AD, 10F6,95 ;24372          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 29AE, 0AD1,1C ;24373          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 29AF, 3708,15 ;24374          MOV LS[T84] TO WR[0]            ;Setup to shift right FT4
U 29B0, 90F6,15 ;24375          MISC2 [TRAP.ACC] WR[0]          ;Shift right FT4
U 29B1, 10F6,95 ;24376          MISC2 [TRAP.ACC] WR[1]
U 29B2, 087A,BC ;24377          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 29B3, 364C,95 ;24378          MOV LS[BIT6] TO WR[1]           ;Setup expected data
U 29B4, BEA2,95 ;24379          MOV WR[1] TO LS[DATA.1]
U 29B5, E5A4,15 ;24380          CLR LS[DATA.2]
U 29B6, 65A6,15 ;24381          CLR LS[DATA.3]
U 29B7, 887C,6C ;24382          JSR [CHECK.SUB]
U 29B8, 0A9A,04 ;24383          JMP [T43.15]                    ;Loop on error
U 29B9, FF82,15 ;24384          INC LS[ERROR.NUMBER]            ;Go onto error 16
U 29BA, 3716,15 ;24385          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 29BB, C764,15 ;24386          BIS LS[BIT18] TO WR[0]          ; mull and single
U 29BC, 90F6,15 ;24387          MISC2 [TRAP.ACC] WR[0]
U 29BD, E5A2,15 ;24388  T43.16: CLR LS[DATA.1]                  ;Setup data
```

```
U 29BE, B640,15 ;24389          MOV LS[BIT0] TO WR[0]
U 29BF, 3EA4,15 ;24390          MOV WR[0] TO LS[DATA.2]
U 29C0, 65A6,15 ;24391          CLR LS[DATA.3]
U 29C1, 0878,FC ;24392          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 29C2, B610,15 ;24393          MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 29C3, B716,95 ;24394          MOV LS[#300] TO WR[1]       ;Setup to loop on self
U 29C4, 90F6,15 ;24395          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 29C5, 10F6,95 ;24396          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 29C6, 367E,15 ;24397          MOV LS[BIT31] TO WR[0]      ;Setup data
U 29C7, 3EA2,15 ;24398          MOV WR[0] TO LS[DATA.1]
U 29C8, E5A4,15 ;24399          CLR LS[DATA.2]
U 29C9, 65A6,15 ;24400          CLR LS[DATA.3]
U 29CA, 0878,FC ;24401          JSR [LOAD.TRIPLE]           ;Load the data into FT0
U 29CB, 0878,AC ;24402          JSR [MOV.DATA]              ;MOV FT0 TO FT4
U 29CC, 3618,15 ;24403          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 29CD, 90F6,15 ;24404          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 29CE, 10F6,95 ;24405          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 29CF, 3716,15 ;24406          MOV LS[#300] TO WR[0]       ;Set instruction and size bits to
U 29D0, C764,15 ;24407          BIS LS[BIT18] TO WR[0]      ; mull and single
U 29D1, 475C,15 ;24408          BIS LS[BIT14] TO WR[0]
U 29D2, 475A,15 ;24409          BIS LS[BIT13] TO WR[0]
U 29D3, 90F6,15 ;24410          MISC2 [TRAP.ACC] WR[0]
U 29D4, 0AD1,1C ;24411          JSR [MUL.ADD.FT2.FT4]       ;MUL ADD FT2 TO FT4
U 29D5, 3716,15 ;24412          MOV LS[#300] TO WR[0]       ;Set instruction and size bits
U 29D6, C764,15 ;24413          BIS LS[BIT18] TO WR[0]      ; to add and single
U 29D7, 90F6,15 ;24414          MISC2 [TRAP.ACC] WR[0]
U 29D8, 087A,BC ;24415          JSR [STORE.X.TRIPLE]        ;Store FT4 to the CPU
U 29D9, B67E,95 ;24416          MOV LS[BIT31] TO WR[1]      ;Setup expected data
U 29DA, BEA2,95 ;24417          MOV WR[1] TO LS[DATA.1]
U 29DB, E5A4,15 ;24418          CLR LS[DATA.2]
U 29DC, 65A6,15 ;24419          CLR LS[DATA.3]
U 29DD, 887C,6C ;24420          JSR [CHECK.SUB]
U 29DE, 0A9B,D4 ;24421          JMP [T43.16]                ;Loop on error
U 29DF, FF82,15 ;24422          INC LS[ERROR.NUMBER]        ;Go onto error 17
U 29E0, E5A2,15 ;24423  T43.17: CLR LS[DATA.1]              ;Setup data
U 29E1, B640,15 ;24424          MOV LS[BIT0] TO WR[0]
U 29E2, 3EA4,15 ;24425          MOV WR[0] TO LS[DATA.2]
U 29E3, 65A6,15 ;24426          CLR LS[DATA.3]
U 29E4, 0878,FC ;24427          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 29E5, B610,15 ;24428          MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 29E6, B716,95 ;24429          MOV LS[#300] TO WR[1]       ;Setup to loop on self
U 29E7, 90F6,15 ;24430          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 29E8, 10F6,95 ;24431          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 29E9, 367E,15 ;24432          MOV LS[BIT31] TO WR[0]      ;Setup data
U 29EA, C77C,15 ;24433          BIS LS[BIT30] TO WR[0]
U 29EB, 3EA2,15 ;24434          MOV WR[0] TO LS[DATA.1]
U 29EC, E5A4,15 ;24435          CLR LS[DATA.2]
U 29ED, 65A6,15 ;24436          CLR LS[DATA.3]
U 29EE, 0878,FC ;24437          JSR [LOAD.TRIPLE]           ;Load the data into FT0
U 29EF, 0878,AC ;24438          JSR [MOV.DATA]              ;MOV FT0 TO FT4
U 29F0, 3618,15 ;24439          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 29F1, 90F6,15 ;24440          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 29F2, 10F6,95 ;24441          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 29F3, 3716,15 ;24442          MOV LS[#300] TO WR[0]       ;Set instruction and size bits to
U 29F4, C764,15 ;24443          BIS LS[BIT18] TO WR[0]      ; mull and single
```

```
U 29F5, 475C,15 ;24444          BIS LS[BIT14] TO WR[0]
U 29F6, 475A,15 ;24445          BIS LS[BIT13] TO WR[0]
U 29F7, 90F6,15 ;24446          MISC2 [TRAP.ACC] WR[0]
U 29F8, 0AD1,1C ;24447          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 29F9, 3716,15 ;24448          MOV LS[#300] TO WR[0]          ;Set instruction and size bits
U 29FA, C764,15 ;24449          BIS LS[BIT18] TO WR[0]         ; to add and single
U 29FB, 90F6,15 ;24450          MISC2 [TRAP.ACC] WR[0]
U 29FC, 087A,BC ;24451          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 29FD, B67E,95 ;24452          MOV LS[BIT31] TO WR[1]         ;Setup expected data
U 29FE, 477C,95 ;24453          BIS LS[BIT30] TO WR[1]
U 29FF, BEA2,95 ;24454          MOV WR[1] TO LS[DATA.1]
U 2A00, E5A4,15 ;24455          CLR LS[DATA.2]
U 2A01, 65A6,15 ;24456          CLR LS[DATA.3]
U 2A02, 887C,6C ;24457          JSR [CHECK.SUB]
U 2A03, 8A9E,04 ;24458          JMP [T43.17]                   ;Loop on error
U 2A04, FF82,15 ;24459          INC LS[ERROR.NUMBER]           ;Go onto error 18
U 2A05, E5A2,15 ;24460  T43.18: CLR LS[DATA.1]                 ;Setup data
U 2A06, B640,15 ;24461          MOV LS[BIT0] TO WR[0]
U 2A07, 3EA4,15 ;24462          MOV WR[0] TO LS[DATA.2]
U 2A08, 65A6,15 ;24463          CLR LS[DATA.3]
U 2A09, 0878,FC ;24464          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2A0A, B610,15 ;24465          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 2A0B, B716,95 ;24466          MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 2A0C, 90F6,15 ;24467          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2A0D, 10F6,95 ;24468          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A0E, B67C,15 ;24469          MOV LS[BIT30] TO WR[0]         ;Setup data
U 2A0F, 3EA2,15 ;24470          MOV WR[0] TO LS[DATA.1]
U 2A10, E5A4,15 ;24471          CLR LS[DATA.2]
U 2A11, 65A6,15 ;24472          CLR LS[DATA.3]
U 2A12, 0878,FC ;24473          JSR [LOAD.TRIPLE]              ;Load the data into FT0
U 2A13, 0878,AC ;24474          JSR [MOV.DATA]                 ;MOV FT0 TO FT4
U 2A14, 3618,15 ;24475          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 2A15, 90F6,15 ;24476          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 2A16, 10F6,95 ;24477          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A17, 3716,15 ;24478          MOV LS[#300] TO WR[0]          ;Set instruction and size bits to
U 2A18, C764,15 ;24479          BIS LS[BIT18] TO WR[0]         ; mull and single
U 2A19, 475C,15 ;24480          BIS LS[BIT14] TO WR[0]
U 2A1A, 475A,15 ;24481          BIS LS[BIT13] TO WR[0]
U 2A1B, 90F6,15 ;24482          MISC2 [TRAP.ACC] WR[0]
U 2A1C, 0AD1,1C ;24483          JSR [MUL.ADD.FT2.FT4]          ;MUL ADD FT2 TO FT4
U 2A1D, 3716,15 ;24484          MOV LS[#300] TO WR[0]          ;Set instruction and size bits
U 2A1E, C764,15 ;24485          BIS LS[BIT18] TO WR[0]         ; to add and single
U 2A1F, 90F6,15 ;24486          MISC2 [TRAP.ACC] WR[0]
U 2A20, 087A,BC ;24487          JSR [STORE.X.TRIPLE]           ;Store FT4 to the CPU
U 2A21, 367C,95 ;24488          MOV LS[BIT30] TO WR[1]         ;Setup expected data
U 2A22, BEA2,95 ;24489          MOV WR[1] TO LS[DATA.1]
U 2A23, E5A4,15 ;24490          CLR LS[DATA.2]
U 2A24, 65A6,15 ;24491          CLR LS[DATA.3]
U 2A25, 887C,6C ;24492          JSR [CHECK.SUB]
U 2A26, 0AA0,54 ;24493          JMP [T43.18]                   ;Loop on error
U 2A27, FF82,15 ;24494          INC LS[ERROR.NUMBER]           ;Go onto error 19
U 2A28, E5A2,15 ;24495  T43.19: CLR LS[DATA.1]                 ;Setup data
U 2A29, B640,15 ;24496          MOV LS[BIT0] TO WR[0]
U 2A2A, 3EA4,15 ;24497          MOV WR[0] TO LS[DATA.2]
U 2A2B, 65A6,15 ;24498          CLR LS[DATA.3]
```

```
U 2A2C, 0878,FC ;24499          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2A2D, B610,15 ;24500          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2A2E, B716,95 ;24501          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2A2F, 90F6,15 ;24502          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2A30, 10F6,95 ;24503          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2A31, B67C,15 ;24504          MOV LS[BIT30] TO WR[0]          ;Setup data
U 2A32, 3EA2,15 ;24505          MOV WR[0] TO LS[DATA.1]
U 2A33, E5A4,15 ;24506          CLR LS[DATA.2]
U 2A34, 65A6,15 ;24507          CLR LS[DATA.3]
U 2A35, 0878,FC ;24508          JSR [LOAD.TRIPLE]               ;Load the data into FT0
U 2A36, 0878,AC ;24509          JSR [MOV.DATA]                  ;MOV FT0 TO FT4
U 2A37, B67C,15 ;24510          MOV LS[BIT30] TO WR[0]          ;Setup data for FT2
U 2A38, 477E,15 ;24511          BIS LS[BIT31] TO WR[0]
U 2A39, 3EA2,15 ;24512          MOV WR[0] TO LS[DATA.1]
U 2A3A, 0878,FC ;24513          JSR [LOAD.TRIPLE]               ;Load the data into FT0
U 2A3B, 3618,15 ;24514          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2A3C, 90F6,15 ;24515          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2A3D, 10F6,95 ;24516          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2A3E, 3716,15 ;24517          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2A3F, C764,15 ;24518          BIS LS[BIT18] TO WR[0]          ; mull and single
U 2A40, 475C,15 ;24519          BIS LS[BIT14] TO WR[0]
U 2A41, 475A,15 ;24520          BIS LS[BIT13] TO WR[0]
U 2A42, 90F6,15 ;24521          MISC2 [TRAP.ACC] WR[0]
U 2A43, 0AD1,1C ;24522          JSR [MUL.ADD.FT2.FT4]           ;MUL ADD FT2 TO FT4
U 2A44, 3716,15 ;24523          MOV LS[#300] TO WR[0]           ;Set instruction and size bits
U 2A45, C764,15 ;24524          BIS LS[BIT18] TO WR[0]          ; to add and single
U 2A46, 90F6,15 ;24525          MISC2 [TRAP.ACC] WR[0]
U 2A47, 087A,BC ;24526          JSR [STORE.X.TRIPLE]            ;Store FT4 to the CPU
U 2A48, E5A2,15 ;24527          CLR LS[DATA.1]
U 2A49, E5A4,15 ;24528          CLR LS[DATA.2]
U 2A4A, 65A6,15 ;24529          CLR LS[DATA.3]
U 2A4B, 887C,6C ;24530          JSR [CHECK.SUB]
U 2A4C, 0AA2,84 ;24531          JMP [T43.19]                    ;Loop on error
U 2A4D, FF82,15 ;24532          INC LS[ERROR.NUMBER]            ;Go onto error 1A
U 2A4E, 3650,15 ;24533          MOV LS[#100] TO WR[0]           ;Setup memory reference word for
U 2A4F, C74C,15 ;24534          BIS LS[#40] TO WR[0]            ; DIV section
U 2A50, C74A,15 ;24535          BIS LS[#20] TO WR[0]
U 2A51, 4748,15 ;24536          BIS LS[#10] TO WR[0]
U 2A52, BE1E,15 ;24537          MOV WR[0] TO LS[T15]
U 2A53, 3716,15 ;24538          MOV LS[#300] TO WR[0]           ;Setup to MOV FQ TO FT2 in T59
U 2A54, C74C,15 ;24539          BIS LS[#40] TO WR[0]
U 2A55, 4744,15 ;24540          BIS LS[#4] TO WR[0]
U 2A56, 4742,15 ;24541          BIS LS[#2] TO WR[0]
U 2A57, C740,15 ;24542          BIS LS[#1] TO WR[0]
U 2A58, BEB2,15 ;24543          MOV WR[0] TO LS[T59]
U 2A59, B652,15 ;24544          MOV LS[#200] TO WR[0]           ;Setup SHF RIGHT FT0 AND FQ IN[ZERO]
U 2A5A, 474E,15 ;24545          BIS LS[#80] TO WR[0]            ; in T58
U 2A5B, C74C,15 ;24546          BIS LS[#40] TO WR[0]
U 2A5C, 4748,15 ;24547          BIS LS[#10] TO WR[0]
U 2A5D, 3EB0,15 ;24548          MOV WR[0] TO LS[T58]
U 2A5E, 36C0,15 ;24549          MOV LS[#3(H)] TO WR[0]          ;Setup INTEGER DIVIDE STEPS
U 2A5F, C752,15 ;24550          BIS LS[#200] TO WR[0]
U 2A60, 474E,15 ;24551          BIS LS[#80] TO WR[0]
U 2A61, C74A,15 ;24552          BIS LS[#20] TO WR[0]
U 2A62, 3EAE,15 ;24553          MOV WR[0] TO LS[T57]
```

```
U 2A63, 3716,15 ;24554           MOV LS[#300] TO WR[0]          ;Set instruction and size bits to
U 2A64, C764,15 ;24555           BIS LS[BIT18] TO WR[0]         ; DIV and single
U 2A65, C758,15 ;24556           BIS LS[BIT12] TO WR[0]
U 2A66, C754,15 ;24557           BIS LS[BIT10] TO WR[0]
U 2A67, 90F6,15 ;24558           MISC2 [TRAP.ACC] WR[0]
U 2A68, 378C,15 ;24559  T43.1A:  MOV LS[TC5] TO WR[0]           ;Setup to CLR FT0
U 2A69, B716,95 ;24560           MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 2A6A, 90F6,15 ;24561           MISC2 [TRAP.ACC] WR[0]         ;CLR FT0
U 2A6B, 10F6,95 ;24562           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A6C, B610,15 ;24563           MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 2A6D, 90F6,15 ;24564           MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2A6E, 10F6,95 ;24565           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A6F, 087C,1C ;24566           JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 2A70, 3618,15 ;24567           MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 2A71, B716,95 ;24568           MOV LS[#300] TO WR[1]          ;Setup to Loop on self
U 2A72, 90F6,15 ;24569           MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 2A73, 10F6,95 ;24570           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A74, B64C,15 ;24571           MOV LS[BIT6] TO WR[0]          ;Setup data
U 2A75, 3EA2,15 ;24572           MOV WR[0] TO LS[DATA.1]
U 2A76, 0878,FC ;24573           JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2A77, 8AD1,6C ;24574           JSR [DIVH]                     ;DO DIVIDE (T15 INSTRUCTION)
U 2A78, 36B2,15 ;24575           MOV LS[T59] TO WR[0]           ;Setup to MOV FQ TO FT2
U 2A79, 90F6,15 ;24576           MISC2 [TRAP.ACC] WR[0]         ;MOV FQ TO FT2
U 2A7A, 10F6,95 ;24577           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A7B, 087A,1C ;24578           JSR [STORE.2.TRIPLE]           ;Store FT2 to the CPU
U 2A7C, B660,95 ;24579           MOV LS[BIT16] TO WR[1]         ;Setup expected data
U 2A7D, BEA2,95 ;24580           MOV WR[1] TO LS[DATA.1]
U 2A7E, E5A4,15 ;24581           CLR LS[DATA.2]
U 2A7F, 65A6,15 ;24582           CLR LS[DATA.3]
U 2A80, 887C,6C ;24583           JSR [CHECK.SUB]
U 2A81, 8AA6,84 ;24584           JMP [T43.1A]                   ;Loop on error
U 2A82, FF82,15 ;24585           INC LS[ERROR.NUMBER]           ;Go onto error 1B
U 2A83, 378C,15 ;24586  T43.1B:  MOV LS[TC5] TO WR[0]           ;Setup to CLR FT0
U 2A84, B716,95 ;24587           MOV LS[#300] TO WR[1]          ;Setup to loop on self
U 2A85, 90F6,15 ;24588           MISC2 [TRAP.ACC] WR[0]         ;CLR FT0
U 2A86, 10F6,95 ;24589           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A87, B610,15 ;24590           MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 2A88, 90F6,15 ;24591           MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2A89, 10F6,95 ;24592           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A8A, B64C,15 ;24593           MOV LS[BIT6] TO WR[0]          ;Setup data
U 2A8B, 3EA2,15 ;24594           MOV WR[0] TO LS[DATA.1]
U 2A8C, E5A4,15 ;24595           CLR LS[DATA.2]
U 2A8D, 65A6,15 ;24596           CLR LS[DATA.3]
U 2A8E, 0878,FC ;24597           JSR [LOAD.TRIPLE]              ;Load FT0 with zeros and hidden bit
U 2A8F, 3618,15 ;24598           MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 2A90, B716,95 ;24599           MOV LS[#300] TO WR[1]          ;Setup to Loop on self
U 2A91, 90F6,15 ;24600           MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 2A92, 10F6,95 ;24601           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A93, E5A2,15 ;24602           CLR LS[DATA.1]
U 2A94, 0878,FC ;24603           JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2A95, 8AD1,6C ;24604           JSR [DIVH]                     ;DO DIVIDE (T15 INSTRUCTION)
U 2A96, 36B2,15 ;24605           MOV LS[T59] TO WR[0]           ;Setup to MOV FQ TO FT2
U 2A97, 90F6,15 ;24606           MISC2 [TRAP.ACC] WR[0]         ;MOV FQ TO FT2
U 2A98, 10F6,95 ;24607           MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2A99, 087A,1C ;24608           JSR [STORE.2.TRIPLE]           ;Store FT2 to the CPU
```

```
U 2A9A, B660,95 ;24609          MOV LS[BIT16] TO WR[1]          ;Setup expected data
U 2A9B, 4762,95 ;24610          BIS LS[BIT17] TO WR[1]
U 2A9C, BEA2,95 ;24611          MOV WR[1] TO LS[DATA.1]
U 2A9D, E5A4,15 ;24612          CLR LS[DATA.2]
U 2A9E, 65A6,15 ;24613          CLR LS[DATA.3]
U 2A9F, 887C,6C ;24614          JSR [CHECK.SUB]
U 2AA0, 8AA8,34 ;24615          JMP [T43.1B]                    ;Loop on error
U 2AA1, FF82,15 ;24616          INC LS[ERROR.NUMBER]            ;Go onto error 1C
U 2AA2, 378C,15 ;24617  T43.1C: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2AA3, B716,95 ;24618          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2AA4, 90F6,15 ;24619          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2AA5, 10F6,95 ;24620          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AA6, B610,15 ;24621          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2AA7, 90F6,15 ;24622          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2AA8, 10F6,95 ;24623          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AA9, 087C,1C ;24624          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 2AAA, 3618,15 ;24625          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2AAB, B716,95 ;24626          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2AAC, 90F6,15 ;24627          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2AAD, 10F6,95 ;24628          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AAE, 8AD1,6C ;24629          JSR [DIVH]                      ;DO DIVIDE (T15 INSTRUCTION)
U 2AAF, 36B2,15 ;24630          MOV LS[T59] TO WR[0]            ;Setup to MOV FQ TO FT2
U 2AB0, 90F6,15 ;24631          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT2
U 2AB1, 10F6,95 ;24632          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AB2, 087A,1C ;24633          JSR [STORE.2.TRIPLE]            ;Store FT2 to the CPU
U 2AB3, 3662,95 ;24634          MOV  LS[BIT17] TO WR[1]         ;Setup expected data
U 2AB4, BEA2,95 ;24635          MOV WR[1] TO LS[DATA.1]
U 2AB5, E5A4,15 ;24636          CLR LS[DATA.2]
U 2AB6, 65A6,15 ;24637          CLR LS[DATA.3]
U 2AB7, 887C,6C ;24638          JSR [CHECK.SUB]
U 2AB8, 8AAA,24 ;24639          JMP [T43.1C]                    ;Loop on error
U 2AB9, FF82,15 ;24640          INC LS[ERROR.NUMBER]            ;Go onto error 1D
U 2ABA, 378C,15 ;24641  T43.1D: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2ABB, B716,95 ;24642          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2ABC, 90F6,15 ;24643          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2ABD, 10F6,95 ;24644          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2ABE, B610,15 ;24645          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2ABF, 90F6,15 ;24646          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2AC0, 10F6,95 ;24647          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AC1, B64C,15 ;24648          MOV LS[BIT6] TO WR[0]           ;Setup data
U 2AC2, 3EA2,15 ;24649          MOV WR[0] TO LS[DATA.1]
U 2AC3, E5A4,15 ;24650          CLR LS[DATA.2]
U 2AC4, 65A6,15 ;24651          CLR LS[DATA.3]
U 2AC5, 0878,FC ;24652          JSR [LOAD.TRIPLE]               ;Load FT0 with zeros and hidden bit
U 2AC6, 3618,15 ;24653          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2AC7, B716,95 ;24654          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2AC8, 90F6,15 ;24655          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2AC9, 10F6,95 ;24656          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2ACA, E5A2,15 ;24657          CLR LS[DATA.1]                  ;Setup data
U 2ACB, 0878,FC ;24658          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2ACC, B6B0,15 ;24659          MOV LS[T58] TO WR[0]            ;Setup to SHIFT RIGHT FT0& FQ IN[ZERO]
U 2ACD, B716,95 ;24660          MOV LS[#300] TO WR[1]           ;Loop on self
U 2ACE, 90F6,15 ;24661          MISC2 [TRAP.ACC] WR[0]          ;SHIFT RIGHT FT0 & FQ IN[ZERO]
U 2ACF, 10F6,95 ;24662          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2AD0, 8AD1,6C ;24663          JSR [DIVH]                      ;DO DIVIDE (T15 INSTRUCTION)
```

```
U 2AD1, 36B2,15 ;24664          MOV LS[T59] TO WR[0]          ;Setup to MOV FQ TO FT2
U 2AD2, 90F6,15 ;24665          MISC2 [TRAP.ACC] WR[0]        ;MOV FQ TO FT2
U 2AD3, 10F6,95 ;24666          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2AD4, 087A,1C ;24667          JSR [STORE.2.TRIPLE]          ;Store FT2 to the CPU
U 2AD5, B660,95 ;24668          MOV LS[BIT16] TO WR[1]        ;Setup expected data
U 2AD6, 4762,95 ;24669          BIS LS[BIT17] TO WR[1]
U 2AD7, BEA2,95 ;24670          MOV WR[1] TO LS[DATA.1]
U 2AD8, E5A4,15 ;24671          CLR LS[DATA.2]
U 2AD9, 65A6,15 ;24672          CLR LS[DATA.3]
U 2ADA, 887C,6C ;24673          JSR [CHECK.SUB]
U 2ADB, 8AAB,A4 ;24674          JMP [T43.1D]                  ;Loop on error
U 2ADC, FF82,15 ;24675          INC LS[ERROR.NUMBER]          ;Go onto error 1E
U 2ADD, 3716,15 ;24676          MOV LS[#300] TO WR[0]         ;Set instruction and size bits to
U 2ADE, C764,15 ;24677          BIS LS[BIT18] TO WR[0]        ; DIV and double
U 2ADF, C758,15 ;24678          BIS LS[BIT12] TO WR[0]
U 2AE0, C754,15 ;24679          BIS LS[BIT10] TO WR[0]
U 2AE1, 4760,15 ;24680          BIS LS[BIT16] TO WR[0]
U 2AE2, 90F6,15 ;24681          MISC2 [TRAP.ACC] WR[0]
U 2AE3, 378C,15 ;24682  T43.1E: MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 2AE4, B716,95 ;24683          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 2AE5, 90F6,15 ;24684          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2AE6, 10F6,95 ;24685          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2AE7, B610,15 ;24686          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 2AE8, 90F6,15 ;24687          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 2AE9, 10F6,95 ;24688          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2AEA, 087C,1C ;24689          JSR [CLR.FT0]                 ;LOAD ZEROS INTO FT0
U 2AEB, 3618,15 ;24690          MOV LS[T12] TO WR[0]          ;Setup to MOV FT0 TO FT2
U 2AEC, B716,95 ;24691          MOV LS[#300] TO WR[1]         ;Setup to Loop on self
U 2AED, 90F6,15 ;24692          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT2
U 2AEE, 10F6,95 ;24693          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2AEF, B64C,15 ;24694          MOV LS[BIT6] TO WR[0]         ;Setup data
U 2AF0, 3EA2,15 ;24695          MOV WR[0] TO LS[DATA.1]
U 2AF1, 0878,FC ;24696          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2AF2, 8AD1,6C ;24697          JSR [DIVH]                    ;DO DIVIDE (T15 INSTRUCTION)
U 2AF3, 36B2,15 ;24698          MOV LS[T59] TO WR[0]          ;Setup to MOV FQ TO FT2
U 2AF4, 90F6,15 ;24699          MISC2 [TRAP.ACC] WR[0]        ;MOV FQ TO FT2
U 2AF5, 10F6,95 ;24700          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2AF6, 087A,1C ;24701          JSR [STORE.2.TRIPLE]          ;Store FT2 to the CPU
U 2AF7, B660,95 ;24702          MOV LS[BIT16] TO WR[1]        ;Setup expected data
U 2AF8, BEA4,95 ;24703          MOV WR[1] TO LS[DATA.2]
U 2AF9, E5A2,15 ;24704          CLR LS[DATA.1]
U 2AFA, 65A6,15 ;24705          CLR LS[DATA.3]
U 2AFB, 887C,6C ;24706          JSR [CHECK.SUB]
U 2AFC, 8AAE,34 ;24707          JMP [T43.1E]                  ;Loop on error
U 2AFD, FF82,15 ;24708          INC LS[ERROR.NUMBER]          ;Go onto error 1F
U 2AFE, 378C,15 ;24709  T43.1F: MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 2AFF, B716,95 ;24710          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 2B00, 90F6,15 ;24711          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2B01, 10F6,95 ;24712          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B02, B610,15 ;24713          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 2B03, 90F6,15 ;24714          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 2B04, 10F6,95 ;24715          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B05, B64C,15 ;24716          MOV LS[BIT6] TO WR[0]         ;Setup data
U 2B06, 3EA2,15 ;24717          MOV WR[0] TO LS[DATA.1]
U 2B07, E5A4,15 ;24718          CLR LS[DATA.2]
```

```
U 2B08, 65A6,15 ;24719          CLR LS[DATA.3]
U 2B09, 0878,FC ;24720          JSR [LOAD.TRIPLE]           ;Load FT0 with zeros and hidden bit
U 2B0A, 3618,15 ;24721          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2B0B, B716,95 ;24722          MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2B0C, 90F6,15 ;24723          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2B0D, 10F6,95 ;24724          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B0E, E5A2,15 ;24725          CLR LS[DATA.1]
U 2B0F, 0878,FC ;24726          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 2B10, 8AD1,6C ;24727          JSR [DIVH]                  ;DO DIVIDE (T15 INSTRUCTION)
U 2B11, 36B2,15 ;24728          MOV LS[T59] TO WR[0]        ;Setup to MOV FQ TO FT2
U 2B12, 90F6,15 ;24729          MISC2 [TRAP.ACC] WR[0]      ;MOV FQ TO FT2
U 2B13, 10F6,95 ;24730          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B14, 087A,1C ;24731          JSR [STORE.2.TRIPLE]        ;Store FT2 to the CPU
U 2B15, B660,95 ;24732          MOV LS[BIT16] TO WR[1]      ;Setup expected data
U 2B16, 4762,95 ;24733          BIS LS[BIT17] TO WR[1]
U 2B17, BEA4,95 ;24734          MOV WR[1] TO LS[DATA.2]
U 2B18, E5A2,15 ;24735          CLR LS[DATA.1]
U 2B19, 65A6,15 ;24736          CLR LS[DATA.3]
U 2B1A, 887C,6C ;24737          JSR [CHECK.SUB]
U 2B1B, 8AAF,E4 ;24738          JMP [T43.1F]                ;Loop on error
U 2B1C, FF82,15 ;24739          INC LS[ERROR.NUMBER]        ;Go onto error 20
U 2B1D, 378C,15 ;24740  T43.20: MOV LS[TC5] TO WR[0]        ;Setup to CLR FT0
U 2B1E, B716,95 ;24741          MOV LS[#300] TO WR[1]       ;Setup to loop on self
U 2B1F, 90F6,15 ;24742          MISC2 [TRAP.ACC] WR[0]      ;CLR FT0
U 2B20, 10F6,95 ;24743          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B21, B610,15 ;24744          MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 2B22, 90F6,15 ;24745          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 2B23, 10F6,95 ;24746          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B24, 087C,1C ;24747          JSR [CLR.FT0]               ;LOAD ZEROS INTO FT0
U 2B25, 3618,15 ;24748          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2B26, B716,95 ;24749          MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2B27, 90F6,15 ;24750          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2B28, 10F6,95 ;24751          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B29, 8AD1,6C ;24752          JSR [DIVH]                  ;DO DIVIDE (T15 INSTRUCTION)
U 2B2A, 36B2,15 ;24753          MOV LS[T59] TO WR[0]        ;Setup to MOV FQ TO FT2
U 2B2B, 90F6,15 ;24754          MISC2 [TRAP.ACC] WR[0]      ;MOV FQ TO FT2
U 2B2C, 10F6,95 ;24755          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B2D, 087A,1C ;24756          JSR [STORE.2.TRIPLE]        ;Store FT2 to the CPU
U 2B2E, 3662,95 ;24757          MOV LS[BIT17] TO WR[1]      ;Setup expected data
U 2B2F, BEA4,95 ;24758          MOV WR[1] TO LS[DATA.2]
U 2B30, E5A2,15 ;24759          CLR LS[DATA.1]
U 2B31, 65A6,15 ;24760          CLR LS[DATA.3]
U 2B32, 887C,6C ;24761          JSR [CHECK.SUB]
U 2B33, 8AB1,D4 ;24762          JMP [T43.20]                ;Loop on error
U 2B34, FF82,15 ;24763          INC LS[ERROR.NUMBER]        ;Go onto error 21
U 2B35, 378C,15 ;24764  T43.21: MOV LS[TC5] TO WR[0]        ;Setup to CLR FT0
U 2B36, B716,95 ;24765          MOV LS[#300] TO WR[1]       ;Setup to loop on self
U 2B37, 90F6,15 ;24766          MISC2 [TRAP.ACC] WR[0]      ;CLR FT0
U 2B38, 10F6,95 ;24767          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B39, B610,15 ;24768          MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 2B3A, 90F6,15 ;24769          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 2B3B, 10F6,95 ;24770          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B3C, B64C,15 ;24771          MOV LS[BIT6] TO WR[0]       ;Setup data
U 2B3D, 3EA2,15 ;24772          MOV WR[0] TO LS[DATA.1]
U 2B3E, E5A4,15 ;24773          CLR LS[DATA.2]
```

```
U 2B3F, 65A6,15 ;24774          CLR LS[DATA.3]
U 2B40, 0878,FC ;24775          JSR [LOAD.TRIPLE]           ;Load FT0 with zeros and hidden bit
U 2B41, 3618,15 ;24776          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2B42, B716,95 ;24777          MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2B43, 90F6,15 ;24778          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2B44, 10F6,95 ;24779          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B45, E5A2,15 ;24780          CLR LS[DATA.1]              ;Setup data
U 2B46, 0878,FC ;24781          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 2B47, B6B0,15 ;24782          MOV LS[T58] TO WR[0]        ;Setup to SHIFT RIGHT FT0 & FQ IN[ZERO]
U 2B48, B716,95 ;24783          MOV LS[#300] TO WR[1]       ;Loop on self
U 2B49, 90F6,15 ;24784          MISC2 [TRAP.ACC] WR[0]      ;SHIFT LEFT FT0 & FQ IN[ZERO]
U 2B4A, 10F6,95 ;24785          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B4B, 8AD1,6C ;24786          JSR [DIVH]                  ;DO DIVIDE (T15 INSTRUCTION)
U 2B4C, 36B2,15 ;24787          MOV LS[T59] TO WR[0]        ;Setup to MOV FQ TO FT2
U 2B4D, 90F6,15 ;24788          MISC2 [TRAP.ACC] WR[0]      ;MOV FQ TO FT2
U 2B4E, 10F6,95 ;24789          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B4F, 087A,1C ;24790          JSR [STORE.2.TRIPLE]        ;Store FT2 to the CPU
U 2B50, B660,95 ;24791          MOV LS[BIT16] TO WR[1]      ;Setup expected data
U 2B51, 4762,95 ;24792          BIS LS[BIT17] TO WR[1]
U 2B52, BEA4,95 ;24793          MOV WR[1] TO LS[DATA.2]
U 2B53, E5A2,15 ;24794          CLR LS[DATA.1]
U 2B54, 65A6,15 ;24795          CLR LS[DATA.3]
U 2B55, 887C,6C ;24796          JSR [CHECK.SUB]
U 2B56, 8AB3,54 ;24797          JMP [T43.21]                ;Loop on error
U 2B57, FF82,15 ;24798          INC LS[ERROR.NUMBER]        ;Go onto error 22
U 2B58, 378C,15 ;24799  T43.22: MOV LS[TC5] TO WR[0]        ;Setup to CLR FT0
U 2B59, B716,95 ;24800          MOV LS[#300] TO WR[1]       ;Setup to loop on self
U 2B5A, 90F6,15 ;24801          MISC2 [TRAP.ACC] WR[0]      ;CLR FT0
U 2B5B, 10F6,95 ;24802          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B5C, B610,15 ;24803          MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 2B5D, 90F6,15 ;24804          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 2B5E, 10F6,95 ;24805          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B5F, 087C,1C ;24806          JSR [CLR.FT0]               ;LOAD ZEROS INTO FT0
U 2B60, 3618,15 ;24807          MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2B61, B716,95 ;24808          MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2B62, 90F6,15 ;24809          MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2B63, 10F6,95 ;24810          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2B64, B64C,15 ;24811          MOV LS[BIT6] TO WR[0]       ;Setup data
U 2B65, 3EA2,15 ;24812          MOV WR[0] TO LS[DATA.1]
U 2B66, 0878,FC ;24813          JSR [LOAD.TRIPLE]           ;Load data into FT0
U 2B67, 3716,15 ;24814          MOV LS[#300] TO WR[0]       ;Set instruction and size bits to
U 2B68, C764,15 ;24815          BIS LS[BIT18] TO WR[0]      ; DIV and grand
U 2B69, C758,15 ;24816          BIS LS[BIT12] TO WR[0]
U 2B6A, C754,15 ;24817          BIS LS[BIT10] TO WR[0]
U 2B6B, C762,15 ;24818          BIS LS[BIT17] TO WR[0]
U 2B6C, 90F6,15 ;24819          MISC2 [TRAP.ACC] WR[0]
U 2B6D, 8AD1,6C ;24820          JSR [DIVH]                  ;DO DIVIDE (T15 INSTRUCTION)
U 2B6E, 3716,15 ;24821          MOV LS[#300] TO WR[0]       ;Set instruction and size bits to
U 2B6F, C764,15 ;24822          BIS LS[BIT18] TO WR[0]      ; DIV and single
U 2B70, C758,15 ;24823          BIS LS[BIT12] TO WR[0]
U 2B71, C754,15 ;24824          BIS LS[BIT10] TO WR[0]
U 2B72, 90F6,15 ;24825          MISC2 [TRAP.ACC] WR[0]
U 2B73, 36B2,15 ;24826          MOV LS[T59] TO WR[0]        ;Setup to MOV FQ TO FT2
U 2B74, 90F6,15 ;24827          MISC2 [TRAP.ACC] WR[0]      ;MOV FQ TO FT2
U 2B75, 10F6,95 ;24828          MISC2 [TRAP.ACC] WR[1]      ;Loop on self
```

```
U 2B76, 087A,1C ;24829          JSR [STORE.2.TRIPLE]          ;Store FT2 to the CPU
U 2B77, B660,95 ;24830          MOV LS[BIT16] TO WR[1]        ;Setup expected data
U 2B78, BEA4,95 ;24831          MOV WR[1] TO LS[DATA.2]
U 2B79, E5A2,15 ;24832          CLR LS[DATA.1]
U 2B7A, 65A6,15 ;24833          CLR LS[DATA.3]
U 2B7B, 887C,6C ;24834          JSR [CHECK.SUB]
U 2B7C, 0AB5,84 ;24835          JMP [T43.22]                  ;Loop on error
U 2B7D, FF82,15 ;24836          INC LS[ERROR.NUMBER]          ;Go onto error 23
U 2B7E, 378C,15 ;24837  T43.23: MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 2B7F, B716,95 ;24838          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 2B80, 90F6,15 ;24839          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2B81, 10F6,95 ;24840          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B82, B610,15 ;24841          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 2B83, 90F6,15 ;24842          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 2B84, 10F6,95 ;24843          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B85, B64C,15 ;24844          MOV LS[BIT6] TO WR[0]         ;Setup data
U 2B86, 3EA2,15 ;24845          MOV WR[0] TO LS[DATA.1]
U 2B87, E5A4,15 ;24846          CLR LS[DATA.2]
U 2B88, 65A6,15 ;24847          CLR LS[DATA.3]
U 2B89, 0878,FC ;24848          JSR [LOAD.TRIPLE]             ;Load FT0 with zeros and hidden bit
U 2B8A, 3618,15 ;24849          MOV LS[T12] TO WR[0]          ;Setup to MOV FT0 TO FT2
U 2B8B, B716,95 ;24850          MOV LS[#300] TO WR[1]         ;Setup to Loop on self
U 2B8C, 90F6,15 ;24851          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT2
U 2B8D, 10F6,95 ;24852          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B8E, E5A2,15 ;24853          CLR LS[DATA.1]
U 2B8F, 0878,FC ;24854          JSR [LOAD.TRIPLE]             ;Load data into FT0
U 2B90, 3716,15 ;24855          MOV LS[#300] TO WR[0]         ;Set instruction and size bits to
U 2B91, C764,15 ;24856          BIS LS[BIT18] TO WR[0]        ; DIV and grand
U 2B92, C758,15 ;24857          BIS LS[BIT12] TO WR[0]
U 2B93, C754,15 ;24858          BIS LS[BIT10] TO WR[0]
U 2B94, C762,15 ;24859          BIS LS[BIT17] TO WR[0]
U 2B95, 90F6,15 ;24860          MISC2 [TRAP.ACC] WR[0]
U 2B96, 8AD1,6C ;24861          JSR [DIVH]                    ;DO DIVIDE (T15 INSTRUCTION)
U 2B97, 3716,15 ;24862          MOV LS[#300] TO WR[0]         ;Set instruction and size bits to
U 2B98, C764,15 ;24863          BIS LS[BIT18] TO WR[0]        ; DIV and single
U 2B99, C758,15 ;24864          BIS LS[BIT12] TO WR[0]
U 2B9A, C754,15 ;24865          BIS LS[BIT10] TO WR[0]
U 2B9B, 90F6,15 ;24866          MISC2 [TRAP.ACC] WR[0]
U 2B9C, 36B2,15 ;24867          MOV LS[T59] TO WR[0]          ;Setup to MOV FQ TO FT2
U 2B9D, 90F6,15 ;24868          MISC2 [TRAP.ACC] WR[0]        ;MOV FQ TO FT2
U 2B9E, 10F6,95 ;24869          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2B9F, 087A,1C ;24870          JSR [STORE.2.TRIPLE]          ;Store FT2 to the CPU
U 2BA0, B660,95 ;24871          MOV LS[BIT16] TO WR[1]        ;Setup expected data
U 2BA1, 4762,95 ;24872          BIS LS[BIT17] TO WR[1]
U 2BA2, BEA4,95 ;24873          MOV WR[1] TO LS[DATA.2]
U 2BA3, E5A2,15 ;24874          CLR LS[DATA.1]
U 2BA4, 65A6,15 ;24875          CLR LS[DATA.3]
U 2BA5, 887C,6C ;24876          JSR [CHECK.SUB]
U 2BA6, 8AB7,E4 ;24877          JMP [T43.23]                  ;Loop on error
U 2BA7, FF82,15 ;24878          INC LS[ERROR.NUMBER]          ;Go onto error 24
U 2BA8, 378C,15 ;24879  T43.24: MOV LS[TC5] TO WR[0]          ;Setup to CLR FT0
U 2BA9, B716,95 ;24880          MOV LS[#300] TO WR[1]         ;Setup to loop on self
U 2BAA, 90F6,15 ;24881          MISC2 [TRAP.ACC] WR[0]        ;CLR FT0
U 2BAB, 10F6,95 ;24882          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2BAC, B610,15 ;24883          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
```

```
U 2BAD, 90F6,15 ;24884          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2BAE, 10F6,95 ;24885          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BAF, 087C,1C ;24886          JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 2BB0, 3618,15 ;24887          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2BB1, B716,95 ;24888          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2BB2, 90F6,15 ;24889          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2BB3, 10F6,95 ;24890          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BB4, 3716,15 ;24891          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2BB5, C764,15 ;24892          BIS LS[BIT18] TO WR[0]          ; DIV and grand
U 2BB6, C758,15 ;24893          BIS LS[BIT12] TO WR[0]
U 2BB7, C754,15 ;24894          BIS LS[BIT10] TO WR[0]
U 2BB8, C762,15 ;24895          BIS LS[BIT17] TO WR[0]
U 2BB9, 90F6,15 ;24896          MISC2 [TRAP.ACC] WR[0]
U 2BBA, 8AD1,6C ;24897          JSR [DIVH]                      ;DO DIVIDE (T15 INSTRUCTION)
U 2BBB, 3716,15 ;24898          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2BBC, C764,15 ;24899          BIS LS[BIT18] TO WR[0]          ; DIV and single
U 2BBD, C758,15 ;24900          BIS LS[BIT12] TO WR[0]
U 2BBE, C754,15 ;24901          BIS LS[BIT10] TO WR[0]
U 2BBF, 90F6,15 ;24902          MISC2 [TRAP.ACC] WR[0]
U 2BC0, 36B2,15 ;24903          MOV LS[T59] TO WR[0]            ;Setup to MOV FQ TO FT2
U 2BC1, 90F6,15 ;24904          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT2
U 2BC2, 10F6,95 ;24905          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BC3, 087A,1C ;24906          JSR [STORE.2.TRIPLE]            ;Store FT2 to the CPU
U 2BC4, 3662,95 ;24907          MOV LS[BIT17] TO WR[1]          ;Setup expected data
U 2BC5, BEA4,95 ;24908          MOV WR[1] TO LS[DATA.2]
U 2BC6, E5A2,15 ;24909          CLR LS[DATA.1]
U 2BC7, 65A6,15 ;24910          CLR LS[DATA.3]
U 2BC8, 887C,6C ;24911          JSR [CHECK.SUB]
U 2BC9, 0ABA,84 ;24912          JMP [T43.24]                    ;Loop on error
U 2BCA, FF82,15 ;24913          INC LS[ERROR.NUMBER]            ;Go onto error 25
U 2BCB, 378C,15 ;24914  T43.25: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2BCC, B716,95 ;24915          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2BCD, 90F6,15 ;24916          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2BCE, 10F6,95 ;24917          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BCF, B610,15 ;24918          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2BD0, 90F6,15 ;24919          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2BD1, 10F6,95 ;24920          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BD2, B64C,15 ;24921          MOV LS[BIT6] TO WR[0]           ;Setup data
U 2BD3, 3EA2,15 ;24922          MOV WR[0] TO LS[DATA.1]
U 2BD4, E5A4,15 ;24923          CLR LS[DATA.2]
U 2BD5, 65A6,15 ;24924          CLR LS[DATA.3]
U 2BD6, 0878,FC ;24925          JSR [LOAD.TRIPLE]               ;Load FT0 with zeros and hidden bit
U 2BD7, 3618,15 ;24926          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2BD8, B716,95 ;24927          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2BD9, 90F6,15 ;24928          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2BDA, 10F6,95 ;24929          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BDB, E5A2,15 ;24930          CLR LS[DATA.1]                  ;Setup data
U 2BDC, 0878,FC ;24931          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2BDD, B6B0,15 ;24932          MOV LS[T58] TO WR[0]            ;Setup to SHIFT RIGHT FT0 & FQ IN[ZERO]
U 2BDE, B716,95 ;24933          MOV LS[#300] TO WR[1]           ;Loop on self
U 2BDF, 90F6,15 ;24934          MISC2 [TRAP.ACC] WR[0]          ;SHIFT LEFT FT0 & FQ IN[ZERO]
U 2BE0, 10F6,95 ;24935          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BE1, 3716,15 ;24936          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2BE2, C764,15 ;24937          BIS LS[BIT18] TO WR[0]          ; DIV and grand
U 2BE3, C758,15 ;24938          BIS LS[BIT12] TO WR[0]
```

```
U 2BE4, C754,15 ;24939          BIS LS[BIT10] TO WR[0]
U 2BE5, C762,15 ;24940          BIS LS[BIT17] TO WR[0]
U 2BE6, 90F6,15 ;24941          MISC2 [TRAP.ACC] WR[0]
U 2BE7, 8AD1,6C ;24942          JSR [DIVH]                      ;DO DIVIDE (T15 INSTRUCTION)
U 2BE8, 3716,15 ;24943          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2BE9, C764,15 ;24944          BIS LS[BIT18] TO WR[0]          ; DIV and single
U 2BEA, C758,15 ;24945          BIS LS[BIT12] TO WR[0]
U 2BEB, C754,15 ;24946          BIS LS[BIT10] TO WR[0]
U 2BEC, 90F6,15 ;24947          MISC2 [TRAP.ACC] WR[0]
U 2BED, 36B2,15 ;24948          MOV LS[T59] TO WR[0]            ;Setup to MOV FQ TO FT2
U 2BEE, 90F6,15 ;24949          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT2
U 2BEF, 10F6,95 ;24950          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BF0, 087A,1C ;24951          JSR [STORE.2.TRIPLE]            ;Store FT2 to the CPU
U 2BF1, B660,95 ;24952          MOV LS[BIT16] TO WR[1]          ;Setup expected data
U 2BF2, 4762,95 ;24953          BIS LS[BIT17] TO WR[1]
U 2BF3, BEA4,95 ;24954          MOV WR[1] TO LS[DATA.2]
U 2BF4, E5A2,15 ;24955          CLR LS[DATA.1]
U 2BF5, 65A6,15 ;24956          CLR LS[DATA.3]
U 2BF6, 887C,6C ;24957          JSR [CHECK.SUB]
U 2BF7, 0ABC,B4 ;24958          JMP [T43.25]                    ;Loop on error
U 2BF8, FF82,15 ;24959          INC LS[ERROR.NUMBER]            ;Go onto error 26
U 2BF9, 378C,15 ;24960  T43.26: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2BFA, B716,95 ;24961          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2BFB, 90F6,15 ;24962          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2BFC, 10F6,95 ;24963          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2BFD, B610,15 ;24964          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2BFE, 90F6,15 ;24965          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2BFF, 10F6,95 ;24966          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C00, B67C,15 ;24967          MOV LS[BIT30] TO WR[0]          ;Setup data
U 2C01, 3EA2,15 ;24968          MOV WR[0] TO LS[DATA.1]
U 2C02, E5A4,15 ;24969          CLR LS[DATA.2]
U 2C03, 65A6,15 ;24970          CLR LS[DATA.3]
U 2C04, 0878,FC ;24971          JSR [LOAD.TRIPLE]               ;Load FT0 with DATA
U 2C05, 3618,15 ;24972          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2C06, B716,95 ;24973          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2C07, 90F6,15 ;24974          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2C08, 10F6,95 ;24975          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C09, 0AD1,DC ;24976          JSR [DIVL]                      ;SET INST AND SIZE TO DIVL AND SINGLE
                ;24977                                          ; DO INTEGER DIVIDE
                ;24978                                          ;SET INST AND SIZE TO DIV AND SINGLE
                ;24979                                          ;MOV FQ TO FT2 AND STORE
U 2C0A, B642,95 ;24980          MOV LS[BIT1] TO WR[1]           ;Setup expected data
U 2C0B, BEA4,95 ;24981          MOV WR[1] TO LS[DATA.2]
U 2C0C, E5A2,15 ;24982          CLR LS[DATA.1]
U 2C0D, 65A6,15 ;24983          CLR LS[DATA.3]
U 2C0E, 887C,6C ;24984          JSR [CHECK.SUB]
U 2C0F, 8ABF,94 ;24985          JMP [T43.26]                    ;Loop on error
U 2C10, FF82,15 ;24986          INC LS[ERROR.NUMBER]            ;Go onto error 27
U 2C11, 378C,15 ;24987  T43.27: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2C12, B716,95 ;24988          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2C13, 90F6,15 ;24989          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2C14, 10F6,95 ;24990          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C15, B610,15 ;24991          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2C16, 90F6,15 ;24992          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2C17, 10F6,95 ;24993          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
```

```
U 2C18, B67C,15 ;24994          MOV LS[BIT30] TO WR[0]
U 2C19, C77A,15 ;24995          BIS LS[BIT29] TO WR[0]
U 2C1A, 3EA2,15 ;24996          MOV WR[0] TO LS[DATA.1]
U 2C1B, E5A4,15 ;24997          CLR LS[DATA.2]
U 2C1C, 65A6,15 ;24998          CLR LS[DATA.3]
U 2C1D, 0878,FC ;24999          JSR [LOAD.TRIPLE]               ;Load FT0 with DATA
U 2C1E, 3618,15 ;25000          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2C1F, B716,95 ;25001          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2C20, 90F6,15 ;25002          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2C21, 10F6,95 ;25003          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C22, B67A,15 ;25004          MOV LS[BIT29] TO WR[0]          ;Setup data
U 2C23, 3EA2,15 ;25005          MOV WR[0] TO LS[DATA.1]
U 2C24, 0878,FC ;25006          JSR [LOAD.TRIPLE]               ;Load FT0 with data
U 2C25, 0AD1,DC ;25007          JSR [DIVL]                      ;SET INST AND SIZE TO DIVL AND SINGLE
                ;25008                                          ; DO INTEGER DIVIDE
                ;25009                                          ;SET INST AND SIZE TO DIV AND SINGLE
                ;25010                                          ;MOV FQ TO FT2 AND STORE
U 2C26, B642,95 ;25011          MOV LS[BIT1] TO WR[1]           ;Setup expected data
U 2C27, 4740,95 ;25012          BIS LS[BIT0] TO WR[1]
U 2C28, BEA4,95 ;25013          MOV WR[1] TO LS[DATA.2]
U 2C29, E5A2,15 ;25014          CLR LS[DATA.1]
U 2C2A, 65A6,15 ;25015          CLR LS[DATA.3]
U 2C2B, 887C,6C ;25016          JSR [CHECK.SUB]
U 2C2C, 0AC1,14 ;25017          JMP [T43.27]                    ;Loop on error
U 2C2D, FF82,15 ;25018          INC LS[ERROR.NUMBER]            ;Go onto error 28
U 2C2E, 378C,15 ;25019  T43.28: MOV LS[TC5] TO WR[0]            ;Setup to CLR FT0
U 2C2F, B716,95 ;25020          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2C30, 90F6,15 ;25021          MISC2 [TRAP.ACC] WR[0]          ;CLR FT0
U 2C31, 10F6,95 ;25022          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C32, B610,15 ;25023          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2C33, 90F6,15 ;25024          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2C34, 10F6,95 ;25025          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C35, B67C,15 ;25026          MOV LS[BIT30] TO WR[0]          ;Setup data
U 2C36, 3EA2,15 ;25027          MOV WR[0] TO LS[DATA.1]
U 2C37, E5A4,15 ;25028          CLR LS[DATA.2]
U 2C38, 65A6,15 ;25029          CLR LS[DATA.3]
U 2C39, 0878,FC ;25030          JSR [LOAD.TRIPLE]               ;Load FT0 with DATA
U 2C3A, 3618,15 ;25031          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2C3B, B716,95 ;25032          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2C3C, 90F6,15 ;25033          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2C3D, 10F6,95 ;25034          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C3E, B67A,15 ;25035          MOV LS[BIT29] TO WR[0]          ;Setup data
U 2C3F, C77C,15 ;25036          BIS LS[BIT30] TO WR[0]
U 2C40, 3EA2,15 ;25037          MOV WR[0] TO LS[DATA.1]
U 2C41, 0878,FC ;25038          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2C42, 0AD1,DC ;25039          JSR [DIVL]                      ;SET INST AND SIZE TO DIVL AND SINGLE
                ;25040                                          ; DO INTEGER DIVIDE
                ;25041                                          ;SET INST AND SIZE TO DIV AND SINGLE
                ;25042                                          ;MOV FQ TO FT2 AND STORE
U 2C43, 3640,95 ;25043          MOV LS[BIT0] TO WR[1]           ;Setup expected data
U 2C44, BEA4,95 ;25044          MOV WR[1] TO LS[DATA.2]
U 2C45, E5A2,15 ;25045          CLR LS[DATA.1]
U 2C46, 65A6,15 ;25046          CLR LS[DATA.3]
U 2C47, 887C,6C ;25047          JSR [CHECK.SUB]
U 2C48, 0AC2,E4 ;25048          JMP [T43.28]                    ;Loop on error
```

```
U 2C49, FF82,15 ;25049          INC LS[ERROR.NUMBER]            ;Go onto error 29
U 2C4A, 8870,3C ;25050          JSR [LO]                        ;Get address of MOV FT0 TO FT1
U 2C4B, 3E1A,15 ;25051          MOV WR[0] TO LS[T13]
U 2C4C, 8870,3C ;25052          JSR [LO]                        ;Get address of MOV FT0 TO FT3
U 2C4D, 3E16,15 ;25053          MOV WR[0] TO LS[T11]
U 2C4E, 8870,3C ;25054          JSR [LO]                        ;Get address of ADD.LOOP
U 2C4F, BE1E,15 ;25055          MOV WR[0] TO LS[T15]
U 2C50, 8870,3C ;25056          JSR [LO]                        ;Get address of SUB.LOOP
U 2C51, 3EAE,15 ;25057          MOV WR[0] TO LS[T57]
U 2C52, 8870,3C ;25058          JSR [LO]                        ;Get addresss of SHIFT RIGHT FT0
U 2C53, BE14,15 ;25059          MOV WR[0] TO LS[T10]            ; IN[ZERO]
U 2C54, 2F87,95 ;25060          CLR WR[3]                       ;Setup new third float error mask
U 2C55, 0874,3C ;25061          JSR [CLR.INST.AND.SIZE.BITS]    ;SET INSTRUCTION AND SIZE BITS TO ZERO
U 2C56, 087C,1C ;25062  T43.29: JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 2C57, B610,15 ;25063          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2C58, R716,95 ;25064          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2C59, 90F6,15 ;25065          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2C5A, 10F6,95 ;25066          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C5B, B61A,15 ;25067          MOV LS[T13] TO WR[0]            ;Setup to MOV FT0 TO FT1
U 2C5C, 90F6,15 ;25068          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 2C5D, 10F6,95 ;25069          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C5E, B616,15 ;25070          MOV LS[T11] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 2C5F, 90F6,15 ;25071          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 2C60, 10F6,95 ;25072          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C61, B64C,15 ;25073          MOV LS[BIT6] TO WR[0]           ;Setup data
U 2C62, 3EA2,15 ;25074          MOV WR[0] TO LS[DATA.1]
U 2C63, E5A4,15 ;25075          CLR LS[DATA.2]
U 2C64, 65A6,15 ;25076          CLR LS[DATA.3]
U 2C65, 0878,FC ;25077          JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2C66, 3618,15 ;25078          MOV LS[T12] TO WR[0]            ;Setup to MOV FT0 TO FT2
U 2C67, B716,95 ;25079          MOV LS[#300] TO WR[1]           ;Setup to Loop on self
U 2C68, 90F6,15 ;25080          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT2
U 2C69, 10F6,95 ;25081          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
J 2C6A, 0AD3,3C ;25082          JSR [INST.SIZE.DIV.HUGE]        ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25083                                          ; AND HUGE
U 2C6B, 361E,15 ;25084          MOV LS[T15] TO WR[0]            ;Setup to add loop
U 2C6C, 8AD3,BC ;25085          JSR [ADD.OR.SUB]                ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25086                                          ; MOV FQ TO FT2 AND STORE
U 2C6D, E5A2,15 ;25087          CLR LS[DATA.1]
U 2C6E, E5A4,15 ;25088          CLR LS[DATA.2]
U 2C6F, 65A6,15 ;25089          CLR LS[DATA.3]
U 2C70, 887C,6C ;25090          JSR [CHECK.SUB]
U 2C71, 0AC5,64 ;25091          JMP [T43.29]                    ;Loop on error
U 2C72, FF82,15 ;25092          INC LS[ERROR.NUMBER]            ;Go onto error 2A
U 2C73, 087C,1C ;25093  T43.2A: JSR [CLR.FT0]                   ;LOAD ZEROS INTO FT0
U 2C74, B610,15 ;25094          MOV LS[T8] TO WR[0]             ;Setup to MOV FT0 TO FQ
U 2C75, B716,95 ;25095          MOV LS[#300] TO WR[1]           ;Setup to loop on self
U 2C76, 90F6,15 ;25096          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FQ
U 2C77, 10F6,95 ;25097          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C78, B61A,15 ;25098          MOV LS[T13] TO WR[0]            ;Setup to MOV FT0 TO FT1
U 2C79, 90F6,15 ;25099          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT1
U 2C7A, 10F6,95 ;25100          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2C7B, B616,15 ;25101          MOV LS[T11] TO WR[0]            ;Setup to MOV FT0 TO FT3
U 2C7C, 90F6,15 ;25102          MISC2 [TRAP.ACC] WR[0]          ;MOV FT0 TO FT3
U 2C7D, 10F6,95 ;25103          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
```

```
U 2C7E, B64C,15 ;25104          MOV LS[BIT6] TO WR[0]          ;Setup data
U 2C7F, 3EA2,15 ;25105          MOV WR[0] TO LS[DATA.1]
U 2C80, E5A4,15 ;25106          CLR LS[DATA.2]
U 2C81, 65A6,15 ;25107          CLR LS[DATA.3]
U 2C82, 0878,FC ;25108          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2C83, 3618,15 ;25109          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 2C84, B716,95 ;25110          MOV LS[#300] TO WR[1]          ;Setup to Loop on self
U 2C85, 90F6,15 ;25111          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 2C86, 10F6,95 ;25112          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2C87, 0AD3,3C ;25113          JSR [INST.SIZE.DIV.HUGE]       ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25114                                         ; AND HUGE
U 2C88, B6AE,15 ;25115          MOV LS[T57] TO WR[0]           ;Setup to sub loop
U 2C89, 8AD3,BC ;25116          JSR [ADD.OR.SUB]               ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25117                                         ; MOV FQ TO FT2 AND STORE
U 2C8A, 3652,95 ;25118          MOV LS[BIT9] TO WR[1]          ;Setup expected data
U 2C8B, 3EA6,95 ;25119          MOV WR[1] TO LS[DATA.3]
U 2C8C, E5A4,15 ;25120          CLR LS[DATA.2]
U 2C8D, E5A2,15 ;25121          CLR LS[DATA.1]
U 2C8E, 887C,6C ;25122          JSR [CHECK.SUB]
U 2C8F, 8AC7,34 ;25123          JMP [T43.2A]                   ;Loop on error
U 2C90, FF82,15 ;25124          INC LS[ERROR.NUMBER]           ;Go onto error 2B
U 2C91, 087C,1C ;25125  T43.2B: JSR [CLR.FT0]                  ;LOAD ZEROS INTO FT0
U 2C92, B610,15 ;25126          MOV LS[T8] TO WR[0]            ;Setup to MOV FT0 TO FQ
U 2C93, B716,95 ;25127          MOV LS[#300] TO WR[1]
U 2C94, 90F6,15 ;25128          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FQ
U 2C95, 10F6,95 ;25129          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2C96, B61A,15 ;25130          MOV LS[T13] TO WR[0]           ;Setup to MOV FT0 TO FT1
U 2C97, 90F6,15 ;25131          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT1
U 2C98, 10F6,95 ;25132          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2C99, B616,15 ;25133          MOV LS[T11] TO WR[0]           ;Setup to MOV FT0 TO FT3
U 2C9A, 90F6,15 ;25134          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT3
U 2C9B, 10F6,95 ;25135          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2C9C, B64A,15 ;25136          MOV LS[BIT5] TO WR[0]          ;Setup data
U 2C9D, 3EA2,15 ;25137          MOV WR[0] TO LS[DATA.1]
U 2C9E, E5A4,15 ;25138          CLR LS[DATA.2]
U 2C9F, 65A6,15 ;25139          CLR LS[DATA.3]
U 2CA0, 0878,FC ;25140          JSR [LOAD.TRIPLE]
U 2CA1, 3618,15 ;25141          MOV LS[T12] TO WR[0]           ;Setup to MOV FT0 TO FT2
U 2CA2, B716,95 ;25142          MOV LS[#300] TO WR[1]          ;Setup to Loop on self
U 2CA3, 90F6,15 ;25143          MISC2 [TRAP.ACC] WR[0]         ;MOV FT0 TO FT2
U 2CA4, 10F6,95 ;25144          MISC2 [TRAP.ACC] WR[1]         ;Loop on self
U 2CA5, B64C,15 ;25145          MOV LS[BIT6] TO WR[0]          ;Setup data
U 2CA6, 3EA2,15 ;25146          MOV WR[0] TO LS[DATA.1]
U 2CA7, 0878,FC ;25147          JSR [LOAD.TRIPLE]              ;Load data into FT0
U 2CA8, 0AD3,3C ;25148          JSR [INST.SIZE.DIV.HUGE]       ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25149                                         ; AND HUGE
U 2CA9, 361E,15 ;25150          MOV LS[T15] TO WR[0]           ;Setup to add loop
U 2CAA, 8AD3,BC ;25151          JSR [ADD.OR.SUB]               ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25152                                         ; MOV FQ TO FT2 AND STORE
U 2CAB, E5A2,15 ;25153          CLR LS[DATA.1]
U 2CAC, E5A4,15 ;25154          CLR LS[DATA.2]
U 2CAD, 65A6,15 ;25155          CLR LS[DATA.3]
U 2CAE, 887C,6C ;25156          JSR [CHECK.SUB]
U 2CAF, 8AC9,14 ;25157          JMP [T43.2B]                   ;Loop on error
U 2CB0, FF82,15 ;25158          INC LS[ERROR.NUMBER]           ;Go onto error 2C
```

```
U 2CB1, 087C,1C ;25159  T43.2C: JSR [CLR.FT0]                ;LOAD ZEROS INTO FT0
U 2CB2, B610,15 ;25160           MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 2CB3, B716,95 ;25161           MOV LS[#300] TO WR[1]       ;Setup Loop self
U 2CB4, 90F6,15 ;25162           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 2CB5, 10F6,95 ;25163           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CB6, B61A,15 ;25164           MOV LS[T13] TO WR[0]        ;Setup to MOV FT0 TO FT1
U 2CB7, 90F6,15 ;25165           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT1
U 2CB8, 10F6,95 ;25166           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CB9, B616,15 ;25167           MOV LS[T11] TO WR[0]        ;Setup to MOV FT0 TO FT3
U 2CBA, 90F6,15 ;25168           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT3
U 2CBB, 10F6,95 ;25169           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CBC, B64A,15 ;25170           MOV LS[BIT5] TO WR[0]       ;Setup data
U 2CBD, 3EA2,15 ;25171           MOV WR[0] TO LS[DATA.1]
U 2CBE, E5A4,15 ;25172           CLR LS[DATA.2]
U 2CBF, 65A6,15 ;25173           CLR LS[DATA.3]
U 2CC0, 0878,FC ;25174           JSR [LOAD.TRIPLE]
U 2CC1, 3618,15 ;25175           MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2CC2, B716,95 ;25176           MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2CC3, 90F6,15 ;25177           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2CC4, 10F6,95 ;25178           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CC5, B64C,15 ;25179           MOV LS[BIT6] TO WR[0]       ;Setup data
U 2CC6, 3EA2,15 ;25180           MOV WR[0] TO LS[DATA.1]
U 2CC7, 0878,FC ;25181           JSR [LOAD.TRIPLE]           ;Load data into FT0
U 2CC8, 0AD3,3C ;25182           JSR [INST.SIZE.DIV.HUGE]    ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25183                                       ; AND HUGE
U 2CC9, B6AE,15 ;25184           MOV LS[T57] TO WR[0]        ;Setup to sub loop
U 2CCA, 8AD3,BC ;25185           JSR [ADD.OR.SUB]            ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25186                                       ; MOV FQ TO FT2 AND STORE
U 2CCB, B650,95 ;25187           MOV LS[BIT8] TO WR[1]       ;Setup expected data
U 2CCC, 3EA6,95 ;25188           MOV WR[1] TO LS[DATA.3]
U 2CCD, E5A4,15 ;25189           CLR LS[DATA.2]
U 2CCE, E5A2,15 ;25190           CLR LS[DATA.1]
U 2CCF, 887C,6C ;25191           JSR [CHECK.SUB]
U 2CD0, 0ACB,14 ;25192           JMP [T43.2C]                ;Loop on error
U 2CD1, FF82,15 ;25193           INC LS[ERROR.NUMBER]        ;Go onto error 2D
U 2CD2, 087C,1C ;25194  T43.2D: JSR [CLR.FT0]                ;LOAD ZEROS INTO FT0
U 2CD3, B610,15 ;25195           MOV LS[T8] TO WR[0]         ;Setup to MOV FT0 TO FQ
U 2CD4, B716,95 ;25196           MOV LS[#300] TO WR[1]       ;Setup to Loop self
U 2CD5, 90F6,15 ;25197           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FQ
U 2CD6, 10F6,95 ;25198           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CD7, B61A,15 ;25199           MOV LS[T13] TO WR[0]        ;Setup to MOV FT0 TO FT1
U 2CD8, 90F6,15 ;25200           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT1
U 2CD9, 10F6,95 ;25201           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CDA, B616,15 ;25202           MOV LS[T11] TO WR[0]        ;Setup to MOV FT0 TO FT3
U 2CDB, 90F6,15 ;25203           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT3
U 2CDC, 10F6,95 ;25204           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
U 2CDD, B64A,15 ;25205           MOV LS[BIT5] TO WR[0]       ;Setup data
U 2CDE, 3EA2,15 ;25206           MOV WR[0] TO LS[DATA.1]
U 2CDF, E5A4,15 ;25207           CLR LS[DATA.2]
U 2CE0, 65A6,15 ;25208           CLR LS[DATA.3]
U 2CE1, 0878,FC ;25209           JSR [LOAD.TRIPLE]
U 2CE2, 3618,15 ;25210           MOV LS[T12] TO WR[0]        ;Setup to MOV FT0 TO FT2
U 2CE3, B716,95 ;25211           MOV LS[#300] TO WR[1]       ;Setup to Loop on self
U 2CE4, 90F6,15 ;25212           MISC2 [TRAP.ACC] WR[0]      ;MOV FT0 TO FT2
U 2CE5, 10F6,95 ;25213           MISC2 [TRAP.ACC] WR[1]      ;Loop on self
```

```
U 2CE6, 3614,15 ;25214          MOV LS[T10] TO WR[0]          ;Setup to SHIFT LEFT FT0 IN[ZERO]
U 2CE7, 90F6,15 ;25215          MISC2 [TRAP.ACC] WR[0]        ;SHIFT LEFT FT0 IN[ONE]
U 2CE8, 10F6,95 ;25216          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 2CE9, 0AD3,3C ;25217          JSR [INST.SIZE.DIV.HUGE]      ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25218                                        ; AND HUGE
U 2CEA, 361E,15 ;25219          MOV LS[T15] TO WR[0]          ;Setup to add loop
U 2CEB, 8AD3,BC ;25220          JSR [ADD.OR.SUB]              ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25221                                        ; MOV FQ TO FT2 AND STORE
U 2CEC, 3652,95 ;25222          MOV LS[BIT9] TO WR[1]         ;Setup expected data
U 2CED, 3EA6,95 ;25223          MOV WR[1] TO LS[DATA.3]
U 2CEE, E5A4,15 ;25224          CLR LS[DATA.2]
U 2CEF, E5A2,15 ;25225          CLR LS[DATA.1]
U 2CF0, 887C,6C ;25226          JSR [CHECK.SUB]
U 2CF1, 0ACD,24 ;25227          JMP [T43.2D]                  ;Loop on error
U 2CF2, FF82,15 ;25228          INC LS[ERROR.NUMBER]          ;Go onto error 2E
U 2CF3, 087C,1C ;25229  T43.2E: JSR [CLR.FT0]                 ;LOAD ZEROS INTO FT0
U 2CF4, B610,15 ;25230          MOV LS[T8] TO WR[0]           ;Setup to MOV FT0 TO FQ
U 2CF5, B716,95 ;25231          MOV LS[#300] TO WR[1]         ;Setup Loop self
U 2CF6, 90F6,15 ;25232          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FQ
U 2CF7, 10F6,95 ;25233          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2CF8, B61A,15 ;25234          MOV LS[T13] TO WR[0]          ;Setup to MOV FT0 TO FT1
U 2CF9, 90F6,15 ;25235          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT1
U 2CFA, 10F6,95 ;25236          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2CFB, B616,15 ;25237          MOV LS[T11] TO WR[0]          ;Setup to MOV FT0 TO FT3
U 2CFC, 90F6,15 ;25238          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT3
U 2CFD, 10F6,95 ;25239          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2CFE, 087C,1C ;25240          JSR [CLR.FT0]                 ;LOAD ZEROS INTO FT0
U 2CFF, 3614,15 ;25241          MOV LS[T10] TO WR[0]          ;Setup to SHIFT LEFT FT0 IN[ZERO]
U 2D00, B716,95 ;25242          MOV LS[#300] TO WR[1]         ;Setup to Loop on self
U 2D01, 90F6,15 ;25243          MISC2 [TRAP.ACC] WR[0]        ;SHIFT LEFT FT0 IN[ZERO]
U 2D02, 10F6,95 ;25244          MISC2 [TRAP.ACC] WR[1]        ;Loop self
U 2D03, 3618,15 ;25245          MOV LS[T12] TO WR[0]          ;Setup to MOV FT0 TO FT2
U 2D04, 90F6,15 ;25246          MISC2 [TRAP.ACC] WR[0]        ;MOV FT0 TO FT2
U 2D05, 10F6,95 ;25247          MISC2 [TRAP.ACC] WR[1]        ;Loop on self
U 2D06, 0AD3,3C ;25248          JSR [INST.SIZE.DIV.HUGE]      ;SET INSTRUCTION AND SIZE BITS TO DIV
                ;25249                                        ; AND HUGE
U 2D07, 361E,15 ;25250          MOV LS[T15] TO WR[0]          ;Setup to add loop
U 2D08, 8AD3,BC ;25251          JSR [ADD.OR.SUB]              ;DO ADD OR SUB, CLEAR INST AND SIZE,
                ;25252                                        ; MOV FQ TO FT2 AND STORE
U 2D09, B650,95 ;25253          MOV LS[BIT8] TO WR[1]         ;Setup expected data
U 2D0A, 4752,95 ;25254          BIS LS[BIT9] TO WR[1]
U 2D0B, 3EA6,95 ;25255          MOV WR[1] TO LS[DATA.3]
U 2D0C, E5A4,15 ;25256          CLR LS[DATA.2]
U 2D0D, E5A2,15 ;25257          CLR LS[DATA.1]
U 2D0E, 887C,6C ;25258          JSR [CHECK.SUB]
U 2D0F, 0ACF,34 ;25259          JMP [T43.2E]                  ;Loop on error
                ;25260
U 2D10, 0AD4,A4 ;25261          JMP [T43.END]                 ;DONE
                ;25262
                ;25263  ;
                ;25264  ; SUBROUTINE TO MUL ADD FT2 TO FT4
                ;25265  ;
                ;25266  MUL.ADD.FT2.FT4:
U 2D11, 361E,15 ;25267          MOV LS[T15] TO WR[0]          ;Setup to MUL ADD FT2 TO FT4
U 2D12, 90F6,15 ;25268          MISC2 [TRAP.ACC] WR[0]        ;MUL ADD FT2 TO FT4
```

```
U 2D13, DB00,15 ;25269          NOP
U 2D14, 10F6,95 ;25270          MISC2 [TRAP.ACC] WR[1]
U 2D15, 5B00,14 ;25271          RETURN                          ;DONE WITH MUL ADD
                ;25272
                ;25273  ;
                ;25274  ; SUBROUTINE TO DO DIVH
                ;25275  ;
                ;25276  DIVH:
U 2D16, 361E,15 ;25277          MOV LS[T15] TO WR[0]            ;Setup to do the divide steps
U 2D17, 90F6,15 ;25278          MISC2 [TRAP.ACC] WR[0]          ;Do the divide
U 2D18, B716,95 ;25279          MOV LS[#300] TO WR[1]           ;Setup to loop on SELF
U 2D19, DB00,15 ;25280          NOP
U 2D1A, DB00,15 ;25281          NOP
U 2D1B, 10F6,95 ;25282          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2D1C, 5B00,14 ;25283          RETURN                          ;DONE WITH DIVH
                ;25284
                ;25285  ;
                ;25286  ; SUBROUTINE TO DO DIVL TESTS
                ;25287  ;
                ;25288  DIVL:
U 2D1D, 3716,15 ;25289          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2D1E, C764,15 ;25290          BIS LS[BIT18] TO WR[0]          ; DIVL and single
U 2D1F, 475C,15 ;25291          BIS LS[BIT14] TO WR[0]
U 2D20, 475A,15 ;25292          BIS LS[BIT13] TO WR[0]
U 2D21, 4756,15 ;25293          BIS LS[BIT11] TO WR[0]
U 2D22, 90F6,15 ;25294          MISC2 [TRAP.ACC] WR[0]
U 2D23, B6AE,15 ;25295          MOV LS[T57] TO WR[0]            ;Setup for integer divide
U 2D24, B716,95 ;25296          MOV LS[#300] TO WR[1]           ;Setup for loop on error
U 2D25, 90F6,15 ;25297          MISC2 [TRAP.ACC] WR[0]          ;Integer divide
U 2D26, DB00,15 ;25298          NOP
U 2D27, DB00,15 ;25299          NOP
U 2D28, 10F6,95 ;25300          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2D29, 3716,15 ;25301          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2D2A, C764,15 ;25302          BIS LS[BIT18] TO WR[0]          ; DIV and single
U 2D2B, C758,15 ;25303          BIS LS[BIT12] TO WR[0]
U 2D2C, C754,15 ;25304          BIS LS[BIT10] TO WR[0]
U 2D2D, 90F6,15 ;25305          MISC2 [TRAP.ACC] WR[0]
U 2D2E, 36B2,15 ;25306          MOV LS[T59] TO WR[0]            ;Setup to MOV FQ TO FT2
U 2D2F, 90F6,15 ;25307          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT2
U 2D30, 10F6,95 ;25308          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2D31, 087A,1C ;25309          JSR [STORE.2.TRIPLE]            ;Store FT2 to the CPU
U 2D32, 5B00,14 ;25310          RETURN                          ;CHECK RESULT
                ;25311
                ;25312  ;
                ;25313  ; SUBROUTINE TO SET INSTRUCTION AND SIZE BITS TO DIV AND HUGE
                ;25314  ;
                ;25315  INST.SIZE.DIV.HUGE:
U 2D33, 3716,15 ;25316          MOV LS[#300] TO WR[0]           ;Set instruction and size bits to
U 2D34, C764,15 ;25317          BIS LS[BIT18] TO WR[0]          ; DIV and HUGE
U 2D35, C762,15 ;25318          BIS LS[BIT17] TO WR[0]
U 2D36, 4760,15 ;25319          BIS LS[BIT16] TO WR[0]
U 2D37, C758,15 ;25320          BIS LS[BIT12] TO WR[0]
U 2D38, C754,15 ;25321          BIS LS[BIT10] TO WR[0]
U 2D39, 90F6,15 ;25322          MISC2 [TRAP.ACC] WR[0]
U 2D3A, 5B00,14 ;25323          RETURN                          ;DONE
```

```
                ;25324  ;
                ;25325  ; SUBROUTINE TO DO ADD OR SUB LOOP AND STORE RESULT IN CPU
                ;25326  ;
                ;25327  ADD.OR.SUB:
U 2D3B, B716,95 ;25328          MOV LS[#300] TO WR[1]           ;LOOP SELF INSTRUCTION
U 2D3C, 90F6,15 ;25329          MISC2 [TRAP.ACC] WR[0]          ;Force to ADD LOOP
U 2D3D, DB00,15 ;25330          NOP                             ;Wait during FPA MICROCODE
U 2D3E, DB00,15 ;25331          NOP
U 2D3F, DB00,15 ;25332          NOP
U 2D40, DB00,15 ;25333          NOP
U 2D41, DB00,15 ;25334          NOP
U 2D42, DB00,15 ;25335          NOP
U 2D43, 10F6,95 ;25336          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2D44, 0874,3C ;25337          JSR [CLR.INST.AND.SIZE.BITS]    ;CLEAR INSTRUCTION AND SIZE BITS
U 2D45, 36B2,15 ;25338          MOV LS[T59] TO WR[0]            ;Setup to MOV FQ TO FT2
U 2D46, 90F6,15 ;25339          MISC2 [TRAP.ACC] WR[0]          ;MOV FQ TO FT2
U 2D47, 10F6,95 ;25340          MISC2 [TRAP.ACC] WR[1]          ;Loop on self
U 2D48, 087A,1C ;25341          JSR [STORE.2.TRIPLE]            ;Store FT2 to the CPU
U 2D49, 5B00,14 ;25342          RETURN                          ;GO CHECK RESULTS
                ;25343
                ;25344  T43.END:
```

```
                ;25345  .PAGE
                ;25346  .TOC    "TEST 44 INSTRUCTION ROM TEST(M8389 FPA MODULE)"
                ;25347  ;++
                ;25348  ;
                ;25349  ;
                ;25350  ; Test Description:
                ;25351  ;
                ;25352  ;       The purpose of this test is to check the INSTRUCTION PROM. The most
                ;25353  ;       direct way of testing the outputs of the PROM is by using the micro-
                ;25354  ;       sequencer. The outputs of the PROM are the direct inputs to the micro-
                ;25355  ;       sequencer and as such can be branched on if the location being branched
                ;25356  ;       from contains XXXX00000XXX. In this manner all of the PROM locations
                ;25357  ;       can be tested for instruction bits. The size bits can be tested by
                ;25358  ;       branching if it is a real instruction. The inputs to the PROM are the
                ;25359  ;       IB BUS from the CPU that can be defined by issuing a DECODE instruction
                ;25360  ;       and FPAA EXTEND FUNC (1)H. EXTEND FUNC will be asserted if the opcode
                ;25361  ;       is FD and IRD STATE is asserted.
                ;25362  ;
                ;25363  ; Assumptions:
                ;25364  ;
                ;25365  ;       It is assumed that the branch tests have been run previously and
                ;25366  ;       found to be successful.
                ;25367  ;
                ;25368  ; Test Steps:
                ;25369  ;
                ;25370  ;       1) Issue a DECODE instruction to set the DECODE ROM bits. This will
                ;25371  ;          cause a jump to someplace in the control store. From the CPU force a
                ;25372  ;          return from subroutine in the FPA as a call to a fetch routine is
                ;25373  ;          the first instruction after a DECODE isntruction. Force a read to
                ;25374  ;          see what the micro PC is. If there is an error issue error1 and
                ;25375  ;          the location of failure.
                ;25376  ;
                ;25377  ; Error:
                ;25378  ;
                ;25379  ;       Error1 - Branch to control store failed.
                ;25380  ;
                ;25381  ; Debug:
                ;25382  ;
                ;25383  ;       Error1: There are two reasons likely for an error to occur, they are
                ;25384  ;               the PROM is faulty or the microsequencer is faulty. By
                ;25385  ;               examining the error statements one of the two possibilities
                ;25386  ;               will probably be more likely. If the failure is associated
                ;25387  ;               with just one output then it's likely that the error is a
                ;25388  ;               result of microsequncer failure. If the error is a single
                ;25389  ;               error there's a good possibility that the PROM is at fault.
                ;25390  ;               If half of the ROM is in error then it's likely that EXTEND
                ;25391  ;               FUNC is in error. If this is the case then the gates and
                ;25392  ;               flip flop coming from the IRD ROM should be checked for error.
                ;25393  ;
                ;25394  ;
                ;25395  ;--
                ;25396  T.44:
U 2D4A, B65E,15 ;25397          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;25398                                          ; STATUS WORD
U 2D4B, 3E80,15 ;25399          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
```

```
                ;25400                                          ; BIT 15 INDICATES START OF TEST TO
                ;25401                                          ; CONSOLE
U 2D4C, 10E0,15 ;25402          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;25403  WAIT.T44.0:
U 2D4D, 8AD4,D4 ;25404          JMP [WAIT.T44.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;25405                                          ; RESPOND
U 2D4E, 087E,6C ;25406          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;25407                                          ; AND SIZE BITS
U 2D4F, B700,15 ;25408          MOV LS[FFFFFC00.MASK] TO WR[0]  ;Setup error mask
U 2D50, C740,15 ;25409          BIS LS[BIT0] TO WR[0]
U 2D51, 4742,15 ;25410          BIS LS[BIT1] TO WR[0]
U 2D52, 4744,15 ;25411          BIS LS[BIT2] TO WR[0]
U 2D53, 4750,15 ;25412          BIS LS[BIT8] TO WR[0]
U 2D54, C752,15 ;25413          BIS LS[BIT9] TO WR[0]
U 2D55, 3E8A,15 ;25414          MOV WR[0] TO LS[ERROR.MASK]
U 2D56, B670,15 ;25415          MOV LS[OTHER.DATA] TO WR[0]     ;Setup other data field
U 2D57, 3E80,15 ;25416          MOV WR[0] TO LS[CONTROL.STATUS]
U 2D58, B7B4,15 ;25417          MOV LS[T44.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2D59, BE12,15 ;25418          MOV WR[0] TO LS[T9]
U 2D5A, 8870,3C ;25419          JSR [L0]                        ;Get address of RETURN
U 2D5B, 3E16,15 ;25420          MOV WR[0] TO LS[T11]
U 2D5C, 8870,3C ;25421          JSR [L0]                        ;Get address of BRANCH EXTENDED on SIZE
U 2D5D, BE18,15 ;25422          MOV WR[0] TO LS[T12]            ; bits
U 2D5E, 2F85,15 ;25423          CLR WR[2]
U 2D5F, 8AE0,2C ;25424          JSR [L8]                        ;Get first memory word
U 2D60, 3E15,15 ;25425          MOV WR[2] TO LS[T10]            ;Setup to Load up memory for the decode
                ;25426  T44.PROM.COUNT:
U 2D61, 9956,75 ;25427          MEM.REQ[WRITE.P] ADRS[#800] DT[LONG]
U 2D62, B214,15 ;25428          WRITE.MEM LS[T10]
U 2D63, B615,15 ;25429          MOV LS[T10] TO WR[2]            ;Move the counter to WR 2
                ;25430          CMP WR[2] WITH LS[#100],        ;Check for last location
U 2D64, CF51,35 ;25431           DT(LONG)&SET.ALU.CC
U 2D65, 8ADA,B9 ;25432          JMP.IF[EQL] TO [T44.EXT.INSTR]
U 2D66, 65FC,15 ;25433          CLR LS[SIZE]                    ;Setup to compare byte
                ;25434          CMP WR[2] WITH LS[T8],          ;Compare counter with instruction words
U 2D67, 4F11,55 ;25435           DT(SIZE)&SET.ALU.CC            ; in memory
U 2D68, 8AD7,C9 ;25436          JMP.IF[EQL] TO [T44.INSTR.CHECK];If equal go to instruction check
U 2D69, E520,15 ;25437  T44.1:  CLR LS[PC]
U 2D6A, 9B57,75 ;25438          MEM.REQ[IB.FILL] ADRS[#800] DT[LONG]
                ;25439                                          ;Else read memory to IB
                ;25440
U 2D6B, 8407,1E ;25441          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25442                                          ;Eanble the first byte of the IB onto
                ;25443                                          ;the BUS and assert IRD State
U 2D6C, DB00,15 ;25444          NOP
U 2D6D, DB00,15 ;25445          NOP
U 2D6E, 10F8,15 ;25446          MISC2 [READ.ACC.UPC]            ;Enable the NUA onto the FPA BUS
U 2D6F, BA1A,15 ;25447          MOV FPA TO LS[T13]              ;Enable the FPA BUS onto the Y BUS
U 2D70, 3614,15 ;25448          MOV LS[T10] TO WR[0]            ;Setup counter value in other data
U 2D71, BE88,15 ;25449          MOV WR[0] TO LS[ADDRESS.DATA]   ; field
U 2D72, B61A,15 ;25450          MOV LS[T13] TO WR[0]            ;Setup received data
U 2D73, B64E,95 ;25451          MOV LS[#80] TO WR[1]
U 2D74, 474C,95 ;25452          BIS LS[#40] TO WR[1]
U 2D75, 474A,95 ;25453          BIS LS[#20] TO WR[1]
U 2D76, 4752,95 ;25454          BIS LS[#200] TO WR[1]
```

```
U 2D77, 4740,95 ;25455          BIS LS[BIT0] TO WR[1]
U 2D78, 0869,3C ;25456          JSR [CHECK.RESULT]                  ;Check for error
U 2D79, 8AD6,94 ;25457          JMP [T44.1]                         ;Loop on error
U 2D7A, FF14,15 ;25458          INC LS[T10]                         ;Add one to the counter
U 2D7B, 0AD6,14 ;25459          JMP [T44.PROM.COUNT]                ;Go onto next prom location
                ;25460  T44.INSTR.CHECK:
U 2D7C, 9B57,75 ;25461          MEM.REQ[IB.FILL] ADRS[#800] DT[LONG]
                ;25462                                              ;Else read memory to IB
U 2D7D, E520,15 ;25463          CLR LS[PC]
U 2D7E, 8407,1E ;25464          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25465                                              ;Eanble the first byte of the IB onto
                ;25466                                              ;the BUS and assert IRD State
U 2D7F, DB00,15 ;25467          NOP
U 2D80, DB00,15 ;25468          NOP
U 2D81, B616,15 ;25469          MOV LS[T11] TO WR[0]                ;Setup to RETURN
U 2D82, B716,95 ;25470          MOV LS[#300] TO WR[1]               ;Setup to Loop self
U 2D83, 90F6,15 ;25471          MISC2 [TRAP.ACC] WR[0]              ;Return
U 2D84, 10F8,15 ;25472          MISC2 [READ.ACC.UPC]                ;Enable NUA BUS onto the FPA BUS
U 2D85, BA1A,15 ;25473          MOV FPA TO LS[T13]                  ;Enable FPA BUS onto the Y BUS
U 2D86, 10F6,95 ;25474          MISC2 [TRAP.ACC] WR[1]              ;Loop self
U 2D87, 3614,15 ;25475          MOV LS[T10] TO WR[0]                ;Put Prom location into other data
U 2D88, BE88,15 ;25476          MOV WR[0] TO LS[ADDRESS.DATA]
U 2D89, 3611,15 ;25477          MOV LS[T8] TO WR[2]
U 2D8A, 2F87,95 ;25478          CLR WR[3]                           ;Shift right 8 times to get the
                ;25479  T44.SHIFT:
U 2D8B, 2351,15 ;25480          ASHR WR[2]                          ; the expected data into the first
U 2D8C, 2047,95 ;25481          INC WR[3]                           ; byte
U 2D8D, B646,15 ;25482          MOV LS[#8] TO WR[0]
                ;25483          CMP WR[3] WITH WR[0],
U 2D8E, A646,35 ;25484           DT(LONG)&SET.ALU.CC
U 2D8F, 8AD8,B1 ;25485          JMP.IF[NEQ] TO [T44.SHIFT]
U 2D90, A004,95 ;25486          MOV WR[2] TO WR[1]                  ;Setup expected data
U 2D91, B61A,15 ;25487          MOV LS[T13] TO WR[0]                ;Setup received data
U 2D92, 0869,3C ;25488          JSR [CHECK.RESULT]                  ;Check for error
U 2D93, 0AD7,C4 ;25489          JMP [T44.INSTR.CHECK]               ;Loop on error
                ;25490  T44.SIZE:
U 2D94, BE11,15 ;25491          MOV WR[2] TO LS[T8]                 ;Move the shifted down data back to T8
U 2D95, 8AE0,8C ;25492          JSR [GET.EX.DA.FOR.SIZE]            ;Set expected data for size bits
U 2D96, 3618,15 ;25493          MOV LS[T12] TO WR[0]                ;Set up to branch on size bits
U 2D97, 90F6,15 ;25494          MISC2 [TRAP.ACC] WR[0]              ;Branch on size bits
U 2D98, 10F8,15 ;25495          MISC2 [READ.ACC.UPC]                ;Enable NUA BUS onto the FPA BUS
U 2D99, BA1A,15 ;25496          MOV FPA TO LS[T13]                  ;Enable FPA BUS onto Y BUS
U 2D9A, B61A,15 ;25497          MOV LS[T13] TO WR[0]                ;Setup received data
U 2D9B, B701,95 ;25498          MOV LS[FFFFFC00.MASK] TO WR[3]      ;Setup error mask
U 2D9C, C741,95 ;25499          BIS LS[BIT0] TO WR[3]
U 2D9D, 4743,95 ;25500          BIS LS[BIT1] TO WR[3]
U 2D9E, 3E8B,95 ;25501          MOV WR[3] TO LS[ERROR.MASK]
U 2D9F, 0869,3C ;25502          JSR [CHECK.RESULT]                  ;Check for error
U 2DA0, 0AD9,44 ;25503          JMP [T44.SIZE]                      ;Loop on error
U 2DA1, B701,95 ;25504          MOV LS[FFFFFC00.MASK] TO WR[3]      ;Reset error mask
U 2DA2, C741,95 ;25505          BIS LS[BIT0] TO WR[3]
U 2DA3, 4743,95 ;25506          BIS LS[BIT1] TO WR[3]
U 2DA4, 4745,95 ;25507          BIS LS[BIT2] TO WR[3]
U 2DA5, 4751,95 ;25508          BIS LS[BIT8] TO WR[3]
U 2DA6, C753,95 ;25509          BIS LS[BIT9] TO WR[3]
```

```
U 2DA7, 3E8B,95 ;25510          MOV WR[3] TO LS[ERROR.MASK]
U 2DA8, 8AE0,2C ;25511          JSR [L8]                        ;Get next memory word
U 2DA9, FF14,15 ;25512          INC LS[T10]                     ;Add on to the counter and go onto
U 2DAA, 0AD6,14 ;25513          JMP [T44.PROM.COUNT]            ; next prom location
                ;25514  T44.EXT.INSTR:
U 2DAB, B651,15 ;25515          MOV LS[#100] TO WR[2]           ;Reset counter and load into WR and
U 2DAC, 3E15,15 ;25516          MOV WR[2] TO LS[T10]            ; LS location
                ;25517  T44.EXT.PROM.COUNT:
U 2DAD, B615,15 ;25518          MOV LS[T10] TO WR[2]            ;Move the counter to WR 2
                ;25519          CMP WR[2] WITH LS[#200],        ;Check for last location
U 2DAE, 4F53,35 ;25520           DT(LONG)&SET.ALU.CC
U 2DAF, 8AE2,09 ;25521          JMP.IF[EQL] TO [T44.END]
U 2DB0, B626,15 ;25522          MOV LS[#FF] TO WR[0]            ;Setup to put FD on the IB BUS for an
U 2DB1, 2100,15 ;25523          DEC WR[0]                       ; extended instruction
U 2DB2, 2100,15 ;25524          DEC WR[0]
U 2DB3, 3E1C,15 ;25525          MOV WR[0] TO LS[T14]
U 2DB4, 1956,35 ;25526          MEM.REQ[WRITE.P] ADRS[#800] DT[WORD]
U 2DB5, 321C,15 ;25527          WRITE.MEM LS[T14]
U 2DB6, 9B57,75 ;25528          MEM.REQ[IB.FILL] ADRS[#800] DT[LONG]
                ;25529                                          ;Load the IB with extended instruction
U 2DB7, E520,15 ;25530          CLR LS[PC]
U 2DB8, 8407,1E ;25531          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25532                                          ;Put the IB on the BUS and assert IRD
U 2DB9, DB00,15 ;25533          NOP
U 2DBA, DB00,15 ;25534          NOP
U 2DBB, 1956,35 ;25535          MEM.REQ[WRITE.P] ADRS[#800] DT[WORD]
                ;25536                                          ;write the counter to memory
U 2DBC, B214,15 ;25537          WRITE.MEM LS[T10]
U 2DBD, 65FC,15 ;25538          CLR LS[SIZE]                    ;Setup to compare byte
                ;25539          CMP WR[2] WITH LS[T8],          ;Compare counter with instruction words
U 2DBE, 4F11,55 ;25540           DT(SIZE)&SET.ALU.CC            ; in memory
U 2DBF, 8ADD,39 ;25541          JMP.IF[EQL] TO [T44.EXT.INSTR.CHECK]
                ;25542                                          ;If equal go to instruction check
                ;25543  T44.EXT.1:
U 2DC0, 9B57,75 ;25544          MEM.REQ[IB.FILL] ADRS[#800] DT[LONG]
                ;25545                                          ;Else read memory to IB
U 2DC1, E520,15 ;25546          CLR LS[PC]
U 2DC2, 8407,1E ;25547          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25548                                          ;Eanble the first byte of the IB onto
                ;25549                                          ;the BUS and assert IRD State
U 2DC3, DB00,15 ;25550          NOP
U 2DC4, DB00,15 ;25551          NOP
U 2DC5, 10F8,15 ;25552          MISC2 [READ.ACC.UPC]            ;Enable the NUA onto the FPA BUS
U 2DC6, BA1A,15 ;25553          MOV FPA TO LS[T13]              ;Enable the FPA BUS onto the Y BUS
U 2DC7, 3614,15 ;25554          MOV LS[T10] TO WR[0]            ;Setup counter value in other data
U 2DC8, BE88,15 ;25555          MOV WR[0] TO LS[ADDRESS.DATA]   ; field
U 2DC9, B61A,15 ;25556          MOV LS[T13] TO WR[0]            ;Setup received data
U 2DCA, B64E,95 ;25557          MOV LS[#80] TO WR[1]
U 2DCB, 474C,95 ;25558          BIS LS[#40] TO WR[1]
U 2DCC, 474A,95 ;25559          BIS LS[#20] TO WR[1]
U 2DCD, 4752,95 ;25560          BIS LS[#200] TO WR[1]
U 2DCE, 4740,95 ;25561          BIS LS[BIT0] TO WR[1]
U 2DCF, 0869,3C ;25562          JSR [CHECK.RESULT]              ;Check for error
U 2DD0, 8AD6,94 ;25563          JMP [T44.1]                     ;Loop on error
U 2DD1, FF14,15 ;25564          INC LS[T10]                     ;Add one to the counter
```

```
U 2DD2, 0ADA,D4 ;25565          JMP [T44.EXT.PROM.COUNT]          ;Go onto next prom location
                ;25566  T44.EXT.INSTR.CHECK:
U 2DD3, 9B57,75 ;25567          MEM.REQ[IB.FILL] ADRS[#800] DT[LONG]
                ;25568                                            ;Else read memory to IB
U 2DD4, E520,15 ;25569          CLR LS[PC]
U 2DD5, 8407,1E ;25570          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25571                                            ;Eanble the first byte of the IB onto
                ;25572                                            ;the BUS and assert IRD State
U 2DD6, DB00,15 ;25573          NOP
U 2DD7, DB00,15 ;25574          NOP
U 2DD8, B616,15 ;25575          MOV LS[T11] TO WR[0]              ;Setup to RETURN
U 2DD9, B716,95 ;25576          MOV LS[#300] TO WR[1]             ;Setup to Loop self
U 2DDA, 90F6,15 ;25577          MISC2 [TRAP.ACC] WR[0]            ;Return
U 2DDB, 10F8,15 ;25578          MISC2 [READ.ACC.UPC]              ;Enable NUA BUS onto the FPA BUS
U 2DDC, BA1A,15 ;25579          MOV FPA TO LS[T13]                ;Enable FPA BUS onto the Y BUS
U 2DDD, 10F6,95 ;25580          MISC2 [TRAP.ACC] WR[1]            ;Loop self
U 2DDE, 3614,15 ;25581          MOV LS[T10] TO WR[0]              ;Put Prom location into other data
U 2DDF, BE88,15 ;25582          MOV WR[0] TO LS[ADDRESS.DATA]
U 2DE0, 3611,15 ;25583          MOV LS[T8] TO WR[2]
U 2DE1, 2F87,95 ;25584          CLR WR[3]                         ;Shift right 8 times to get the
                ;25585  T44.EXT.SHIFT:
U 2DE2, 2351,15 ;25586          ASHR WR[2]                        ; the expected data into the first
U 2DE3, 2047,95 ;25587          INC WR[3]                         ; byte
U 2DE4, B646,15 ;25588          MOV LS[#8] TO WR[0]
                ;25589          CMP WR[3] WITH WR[0],
U 2DE5, A646,35 ;25590           DT(LONG)&SET.ALU.CC
U 2DE6, 8ADE,21 ;25591          JMP.IF[NEQ] TO [T44.EXT.SHIFT]
U 2DE7, A004,95 ;25592          MOV WR[2] TO WR[1]                ;Setup expected data
U 2DE8, B61A,15 ;25593          MOV LS[T13] TO WR[0]              ;Setup received data
U 2DE9, 0869,3C ;25594          JSR [CHECK.RESULT]                ;Check for error
U 2DEA, 0ADD,34 ;25595          JMP [T44.EXT.INSTR.CHECK]         ;Loop on error
                ;25596  T44.EXT.SIZE:
U 2DEB, BE11,15 ;25597          MOV WR[2] TO LS[T8]               ;Move the shifted down data back to T8
U 2DEC, 8AE0,8C ;25598          JSR [GET.EX.DA.FOR.SIZE]          ;Set expected data for size bits
U 2DED, 3618,15 ;25599          MOV LS[T12] TO WR[0]              ;Set up to branch on size bits
U 2DEE, 90F6,15 ;25600          MISC2 [TRAP.ACC] WR[0]            ;Branch on size bits
U 2DEF, 10F8,15 ;25601          MISC2 [READ.ACC.UPC]              ;Enable NUA BUS onto the FPA BUS
U 2DF0, BA1A,15 ;25602          MOV FPA TO LS[T13]                ;Enable FPA BUS onto Y BUS
U 2DF1, B61A,15 ;25603          MOV LS[T13] TO WR[0]              ;Setup received data
U 2DF2, B701,95 ;25604          MOV LS[FFFFFC00.MASK] TO WR[3]    ;Setup error mask
U 2DF3, C741,95 ;25605          BIS LS[BIT0] TO WR[3]
U 2DF4, 4743,95 ;25606          BIS LS[BIT1] TO WR[3]
U 2DF5, 3E8B,95 ;25607          MOV WR[3] TO LS[ERROR.MASK]
U 2DF6, 0869,3C ;25608          JSR [CHECK.RESULT]                ;Check for error
U 2DF7, 8ADE,B4 ;25609          JMP [T44.EXT.SIZE]                ;Loop on error
U 2DF8, B701,95 ;25610          MOV LS[FFFFFC00.MASK] TO WR[3]
U 2DF9, C741,95 ;25611          BIS LS[BIT0] TO WR[3]
U 2DFA, 4743,95 ;25612          BIS LS[BIT1] TO WR[3]
U 2DFB, 4745,95 ;25613          BIS LS[BIT2] TO WR[3]
U 2DFC, 4751,95 ;25614          BIS LS[BIT8] TO WR[3]
U 2DFD, C753,95 ;25615          BIS LS[BIT9] TO WR[3]
U 2DFE, 3E8B,95 ;25616          MOV WR[3] TO LS[ERROR.MASK]
U 2DFF, 8AE0,2C ;25617          JSR [L8]                          ;Get next memory word
U 2E00, FF14,15 ;25618          INC LS[T10]                       ;Add on to the counter and go onto
U 2E01, 0ADA,D4 ;25619          JMP [T44.EXT.PROM.COUNT]          ; next prom location
```

```
                    ;25620
                    ;25621
U 2E02, 9813,B5 ;25622  L8:     MEM.REQ[READ.P] ADRS[T9] DT[WORD]       ;Accesses the memory location
                    ;25623                                              ; defined by th contents of T9
U 2E03, 3022,15 ;25624          MOV MEM.DATA TO WR[0]                   ;Load the contents of the mem-
                    ;25625                                              ; ory location accessed into 0
U 2E04, 3E10,15 ;25626          MOV WR[0] TO LS[T8]
U 2E05, FF12,15 ;25627          INC LS[T9]                              ;Update pointer to memory
U 2E06, FF12,15 ;25628          INC LS[T9]
U 2E07, 5B00,14 ;25629          RETURN
                    ;25630
                    ;25631
                    ;25632  ;       The purpose of this subroutine is to get the size bits which are in
                    ;25633  ;       the two LSB's of T8 and translate then into the address that will be
                    ;25634  ;       branched to when an extended branch on the size bits are done.
                    ;25635
                    ;25636  GET.EX.DA.FOR.SIZE:
U 2E08, B640,15 ;25637          MOV LS[BIT0] TO WR[0]                   ;Setup to mask out all but the 2 LSB's
U 2E09, 4742,15 ;25638          BIS LS[BIT1] TO WR[0]
U 2E0A, 6E10,15 ;25639          AND WR[0] TO LS[T8]                     ;Mask out all but two LSB's
                    ;25640          CMP WR[0] WITH LS[T8],                  ;If size bits are not 11
U 2E0B, CF10,35 ;25641           DT(LONG)&SET.ALU.CC
U 2E0C, 8AE1,11 ;25642          JMP.IF[NEQ] TO [SIZE.1]                 ; then go onto next combination
U 2E0D, B716,95 ;25643          MOV LS[#300] TO WR[1]                   ; else setup expected data
U 2E0E, 4746,95 ;25644          BIS LS[BIT3] TO WR[1]
U 2E0F, C748,95 ;25645          BIS LS[BIT4] TO WR[1]
U 2E10, 5B00,14 ;25646          RETURN
U 2E11, 2F80,15 ;25647  SIZE.1: CLR WR[0]                               ;Setup WR[0]
                    ;25648          CMP WR[0] WITH LS[T8],                  ;If size bits are not 00
U 2E12, CF10,35 ;25649           DT(LONG)&SET.ALU.CC
U 2E13, 0AE1,61 ;25650          JMP.IF[NEQ] TO [SIZE.2]                 ; then go onto next combination
U 2E14, B716,95 ;25651          MOV LS[#300] TO WR[1]                   ;Setup expected data
U 2E15, 5B00,14 ;25652          RETURN
U 2E16, C740,15 ;25653  SIZE.2: BIS LS[BIT0] TO WR[0]                   ;Setup WR[0]
                    ;25654          CMP WR[0] WITH LS[T8],                  ;If size bits are not 01
U 2E17, CF10,35 ;25655           DT(LONG)&SET.ALU.CC
U 2E18, 0AE1,C1 ;25656          JMP.IF[NEQ] TO [SIZE.3]                 ; then go onto next combination
U 2E19, B716,95 ;25657          MOV LS[#300] TO WR[1]                   ;Setup expected data
U 2E1A, 4746,95 ;25658          BIS LS[BIT3] TO WR[1]
U 2E1B, 5B00,14 ;25659          RETURN
U 2E1C, 2F82,95 ;25660  SIZE.3: CLR WR[1]                               ;Setup expected data
U 2E1D, B716,95 ;25661          MOV LS[#300] TO WR[1]
U 2E1E, C748,95 ;25662          BIS LS[BIT4] TO WR[1]
U 2E1F, 5B00,14 ;25663          RETURN
                    ;25664
                    ;25665
                    ;25666  T44.END:
```

```
;25667  .PAGE
;25668  .TOC    "TEST 45 CLOCK SYNC TEST VIA TOGGLE CLOCK(M8389 FPA MODULE)"
;25669  ;++
;25670  ;
;25671  ;
;25672  ; Test Description:
;25673  ;
;25674  ;       The purpose of this test is to check the synchronization of the FPA
;25675  ;       with the CPU when the method of speeding and slowing the FPA clocks
;25676  ;       is TOGGLE CLOCK. If the clock field is set to 010 and the mod field
;25677  ;       is set to extend clock then the fast clock flip-flop will be toggled.
;25678  ;       This flip flop can be toggled during PH0, PH1, or PH2 of the CPU. This
;25679  ;       test will check all three of these cases. This will be done by loading
;25680  ;       operands into the FPA and performing an add. The operands that are
;25681  ;       picked will dermine what clock cycle the synchronization occurs. Since
;25682  ;       this is possible three separate pairs of operands will be choosen to
;25683  ;       check the three clock cycles.  The other aspect to check is to see if
;25684  ;       the the fast clock speed is really running up to speed. This can be
;25685  ;       done by issuing a toggle clock exacuting some instructions and checking
;25686  ;       to see that they've executed in the right amount of time.
;25687  ;
;25688  ; Assumptions:
;25689  ;
;25690  ;       It is assumed that the NUA lines, the parity, the control store, and
;25691  ;       option sync have been tested previously and work.
;25692  ;
;25693  ; Test Steps:
;25694  ;
;25695  ;       1) Setup a memory loacation to do a DECODE to an ADD DOUBLE. Issue the
;25696  ;          the DECODE instruction.
;25697  ;       2) Setup up the operands and pass them to the FPA then LOOP ON NO
;25698  ;          INTERUPT and NO SYNC, SKIP IF SYNC.
;25699  ;       3) When you SKIP read the CC back then a float of data wait a cycle
;25700  ;          then read back another float of data. If there's and error issue
;25701  ;          error 1.
;25702  ;       4) Repeat steps 1 - 3 for another pair of operands. If there's an error
;25703  ;          then issue error 2.
;25704  ;       5) Repeat steps 1 - 3 for another pair of operands. If there's an error
;25705  ;          then issue error 3.
;25706  ;       6) Issue an instruction to toggle speed, wait a number of cylces then
;25707  ;          force a loop self. Then check to see if all the instructions that
;25708  ;          had time to be executed were executed.
;25709  ;       7) Repeat instruction 6 except wait twice the number of cyles and
;25710  ;          check to see that all the instructions that should have been exa-
;25711  ;          cuted have been executed.
;25712  ;
;25713  ; Error:
;25714  ;
;25715  ;       Error1 - Error in clock sync during CPU PH0.
;25716  ;       Error2 - Error in clock sync during CPU PH1.
;25717  ;       Error3 - Error in clock sync during CPU PH2.
;25718  ;       Error4 - Error in the speed of the fast clock over three FPA cycles.
;25719  ;       Error5 - Error in the speed of the fast clock over three CPU cycles.
;25720  ;
;25721  ; Debug:
```

```
                ;25722  ;
                ;25723  ;       Error1: If this error occurs then the CLOCK GENER'N PAL should be
                ;25724  ;               checked for error. This error could be the result of the
                ;25725  ;               clock running at a speed that's too fast or too slow. The
                ;25726  ;               error will also occur if the clock doesn't slow down or
                ;25727  ;               speed up (if the flop is stuck high or low).
                ;25728  ;       Error2: Same as error 1 except the CPU phase is different.
                ;25729  ;       Error3: Same as error 1 except the CPU phase is different.
                ;25730  ;       Error4: Same as error 1.
                ;25731  ;       Error5: Same as error 1.
                ;25732  ;
                ;25733  ;
                ;25734  ;--
                ;25735  T.45:
U 2E20, B65E,15 ;25736          MOV LS[BEGIN.TEST] TO WR[0]     ;SET BIT 15 IN WR[0] FOR CONTROL AND`
                ;25737                                          ; STATUS WORD
U 2E21, 3E80,15 ;25738          MOV WR[0] TO LS[CONTROL.STATUS] ;SET BIT 15 IN CONTROL AND STATUS WORD
                ;25739                                          ; BIT 15 INDICATES START OF TEST TO
                ;25740                                          ; CONSOLE
U 2E22, 10E0,15 ;25741          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;25742  WAIT.T45.0:
U 2E23, 0AE2,34 ;25743          JMP [WAIT.T45.0]                ;LOOP HERE WAITING FOR CONSOLE TO
                ;25744                                          ; RESPOND
U 2E24, 087E,6C ;25745          JSR [SETUP]                     ;SET UP ERROR INFO AND CLEAR INSTRUC
                ;25746                                          ; AND SIZE BITS
U 2E25, 37B6,15 ;25747          MOV LS[T45.POINTER] TO WR[0]    ;Initialize pointer for main memory
U 2E26, BE12,15 ;25748          MOV WR[0] TO LS[T9]
U 2E27, 8870,3C ;25749          JSR [L0]                        ;Get address of Toggle fast clock
U 2E28, 3E10,15 ;25750          MOV WR[0] TO LS[T8]
U 2E29, 8870,3C ;25751          JSR [L0]                        ;Get address of MOV FT0 TO FT6
U 2E2A, 3E1C,15 ;25752          MOV WR[0] TO LS[T14]
U 2E2B, 8870,3C ;25753          JSR [L0]                        ;Get address of STORE FF FT6
U 2E2C, 3EAE,15 ;25754          MOV WR[0] TO LS[T57]
U 2E2D, 8870,3C ;25755          JSR [L0]                        ;Get address of STORE SF FT6
U 2E2E, 3EB0,15 ;25756          MOV WR[0] TO LS[T58]
U 2E2F, 8870,3C ;25757          JSR [L0]                        ;Get address of STORE TF FT6
U 2E30, BEB2,15 ;25758          MOV WR[0] TO LS[T59]
U 2E31, 3648,15 ;25759          MOV LS[#10] TO WR[0]            ;Setup to force INIT code
U 2E32, 90F6,15 ;25760          MISC2 [TRAP.ACC] WR[0]          ;Force INIT code
U 2E33, B716,95 ;25761          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 2E34, B64C,15 ;25762          MOV LS[#40] TO WR[0]            ;Setup counter
U 2E35, 4748,15 ;25763          BIS LS[#10] TO WR[0]
                ;25764  T45.INIT.LOOP:
U 2E36, 2100,15 ;25765          DEC WR[0]                       ;Decrement counter
                ;25766          CMP WR[0] WITH LS[#0],          ;Is counter zero yet?
U 2E37, 4F9C,35 ;25767           DT(LONG)&SET.ALU.CC
U 2E38, 8AE3,61 ;25768          JMP.IF[NEQ] TO [T45.INIT.LOOP]  ;If not continue looping
U 2E39, B64C,15 ;25769  T45.1:  MOV LS[BIT6] TO WR[0]           ;Setup data to store to memory
U 2E3A, C74A,15 ;25770          BIS LS[BIT5] TO WR[0]
U 2E3B, BE14,15 ;25771          MOV WR[0] TO LS[T10]
U 2E3C, 9956,75 ;25772          MEM.REQ[WRITE.P] ADRS[#800] DT[LONG]
                ;25773                                          ;Write T10 to memory
U 2E3D, B214,15 ;25774          WRITE.MEM LS[T10]
U 2E3E, 1B57,35 ;25775          MEM.REQ[IB.FILL] ADRS[#800] DT[WORD]
                ;25776                                          ;Load memory to instruction buffer
```

```
U 2E3F, E520,15 ;25777          CLR LS[PC]                          ;Clear pointer to instruction buffer
U 2E40, 8407,1E ;25778          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25779                                              ;Issue an add double to the FPA
U 2E41, DB00,15 ;25780          NOP
U 2E42, 3628,15 ;25781          MOV LS[#FFFF] TO WR[0]              ;Setup operand one
U 2E43, A0C0,15 ;25782          COM WR[0]
U 2E44, C726,15 ;25783          BIS LS[#FF] TO WR[0]
U 2E45, 475C,15 ;25784          BIS LS[BIT14] TO WR[0]
U 2E46, 369E,95 ;25785          MOV LS[ONES] TO WR[1]
U 2E47, 0AE4,8C ;25786          JSR [PUT.ADRS.STACK.1]
                ;25787  PUT.ADRS.STACK.1:
                ;25788          ASSERT.DA.AND.PASS.WR0,             ;Pass operand one
U 2E48, 90F4,17 ;25789          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2E49, DB00,15 ;25790          NOP
U 2E4A, 3E1E,95 ;25791          MOV WR[1] TO LS[T15]
U 2E4B, C54E,15 ;25792          BIC LS[BIT7] TO WR[0]               ;Setup operand two
U 2E4C, 0AE4,DC ;25793          JSR [PUT.ADRS.STACK.2]
                ;25794  PUT.ADRS.STACK.2:
                ;25795          ASSERT.DA.AND.PASS.WR0,             ;Pass operand two
U 2E4D, 90F4,17 ;25796          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2E4E, DB00,15 ;25797          NOP
U 2E4F, 3E1E,95 ;25798          MOV WR[1] TO LS[T15]
U 2E50, 8AE5,1C ;25799          JSR [PUT.ADRS.STACK.3]
                ;25800  PUT.ADRS.STACK.3:
                ;25801          MOV FPA TO LS[T11],                 ;Enable CC's to T11
U 2E51, 3A16,17 ;25802          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
                ;25803                                              ;Loop waiting for sync from the FPA
U 2E52, DB00,15 ;25804          NOP                                 ;SKIP on SYNC
U 2E53, 3A18,15 ;25805          MOV FPA TO LS[T12]                  ;Enable first float to T12
U 2E54, DB00,15 ;25806          NOP                                 ;Wait a cycle
U 2E55, BA1A,15 ;25807          MOV FPA TO LS[T13]                  ;Enable second float to T13
U 2E56, 3618,15 ;25808          MOV LS[T12] TO WR[0]                ;Setup received data
U 2E57, B628,95 ;25809          MOV LS[#FFFF] TO WR[1]              ;Setup expected data
U 2E58, A0C2,95 ;25810          COM WR[1]
U 2E59, 4726,95 ;25811          BIS LS[#FF] TO WR[1]
U 2E5A, 454E,95 ;25812          BIC LS[BIT7] TO WR[1]
U 2E5B, C750,95 ;25813          BIS LS[BIT8] TO WR[1]
U 2E5C, C75C,95 ;25814          BIS LS[BIT14] TO WR[1]
U 2E5D, C54C,95 ;25815          BIC LS[BIT6] TO WR[1]
U 2E5E, 0869,3C ;25816          JSR [CHECK.RESULT]                  ;Check for error
U 2E5F, 8AE3,94 ;25817          JMP [T45.1]                         ;Loop on error
U 2E60, FF82,15 ;25818          INC LS[ERROR.NUMBER]                ;Go onto  error 2
U 2E61, B64C,15 ;25819  T45.2:  MOV LS[BIT6] TO WR[0]               ;Setup data to store to memory
U 2E62, C74A,15 ;25820          BIS LS[BIT5] TO WR[0]
U 2E63, BE14,15 ;25821          MOV WR[0] TO LS[T10]
U 2E64, 9956,75 ;25822          MEM.REQ[WRITE.P] ADRS[#800] DT[LONG]
                ;25823                                              ;Write T10 to memory
U 2E65, B214,15 ;25824          WRITE.MEM LS[T10]
U 2E66, 1B57,35 ;25825          MEM.REQ[IB.FILL] ADRS[#800] DT[WORD]
                ;25826                                              ;Load memory to instruction buffer
U 2E67, E520,15 ;25827          CLR LS[PC]                          ;Clear pointer to instruction buffer
U 2E68, 8407,1E ;25828          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25829                                              ;Issue an add double to the FPA
U 2E69, DB00,15 ;25830          NOP
U 2E6A, 3628,15 ;25831          MOV LS[#FFFF] TO WR[0]              ;Setup operand one
```

```
U 2E6B, A0C0,15 ;25832          COM WR[0]
U 2E6C, C726,15 ;25833          BIS LS[#FF] TO WR[0]
U 2E6D, 475C,15 ;25834          BIS LS[BIT14] TO WR[0]
U 2E6E, C54E,15 ;25835          BIC LS[BIT7] TO WR[0]
U 2E6F, 4750,15 ;25836          BIS LS[BIT8] TO WR[0]
U 2E70, 369E,95 ;25837          MOV LS[ONES] TO WR[1]
U 2E71, 0AE7,2C ;25838          JSR [PUT.ADRS.STACK.4]
                ;25839  PUT.ADRS.STACK.4:
                ;25840          ASSERT.DA.AND.PASS.WR0,          ;Pass operand one
U 2E72, 90F4,17 ;25841          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2E73, DB00,15 ;25842          NOP
U 2E74, 3E1E,95 ;25843          MOV WR[1] TO LS[T15]
U 2E75, C54E,15 ;25844          BIC LS[BIT7] TO WR[0]            ;Setup operand two
U 2E76, C550,15 ;25845          BIC LS[BIT8] TO WR[0]
U 2E77, 0AE7,8C ;25846          JSR [PUT.ADRS.STACK.5]
                ;25847  PUT.ADRS.STACK.5:
                ;25848          ASSERT.DA.AND.PASS.WR0,          ;Pass operand two
U 2E78, 90F4,17 ;25849          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2E79, DB00,15 ;25850          NOP
U 2E7A, 3E1E,95 ;25851          MOV WR[1] TO LS[T15]
U 2E7B, 8AE7,CC ;25852          JSR [PUT.ADRS.STACK.6]
                ;25853  PUT.ADRS.STACK.6:
                ;25854          MOV FPA TO LS[T11],              ;Enable CC's to T11
U 2E7C, 3A16,17 ;25855          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
                ;25856                                           ;Loop waiting for sync from the FPA
U 2E7D, DB00,15 ;25857          NOP                              ;SKIP on SYNC
U 2E7E, 3A18,15 ;25858          MOV FPA TO LS[T12]               ;Enable first float to T12
U 2E7F, DB00,15 ;25859          NOP                              ;Wait a cycle
U 2E80, BA1A,15 ;25860          MOV FPA TO LS[T13]               ;Enable second float to T13
U 2E81, 3618,15 ;25861          MOV LS[T12] TO WR[0]             ;Setup received data
U 2E82, B628,95 ;25862          MOV LS[#FFFF] TO WR[1]           ;Setup expected data
U 2E83, A0C2,95 ;25863          COM WR[1]
U 2E84, 4726,95 ;25864          BIS LS[#FF] TO WR[1]
U 2E85, C750,95 ;25865          BIS LS[BIT8] TO WR[1]
U 2E86, C75C,95 ;25866          BIS LS[BIT14] TO WR[1]
U 2E87, C54A,95 ;25867          BIC LS[BIT5] TO WR[1]
U 2E88, C54C,95 ;25868          BIC LS[BIT6] TO WR[1]
U 2E89, 0869,3C ;25869          JSR [CHECK.RESULT]               ;Check for error
U 2E8A, 0AE6,14 ;25870          JMP [T45.2]                      ;Loop on error
U 2E8B, FF82,15 ;25871          INC LS[ERROR.NUMBER]             ;Go onto  error 3
U 2E8C, B64C,15 ;25872  T45.3:  MOV LS[BIT6] TO WR[0]            ;Setup data to store to memory
U 2E8D, C74A,15 ;25873          BIS LS[BIT5] TO WR[0]
U 2E8E, BE14,15 ;25874          MOV WR[0] TO LS[T10]
U 2E8F, 9956,75 ;25875          MEM.REQ[WRITE.P] ADRS[#800] DT[LONG]
                ;25876                                           ;Write T10 to memory
U 2E90, B214,15 ;25877          WRITE.MEM LS[T10]
U 2E91, 1B57,35 ;25878          MEM.REQ[IB.FILL] ADRS[#800] DT[WORD]
                ;25879                                           ;Load memory to instruction buffer
U 2E92, E520,15 ;25880          CLR LS[PC]                       ;Clear pointer to instruction buffer
U 2E93, 8407,1E ;25881          OPC2/DECODE,IFUNC/VAX.IRD,IB.REQ/IB.REQ,OPC.SPEC/VAX.IRD,JCTL/NO.JUMP.TST
                ;25882                                           ;Issue an add double to the FPA
U 2E94, DB00,15 ;25883          NOP
U 2E95, 3628,15 ;25884          MOV LS[#FFFF] TO WR[0]           ;Setup operand one
U 2E96, A0C0,15 ;25885          COM WR[0]
U 2E97, C726,15 ;25886          BIS LS[#FF] TO WR[0]
```

```
U 2E98, 475C,15 ;25887          BIS LS[BIT14] TO WR[0]
U 2E99, 4750,15 ;25888          BIS LS[BIT8] TO WR[0]
U 2E9A, 369E,95 ;25889          MOV LS[ONES] TO WR[1]
U 2E9B, 0AE9,CC ;25890          JSR [PUT.ADRS.STACK.7]
                ;25891  PUT.ADRS.STACK.7:
                ;25892          ASSERT.DA.AND.PASS.WR0,             ;Pass operand one
U 2E9C, 90F4,17 ;25893          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2E9D, DB00,15 ;25894          NOP
U 2E9E, 3E1E,95 ;25895          MOV WR[1] TO LS[T15]
U 2E9F, C54E,15 ;25896          BIC LS[BIT7] TO WR[0]               ;Setup operand two
U 2EA0, C550,15 ;25897          BIC LS[BIT8] TO WR[0]
U 2EA1, 8AEA,2C ;25898          JSR [PUT.ADRS.STACK.8]
                ;25899  PUT.ADRS.STACK.8:
                ;25900          ASSERT.DA.AND.PASS.WR0,             ;Pass operand two
U 2EA2, 90F4,17 ;25901          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
U 2EA3, DB00,15 ;25902          NOP
U 2EA4, 3E1E,95 ;25903          MOV WR[1] TO LS[T15]
U 2EA5, 0AEA,6C ;25904          JSR [PUT.ADRS.STACK.9]
                ;25905  PUT.ADRS.STACK.9:
                ;25906          MOV FPA TO LS[T11],                 ;Enable CC's to T11
U 2EA6, 3A16,17 ;25907          LOOP.IF(NO INTERRUPT AND NO SYNC) ELSE SKIP.IF(SYNC)
                ;25908                                              ;Loop waiting for sync from the FPA
U 2EA7, DB00,15 ;25909          NOP                                 ;SKIP on SYNC
U 2EA8, 3A18,15 ;25910          MOV FPA TO LS[T12]                  ;Enable first float to T12
U 2EA9, DB00,15 ;25911          NOP                                 ;Wait a cycle
U 2EAA, BA1A,15 ;25912          MOV FPA TO LS[T13]                  ;Enable second float to T13
U 2EAB, 3618,15 ;25913          MOV LS[T12] TO WR[0]                ;Setup received data
U 2EAC, B628,95 ;25914          MOV LS[#FFFF] TO WR[1]              ;Setup expected data
U 2EAD, A0C2,95 ;25915          COM WR[1]
U 2EAE, 4726,95 ;25916          BIS LS[#FF] TO WR[1]
U 2EAF, 454E,95 ;25917          BIC LS[BIT7] TO WR[1]
U 2EB0, 4752,95 ;25918          BIS LS[BIT9] TO WR[1]
U 2EB1, C75C,95 ;25919          BIS LS[BIT14] TO WR[1]
U 2EB2, C54C,95 ;25920          BIC LS[BIT6] TO WR[1]
U 2EB3, C54A,95 ;25921          BIC LS[BIT5] TO WR[1]
U 2EB4, 4548,95 ;25922          BIC LS[BIT4] TO WR[1]
U 2EB5, 0869,3C ;25923          JSR [CHECK.RESULT]                  ;Check for error
U 2EB6, 0AE8,C4 ;25924          JMP [T45.3]                         ;Loop on error
U 2EB7, FF82,15 ;25925          INC LS[ERROR.NUMBER]                ;Go onto  error 4
U 2EB8, E5A2,15 ;25926          CLR LS[DATA.1]                      ;Setup data
U 2EB9, E5A4,15 ;25927          CLR LS[DATA.2]
U 2EBA, 65A6,15 ;25928          CLR LS[DATA.3]
U 2EBB, 2F85,15 ;25929          CLR WR[2]
U 2EBC, 2045,15 ;25930  T45.4:  INC WR[2]
U 2EBD, 0878,FC ;25931          JSR [LOAD.TRIPLE]                   ;Load data into FT0
U 2EBE, 0878,AC ;25932          JSR [MOV.DATA]                      ;MOV FT0 TO FT6
U 2EBF, B610,15 ;25933          MOV LS[T8] TO WR[0]                 ;Setup to force fast clock
U 2EC0, B716,95 ;25934          MOV LS[#300] TO WR[1]               ;Setup to Loop self
U 2EC1, 90F6,15 ;25935          MISC2 [TRAP.ACC] WR[0]              ;Force fast clock
U 2EC2, DB00,15 ;25936          NOP                                 ;Speed clock
U 2EC3, DB00,15 ;25937          NOP                                 ;execute a total of 4 left shifts
U 2EC4, DB00,15 ;25938          NOP
U 2EC5, 10F6,95 ;25939          MISC2 [TRAP.ACC] WR[1]              ;Loop self
U 2EC6, 087A,BC ;25940          JSR [STORE.X.TRIPLE]                ;Store FT6 to the CPU
U 2EC7, B6A8,15 ;25941          MOV LS[FIRST.FLT] TO WR[0]          ;Setup received data
```

```
U 2EC8, 3654,95 ;25942          MOV LS[BIT10] TO WR[1]          ;Setup expected data
                ;25943          CMP WR[2] WITH LS[BIT1],        ;What pass is this
U 2EC9, CF43,35 ;25944           DT(LONG)&SET.ALU.CC
U 2ECA, 8AEC,D9 ;25945          JMP.IF[EQL] TO [T45.4A]
                ;25946          CMP WR[0] WITH WR[1],           ;If there is an error on the 1ST pass
U 2ECB, 2640,B5 ;25947           DT(LONG)&SET.ALU.CC            ; then loop back to avoid the WCS
U 2ECC, 0AEB,C1 ;25948          JMP.IF[NEQ] TO [T45.4]          ; refresh problem
U 2ECD, 0869,3C ;25949  T45.4A: JSR [CHECK.RESULT]              ;Check for error
U 2ECE, 0AEB,C4 ;25950          JMP [T45.4]                     ;Loop on error
U 2ECF, FF82,15 ;25951          INC LS[ERROR.NUMBER]            ;Go onto error 5
U 2ED0, 0878,FC ;25952  T45.5:  JSR [LOAD.TRIPLE]               ;Load data into FT0
U 2ED1, 0878,AC ;25953          JSR [MOV.DATA]                  ;MOV FT0 TO FT6
U 2ED2, B610,15 ;25954          MOV LS[T8] TO WR[0]             ;Setup to force fast clock
U 2ED3, B716,95 ;25955          MOV LS[#300] TO WR[1]           ;Setup to Loop self
U 2ED4, 90F6,15 ;25956          MISC2 [TRAP.ACC] WR[0]          ;Force fast clock
U 2ED5, DB00,15 ;25957          NOP                             ;Speed clock
U 2ED6, DB00,15 ;25958          NOP                             ;execute a total of 5 left shifts
U 2ED7, DB00,15 ;25959          NOP
U 2ED8, DB00,15 ;25960          NOP
U 2ED9, 10F6,95 ;25961          MISC2 [TRAP.ACC] WR[1]          ;Loop self
U 2EDA, 087A,BC ;25962          JSR [STORE.X.TRIPLE]            ;Store FT6 to the CPU
U 2EDB, B6A8,15 ;25963          MOV LS[FIRST.FLT] TO WR[0]      ;Setup received data
U 2EDC, 3658,95 ;25964          MOV LS[BIT12] TO WR[1]          ;Setup expected data
                ;25965          CMP WR[0] WITH WR[1],           ;If there is an error on the 1ST pass
U 2EDD, 2640,B5 ;25966           DT(LONG)&SET.ALU.CC            ; then loop back to avoid the WCS
U 2EDE, 0AED,01 ;25967          JMP.IF[NEQ] TO [T45.5]          ; refresh problem
U 2EDF, 0869,3C ;25968          JSR [CHECK.RESULT]              ;Check for error
U 2EE0, 0AED,04 ;25969          JMP [T45.5]                     ;Loop on error
                ;25970  END.SECTION:
U 2EE1, 365C,15 ;25971          MOV LS[BIT14] TO WR[0]          ;SET UP WR 0 FOR END OF SECTION
U 2EE2, 3E80,15 ;25972          MOV WR[0] TO LS[CONTROL.STATUS] ;SET UP CONTROL AND STATUS WORD TO
                ;25973                                          ; REPORT END OF SECTION
U 2EE3, 10E0,15 ;25974          MISC [SET.CP.ATTN]              ;ISSUE CPU ATTN TO CONSOLE
                ;25975  WAIT.EOS:
U 2EE4, 8AEE,44 ;25976          JMP [WAIT.EOS]                  ;LOOP HERE WAITING FOR CONSOLE TO
                ;25977                                          ; RESPOND
```

Cross Reference Listing - Field Names and Defined Values

Cross Reference Listing - Macro Names

Cross Reference Listing - Expression Names

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 0000	1775:	1776:	1777:	1778:	1779:	1780:	1781:	1782:
C 0008	1783:	1784:	1785:	1786:	1787:	1788:	1789:	1790:
C 0010	1794:	1795:	1796:	1797:	1798:	1799:	1800:	1801:
C 0018	1802:	1803:	1804:	1805:	1806:	1807:	1808:	1809:
C 0020	1818:	1819:	1820:	1821:	1822:	1823:	1824:	1825:
C 0028	1826:	1827:	1828:	1829:	1830:	1831:	1832:	1833:
C 0030	1834:	1835:	1836:	1837:	1838:	1839:	1840:	1841:
C 0038	1842:	1843:	1844:	1845:	1846:	1847:	1848:	1849:
C 0040	1858:	1859:	1860:	1861:	1862:	1863:	1864:	1865:
C 0048	1866:	1867:	1868:	1869:	1870:	1871:	1872:	1873:
C 0050	1874:	1875:	1876:	1877:	1878:	1879:	1880:	1881:
C 0058	1882:	1883:	1884:	1885:	1886:	1887:	1888:	1889:
C 0060	1897:	1898:	1899:	1900:	1901:	1902:	1903:	1904:
C 0068	1905:	1906:	1907:	1908:	1909:	1910:	1911:	1912:
C 0070	1913:	1914:	1915:	1916:	1917:	1918:	1919:	1920:
C 0078	1921:	1922:	1923:	1924:	1925:	1926:	1927:	1928:
C 0080	1935:	1937:	1939:	1941:	1944:	1946:	1948:	1950:
C 0088	1953:	1955:	1957:	1959:	1962:	1964:	1966:	1968:
C 0090	1971:	1973:	1975:	1977:	1980:	1982:	1984:	1986:
C 0098	1989:	1991:	1993:	1995:	1998:	2000:	2002:	2004:
C 00A0	2007:	2009:	2011:	2013:	2016:	2018:	2020:	2022:
C 00A8	2025:	2027:	2029:	2031:	2034:	2036:	2038:	2040:
C 00B0	2043:	2045:	2047:	2049:	2052:	2054:	2056:	2058:
C 00B8	2061:	2063:	2065:	2067:	2070:	2072:	2074:	2076:
C 00C0	2084:	2085:	2086:	2087:	2088:	2089:	2090:	2091:
C 00C8	2094:	2095:	2096:	2097:	2098:	2099:	2100:	2101:
C 00D0	2104:	2105:	2106:	2107:	2108:	2109:	211C:	2111:
C 00D8	2114:	2115:	2116:	2117:	2118:	2119:	2120:	2121:
C 00E0	2125:	2126:	2127:	2128:	2129:	2130:	2131:	2132:
C 00E8	2134:	2135:	2136:	2137:	2138:	2139:	2140:	2141:
C 00F0	2145:	2146:	2147:	2148:	2149:	2150:	2151:	2152:
C 00F8	2155:	2156:	2157:	2158:	2159:	2160:	2161:	2162:
C 0100	2170:	2171:	2172:	2173:	2175:	2176:	2177:	2178:
C 0108	2180:	2181:	2182:	2183:	2185:	2186:	2187:	2188:
C 0110	2191:	2192:	2193:	2194:	2196:	2197:	2198:	2199:
C 0118	2201:	2202:	2203:	2204:	2206:	2207:	2208:	2209:
C 0120	2212:	2213:	2214:	2215:	2217:	2218:	2219:	2220:
C 0128	2222:	2223:	2224:	2225:	2227:	2228:	2229:	2230:
C 0130	2233:	2234:	2235:	2236:	2238:	2239:	2240:	2241:
C 0138	2243:	2244:	2245:	2246:	2248:	2249:	2250:	2251:
C 0140	2254:	2255:	2256:	2257:	2259:	2260:	2261:	2262:
C 0148	2264:	2265:	2266:	2267:	2269:	2270:	2271:	2272:
C 0150	2275:	2276:	2277:	2278:	2280:	2281:	2282:	2283:
C 0158	2285:	2286:	2287:	2288:	2290:	2291:	2292:	2293:
C 0160	2296:	2297:	2298:	2299:	2301:	2302:	2303:	2304:
C 0168	2306:	2307:	2308:	2309:	2311:	2312:	2313:	2314:
C 0170	2317:	2318:	2319:	2320:	2322:	2323:	2324:	2325:
C 0178	2327:	2328:	2329:	2330:	2332:	2333:	2334:	2335:
C 0180	2345:	2346:	2347:	2348:	2350:	2351:	2352:	2353:
C 0188	2355:	2356:	2357:	2358:	2360:	2361:	2362:	2363:
C 0190	2366:	2367:	2368:	2369:	2371:	2372:	2373:	2374:
C 0198	2376:	2377:	2378:	2379:	2381:	2382:	2383:	2384:
C 01A0	2387:	2388:	2389:	2390:	2392:	2393:	2394:	2395:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
C 01A8	2397:	2398:	2399:	2400:	2402:	2403:	2404:	2405:
C 01B0	2408:	2409:	2410:	2411:	2413:	2414:	2415:	2416:
C 01B8	2418:	2419:	2420:	2421:	2423:	2424:	2425:	2426:
C 01C0	2429:	2430:	2431:	2432:	2434:	2435:	2436:	2437:
C 01C8	2439:	2440:	2441:	2442:	2444:	2445:	2446:	2447:
C 01D0	2450:	2451:	2452:	2453:	2455:	2456:	2457:	2458:
C 01D8	2460:	2461:	2462:	2463:	2465:	2466:	2467:	2468:
C 01E0	2471:	2472:	2473:	2474:	2476:	2477:	2478:	2479:
C 01E8	2481:	2482:	2483:	2484:	2486:	2487:	2488:	2489:
C 01F0	2492:	2493:	2494:	2495:	2497:	2498:	2499:	2500:
C 01F8	2502:	2503:	2504:	2505:	2507:	2508:	2509:	2510:

;
; Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0000	4085:	3975:	3976:	3977:	3978:	3979:	3980:	3981:
U 0008	3982:	3983:	3984:	3985:	3986:	3987:	3988:	3989:
U 0010	3990:	3991:	3992:	3993:	3994:	3995:	3996:	3997:
U 0018	3998:	3999:	4000:	4001:	4002:	4003:	4004:	4005:
U 0020	4006:	4007:	4008:	4009:	4010:	4011:	4012:	4013:
U 0028	4014:	4015:	4016:	4017:	4018:	4019:	4020:	4021:
U 0030	4022:	4023:	4024:	4025:	4026:	4027:	4028:	4029:
U 0038	4030:	4031:	4032:	4033:	4034:	4035:	4036:	4037:
U 0040	4038:	4039:	4040:	4041:	4042:	4043:	4044:	4046:
U 0048	4048:	4050:	4052:	4054:	4056:	4058:	4060:	4062:
U 0050	4094:	4096:	4098:	4101:	4104:	4108:	4110:	4112:
U 0058	4114:	4116:	4118:	4120:	4122:	4124:	4126:	4128:
U 0060	4130:	4132:	4134:	4136:	4138:	4140:	4141:	4142:
U 0068	4143:	4144:	4145:	4146:	4147:	4148:	4149:	4150:
U 0070	4180:	4182:	4184:	4185:	4187:	4188:	4190:	4191:
U 0078	4193:	4194:	4196:	4197:	4199:	4200:	4202:	4203:
U 0080	4205:	4206:	4208:	4209:	4211:	4212:	4214:	4215:
U 0088	4217:	4218:	4220:	4221:	4223:	4224:	4226:	4227:
U 0090	4229:	4230:	4232:	4233:	4235:	4236:	4238:	4239:
U 0098	4241:	4242:	4244:	4245:	4247:	4248:	4250:	4251:
U 00A0	4253:	4254:	4256:	4257:	4259:	4260:	4262:	4263:
U 00A8	4265:	4266:	4268:	4269:	4271:	4272:	4274:	4275:
U 00B0	4277:	4278:	4280:	4281:	4283:	4284:	4286:	4287:
U 00B8	4289:	4290:	4292:	4293:	4295:	4296:	4298:	4299:
U 00C0	4301:	4302:	4304:	4305:	4307:	4308:	4310:	4311:
U 00C8	4313:	4314:	4316:	4317:	4319:	4320:	4322:	4323:
U 00D0	4325:	4326:	4328:	4329:	4331:	4332:	4334:	4335:
U 00D8	4337:	4338:	4340:	4341:	4343:	4344:	4346:	4347:
U 00E0	4349:	4350:	4352:	4353:	4355:	4356:	4358:	4359:
U 00E8	4361:	4362:	4364:	4365:	4367:	4368:	4370:	4371:
U 00F0	4373:	4374:	4376:	4377:	4379:	4380:	4382:	4383:
U 00F8	4385:	4386:	4388:	4389:	4391:	4392:	4394:	4395:
U 0100	4397:	4398:	4400:	4401:	4403:	4404:	4406:	4407:
U 0108	4409:	4410:	4412:	4413:	4415:	4416:	4418:	4419:
U 0110	4421:	4422:	4424:	4425:	4427:	4428:	4430:	4431:
U 0118	4433:	4434:	4436:	4437:	4439:	4440:	4442:	4443:
U 0120	4445:	4446:	4448:	4449:	4451:	4452:	4454:	4455:
U 0128	4457:	4458:	4460:	4461:	4463:	4464:	4466:	4467:
U 0130	4469:	4470:	4472:	4473:	4475:	4476:	4478:	4479:
U 0138	4481:	4482:	4484:	4485:	4487:	4488:	4490:	4491:
U 0140	4493:	4494:	4496:	4497:	4499:	4500:	4502:	4503:
U 0148	4505:	4506:	4508:	4509:	4511:	4512:	4514:	4515:
U 0150	4517:	4518:	4520:	4521:	4523:	4524:	4526:	4527:
U 0158	4529:	4530:	4532:	4533:	4535:	4536:	4538:	4539:
U 0160	4541:	4542:	4551:	4553:	4555:	4556:	4558:	4559:
U 0168	4561:	4562:	4564:	4565:	4567:	4568:	4570:	4571:
U 0170	4573:	4574:	4576:	4577:	4579:	4580:	4582:	4583:
U 0178	4585:	4586:	4588:	4589:	4591:	4592:	4594:	4595:
U 0180	4597:	4598:	4600:	4601:	4603:	4604:	4606:	4607:
U 0188	4609:	4610:	4612:	4613:	4615:	4616:	4618:	4619:
U 0190	4621:	4622:	4624:	4625:	4627:	4628:	4630:	4631:
U 0198	4633:	4634:	4636:	4637:	4639:	4640:	4642:	4643:
U 01A0	4645:	4646:	4648:	4649:	4651:	4652:	4654:	4655:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 01A8	4657:	4658:	4660:	4661:	4663:	4664:	4666:	4667:
U 01B0	4669:	4670:	4672:	4673:	4675:	4676:	4678:	4679:
U 01B8	4681:	4682:	4684:	4685:	4687:	4688:	4690:	4691:
U 01C0	4693:	4694:	4696:	4697:	4699:	4700:	4702:	4703:
U 01C8	4705:	4706:	4708:	4709:	4711:	4712:	4714:	4715:
U 01D0	4717:	4718:	4720:	4721:	4723:	4724:	4726:	4727:
U 01D8	4729:	4730:	4732:	4733:	4735:	4736:	4738:	4739:
U 01E0	4741:	4742:	4744:	4745:	4747:	4748:	4750:	4751:
U 01E8	4753:	4754:	4756:	4757:	4759:	4760:	4762:	4763:
U 01F0	4765:	4766:	4768:	4769:	4771:	4772:	4774:	4775:
U 01F8	4777:	4778:	4780:	4781:	4783:	4784:	4786:	4787:
U 0200	4789:	4790:	4792:	4793:	4795:	4796:	4798:	4799:
U 0208	4801:	4802:	4804:	4805:	4807:	4808:	4810:	4811:
U 0210	4813:	4814:	4816:	4817:	4819:	4820:	4822:	4823:
U 0218	4825:	4826:	4828:	4829:	4831:	4832:	4834:	4835:
U 0220	4837:	4838:	4840:	4841:	4843:	4844:	4846:	4847:
U 0228	4849:	4850:	4852:	4853:	4855:	4856:	4858:	4859:
U 0230	4861:	4862:	4864:	4865:	4867:	4868:	4870:	4871:
U 0238	4873:	4874:	4876:	4877:	4879:	4880:	4882:	4883:
U 0240	4885:	4886:	4888:	4889:	4891:	4892:	4894:	4895:
U 0248	4897:	4898:	4900:	4901:	4903:	4904:	4906:	4907:
U 0250	4909:	4910:	4912:	4913:	4932:	4935:	4938:	4939:
U 0258	4941:	4942:	4944:	4945:	4947:	4948:	4950:	4951:
U 0260	4953:	4954:	4956:	4957:	4959:	4960:	4962:	4963:
U 0268	4965:	4966:	4968:	4969:	4971:	4972:	4974:	4975:
U 0270	4977:	4978:	4980:	4981:	4983:	4984:	4986:	4987:
U 0278	4989:	4990:	4992:	4993:	4995:	4996:	4998:	4999:
U 0280	5001:	5002:	5004:	5005:	5007:	5008:	5010:	5011:
U 0288	5013:	5014:	5016:	5017:	5019:	5020:	5022:	5023:
U 0290	5025:	5026:	5028:	5029:	5031:	5032:	5034:	5035:
U 0298	5037:	5038:	5040:	5041:	5043:	5044:	5046:	5047:
U 02A0	5049:	5050:	5052:	5053:	5055:	5056:	5058:	5059:
U 02A8	5061:	5062:	5064:	5065:	5067:	5068:	5070:	5071:
U 02B0	5073:	5074:	5076:	5077:	5079:	5080:	5082:	5083:
U 02B8	5085:	5086:	5088:	5089:	5091:	5092:	5094:	5095:
U 02C0	5097:	5098:	5100:	5101:	5103:	5104:	5106:	5107:
U 02C8	5109:	5110:	5112:	5113:	5115:	5116:	5118:	5119:
U 02D0	5121:	5122:	5124:	5125:	5127:	5128:	5130:	5131:
U 02D8	5133:	5134:	5136:	5137:	5139:	5140:	5142:	5143:
U 02E0	5145:	5146:	5148:	5149:	5151:	5152:	5154:	5155:
U 02E8	5157:	5158:	5160:	5161:	5163:	5164:	5166:	5167:
U 02F0	5169:	5170:	5172:	5173:	5175:	5176:	5178:	5179:
U 02F8	5181:	5182:	5184:	5185:	5187:	5188:	5190:	5191:
U 0300	5193:	5194:	5196:	5197:	5199:	5200:	5202:	5203:
U 0308	5205:	5206:	5208:	5209:	5211:	5212:	5214:	5215:
U 0310	5217:	5218:	5220:	5221:	5223:	5224:	5226:	5227:
U 0318	5229:	5230:	5232:	5233:	5235:	5236:	5238:	5239:
U 0320	5241:	5242:	5244:	5245:	5247:	5248:	5250:	5251:
U 0328	5253:	5254:	5256:	5257:	5259:	5260:	5262:	5263:
U 0330	5265:	5266:	5268:	5269:	5271:	5272:	5274:	5275:
U 0338	5277:	5278:	5280:	5281:	5283:	5284:	5286:	5287:
U 0340	5289:	5290:	5292:	5293:	5295:	5296:	5298:	5299:
U 0348	5301:	5302:	5304:	5305:	5307:	5308:	5310:	5311:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0350	5313:	5314:	5316:	5317:	5319:	5320:	5322:	5323:
U 0358	5325:	5326:	5328:	5329:	5331:	5332:	5334:	5335:
U 0360	5337:	5338:	5340:	5341:	5343:	5344:	5346:	5347:
U 0368	5349:	5350:	5352:	5353:	5355:	5356:	5358:	5359:
U 0370	5361:	5362:	5364:	5365:	5367:	5368:	5370:	5371:
U 0378	5373:	5374:	5376:	5377:	5379:	5380:	5382:	5383:
U 0380	5385:	5386:	5388:	5389:	5391:	5392:	5394:	5395:
U 0388	5397:	5398:	5400:	5401:	5403:	5404:	5406:	5407:
U 0390	5409:	5410:	5412:	5413:	5415:	5416:	5418:	5419:
U 0398	5421:	5422:	5424:	5425:	5427:	5428:	5430:	5431:
U 03A0	5433:	5434:	5436:	5437:	5439:	5440:	5442:	5443:
U 03A8	5445:	5446:	5448:	5449:	5451:	5452:	5454:	5455:
U 03B0	5457:	5458:	5460:	5461:	5463:	5464:	5466:	5467:
U 03B8	5469:	5470:	5472:	5473:	5475:	5476:	5478:	5479:
U 03C0	5481:	5482:	5484:	5485:	5487:	5488:	5490:	5491:
U 03C8	5493:	5494:	5496:	5497:	5499:	5500:	5502:	5503:
U 03D0	5505:	5506:	5508:	5509:	5511:	5512:	5514:	5515:
U 03D8	5517:	5518:	5520:	5521:	5523:	5524:	5526:	5527:
U 03E0	5529:	5530:	5532:	5533:	5535:	5536:	5538:	5539:
U 03E8	5541:	5542:	5544:	5545:	5547:	5548:	5550:	5551:
U 03F0	5553:	5554:	5556:	5557:	5559:	5560:	5562:	5563:
U 03F8	5565:	5566:	5568:	5569:	5571:	5572:	5574:	5575:
U 0400	5577:	5578:	5580:	5581:	5583:	5584:	5586:	5587:
U 0408	5589:	5590:	5592:	5593:	5595:	5596:	5598:	5599:
U 0410	5601:	5602:	5604:	5605:	5607:	5608:	5610:	5611:
U 0418	5613:	5614:	5616:	5617:	5619:	5620:	5622:	5623:
U 0420	5625:	5626:	5628:	5629:	5631:	5632:	5634:	5635:
U 0428	5637:	5638:	5640:	5641:	5643:	5644:	5646:	5647:
U 0430	5649:	5650:	5652:	5653:	5655:	5656:	5658:	5659:
U 0438	5661:	5662:	5664:	5665:	5667:	5668:	5670:	5671:
U 0440	5673:	5674:	5676:	5677:	5679:	5680:	5682:	5683:
U 0448	5685:	5686:	5688:	5689:	5691:	5692:	5694:	5695:
U 0450	5697:	5698:	5700:	5701:	5703:	5704:	5706:	5707:
U 0458	5709:	5710:	5712:	5713:	5715:	5716:	5718:	5719:
U 0460	5721:	5722:	5724:	5725:	5727:	5728:	5730:	5731:
U 0468	5733:	5734:	5736:	5737:	5739:	5740:	5742:	5743:
U 0470	5745:	5746:	5748:	5749:	5751:	5752:	5754:	5755:
U 0478	5757:	5758:	5760:	5761:	5763:	5764:	5766:	5767:
U 0480	5769:	5770:	5772:	5773:	5775:	5776:	5778:	5779:
U 0488	5781:	5782:	5784:	5785:	5787:	5788:	5790:	5791:
U 0490	5793:	5794:	5796:	5797:	5799:	5800:	5802:	5803:
U 0498	5805:	5806:	5808:	5809:	5811:	5812:	5814:	5815:
U 04A0	5817:	5818:	5820:	5821:	5823:	5824:	5826:	5827:
U 04A8	5829:	5830:	5832:	5833:	5835:	5836:	5838:	5839:
U 04B0	5841:	5842:	5844:	5845:	5847:	5848:	5850:	5851:
U 04B8	5853:	5854:	5856:	5857:	5859:	5860:	5862:	5863:
U 04C0	5865:	5866:	5868:	5869:	5871:	5872:	5874:	5875:
U 04C8	5877:	5878:	5880:	5881:	5883:	5884:	5886:	5887:
U 04D0	5889:	5890:	5892:	5893:	5895:	5896:	5898:	5899:
U 04D8	5901:	5902:	5904:	5905:	5907:	5908:	5910:	5911:
U 04E0	5913:	5914:	5916:	5917:	5919:	5920:	5922:	5923:
U 04E8	5925:	5926:	5928:	5929:	5931:	5932:	5934:	5935:
U 04F0	5937:	5938:	5940:	5941:	5943:	5944:	5946:	5947:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 04F8	5949:	5950:	5952:	5953:	5955:	5956:	5958:	5959:
U 0500	5961:	5962:	5964:	5965:	5967:	5968:		
U 0508 - 05FF Unused								
U 0600	6679:	6680:	6682:	6685:	6686:	6687:	6701:	6702:
U 0608	6763:	6765:	6767:	6769:	6770:	6825:	6826:	6828:
U 0610	6830:	6832:	6835:	6837:	6838:	6839:	6843:	6844:
U 0618	6845:	6849:	6850:	6851:	6853:	6856:	6857:	6912:
U 0620	6914:	6915:	6918:	6920:	6924:	6926:	6930:	6936:
U 0628	6938:	6939:	6945:	6947:	6949:	6950:	6951:	6957:
U 0630	6958:	6959:	6960:	6961:	6962:	6963:	6966:	6968:
U 0638	6969:	6973:	7023:	7025:	7027:	7029:	7032:	7033:
U 0640	7035:	7036:	7091:	7093:	7094:	7097:	7099:	7103:
U 0648	7105:	7109:	7115:	7117:	7118:	7124:	7125:	7126:
U 0650	7127:	7128:	7129:	7132:	7182:	7184:	7187:	7189:
U 0658	7192:	7193:	7195:	7196:	7244:	7249:	7251:	7253:
U 0660	7254:	7255:	7305:	7307:	7310:	7314:	7316:	7317:
U 0668	7356:	7357:	7358:	7361:	7362:	7363:	7366:	7367:
U 0670	7368:	7371:	7372:	7373:	7376:	7377:	7381:	7382:
U 0678	7383:	7422:	7423:	7424:	7428:	7429:	7430:	7434:
U 0680	7435:	7436:	7440:	7441:	7445:	7446:	7450:	7453:
U 0688	7455:	7502:	7503:	7504:	7505:	7506:	7553:	7554:
U 0690	7555:	7556:	7557:	7625:	7627:	7630:	7631:	7633:
U 0698	7635:	7636:	7637:	7639:	7643:	7645:	7647:	7651:
U 06A0	7652:	7655:	7656:	7658:	7660:	7661:	7663:	7665:
U 06A8	7670:	7671:	7673:	7675:	7676:	7682:	7683:	7686:
U 06B0	7687:	7688:	7693:	7694:	7696:	7697:	7699:	7701:
U 06B8	7702:	7704:	7706:	7708:	7710:	7712:	7719:	7720:
U 06C0	7722:	7725:	7737:	7738:	7739:	7740:	7741:	7742:
U 06C8	7743:	7744:	7745:	7746:	7747:	7751:	7754:	7755:
U 06D0	7756:	7757:	7758:	7759:	7760:	7761:	7762:	7763:
U 06D8	7764:	7765:	7766:	7767:	7768:	7769:	7772:	7776:
U 06E0	7778:	7779:	7780:	7781:	7785:	7787:	7788:	7789:
U 06E8	7790:	7794:	7796:	7797:	7800:	7801:	7802:	7803:
U 06F0	7805:	7806:	7807:	7808:	7810:	7812:	7813:	7814:
U 06F8	7815:	7816:	7817:	7818:	7819:	7820:	7822:	
U 0700	7833:	7835:	7836:	7839:	7841:	7843:	7844:	7845:
U 0708	7847:	7849:	7851:	7852:	7853:	7855:	7857:	7859:
U 0710	7860:	7861:	7863:	7865:	7867:	7868:	7869:	7879:
U 0718	7881:	7882:	7883:	7884:	7885:	7886:	7888:	7889:
U 0720	7890:	7891:	7892:	7893:	7895:	7896:	7897:	7898:
U 0728	7899:	7900:	7902:	7903:	7904:	7905:	7906:	7907:
U 0730	7909:	7910:	7911:	7912:	7913:	7914:	7915:	7917:
U 0738	7918:	7919:	7920:	7921:	7922:	7923:	7924:	7925:
U 0740	7926:	7927:	7928:	7938:	7939:	7940:	7948:	7950:
U 0748	7951:	7952:	7953:	7954:	7957:	7958:	7959:	7960:
U 0750	7961:	7964:	7965:	7966:	7967:	7968:	7970:	7971:
U 0758	7972:	7980:	7981:	7982:	7984:	7985:	7986:	7988:
U 0760	7989:	7991:	7994:	7995:	7996:	7998:	7999:	8000:
U 0768	8002:	8003:	8005:	8008:	8009:	8010:	8012:	8013:
U 0770	8014:	8016:	8017:	8019:	8022:	8023:	8024:	8026:
U 0778	8027:	8028:	8030:	8031:	8033:	8041:	8042:	8043:
U 0780	8044:	8045:	8048:	8049:	8050:	8051:	8054:	8055:
U 0788	8056:	8057:	8060:	8061:	8062:	8063:	8064:	8075:

;
; Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0790	8076:	8078:	8079:	8080:	8081:	8084:	8085:	8086:
U 0798	8087:	8089:	8093:	8094:	8096:	8097:	8098:	8099:
U 07A0	8101:	8104:	8105:	8106:	8108:	8109:	8110:	8112:
U 07A8	8113:	8114:	8116:	8120:	8121:	8122:	8124:	8125:
U 07B0	8126:	8128:	8129:	8130:	8132:	8136:	8137:	8138:
U 07B8	8139:	8140:	8142:	8143:	8144:	8146:	8147:	8148:
U 07C0	8150:	8158:	8159:	8160:	8161:	8162:	8173:	8174:
U 07C8	8175:	8176:	8177:	8178:	8179:	8180:	8181:	8182:
U 07D0	8183:	8184:	8185:	8186:	8187:	8188:	8189:	8190:
U 07D8	8191:	8192:	8193:	8194:	8198:	8199:	8200:	8201:
U 07E0	8204:	8207:	8208:	8209:	8210:	8212:	8219:	8220:
U 07E8	8221:	8222:	8223:	8224:	8225:	8226:	8229:	8230:
U 07F0	8231:	8232:	8233:	8234:	8235:	8236:	8237:	8292:
U 07F8	8294:	8297:	8299:	8301:	8303:	8304:	8305:	8306:
U 0800	8307:	8309:	8311:	8313:	8314:	8315:	8316:	8317:
U 0808	8318:	8319:	8321:	8322:	8323:	8324:	8325:	8326:
U 0810	8327:	8329:	8330:	8331:	8332:	8333:	8334:	8335:
U 0818	8336:	8337:	8339:	8340:	8342:	8343:	8344:	8345:
U 0820	8346:	8394:	8396:	8399:	8401:	8403:	8405:	8406:
U 0828	8408:	8409:	8410:	8411:	8412:	8413:	8414:	8415:
U 0830	8416:	8417:	8418:	8419:	8420:	8421:	8422:	8423:
U 0838	8424:	8425:	8426:	8428:	8429:	8430:	8431:	8527:
U 0840	8529:	8532:	8534:	8536:	8538:	8539:	8540:	8541:
U 0848	8542:	8544:	8545:	8546:	8547:	8548:	8550:	8552:
U 0850	8553:	8554:	8555:	8556:	8557:	8558:	8561:	8562:
U 0858	8563:	8565:	8566:	8567:	8569:	8570:	8571:	8572:
U 0860	8573:	8574:	8575:	8576:	8577:	8579:	8581:	8582:
U 0868	8583:	8584:	8585:	8586:	8587:	8588:	8589:	8590:
U 0870	8591:	8592:	8593:	8595:	8597:	8598:	8599:	8600:
U 0878	8601:	8602:	8603:	8604:	8605:	8606:	8607:	8609:
U 0880	8611:	8612:	8614:	8615:	8617:	8619:	8620:	8621:
U 0888	8622:	8623:	8624:	8626:	8627:	8731:	8733:	8736:
U 0890	8738:	8740:	8742:	8743:	8744:	8746:	8747:	8749:
U 0898	8750:	8751:	8752:	8753:	8754:	8755:	8756:	8758:
U 08A0	8759:	8760:	8761:	8763:	8764:	8765:	8766:	8767:
U 08A8	8768:	8770:	8772:	8773:	8774:	8775:	8776:	8777:
U 08B0	8838:	8840:	8843:	8845:	8847:	8849:	8850:	8851:
U 08B8	8852:	8853:	8854:	8855:	8856:	8857:	8858:	8859:
U 08C0	8860:	8861:	8863:	8864:	8865:	8866:	8867:	8868:
U 08C8	8870:	8871:	8872:	8873:	8874:	8875:	8877:	8878:
U 08D0	8879:	8880:	8881:	8882:	8884:	8885:	8886:	8887:
U 08D8	8888:	8890:	8891:	8892:	8893:	8894:	8895:	8896:
U 08E0	8897:	8898:	8900:	8901:	8902:	8903:	8904:	8906:
U 08E8	8907:	8908:	8909:	8910:	8911:	8912:	8913:	8915:
U 08F0	8916:	8917:	8918:	8919:	8920:	8921:	8922:	8923:
U 08F8	8924:	8925:	8926:	8927:	8928:	8929:	8930:	8931:
U 0900	8933:	8934:	8935:	8936:	8937:	8939:	8940:	8942:
U 0908	8943:	8944:	8945:	8946:	8948:	8949:	8998:	9000:
U 0910	9003:	9005:	9007:	9009:	9010:	9011:	9012:	9013:
U 0918	9014:	9015:	9016:	9017:	9018:	9019:	9020:	9021:
U 0920	9022:	9023:	9024:	9026:	9084:	9086:	9089:	9091:
U 0928	9093:	9095:	9096:	9097:	9099:	9100:	9101:	9102:
U 0930	9103:	9104:	9106:	9107:	9108:	9109:	9113:	9114:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0938	9115:	9118:	9119:	9120:	9121:	9123:	9124:	9126:
U 0940	9127:	9129:	9130:	9183:	9185:	9188:	9190:	9192:
U 0948	9194:	9195:	9196:	9197:	9198:	9199:	9200:	9201:
U 0950	9202:	9203:	9204:	9205:	9206:	9207:	9208:	9209:
U 0958	9210:	9211:	9212:	9213:	9214:	9215:	9216:	9217:
U 0960	9218:	9219:	9220:	9221:	9222:	9311:	9313:	9316:
U 0968	9318:	9320:	9322:	9323:	9324:	9325:	9326:	9327:
U 0970	9328:	9329:	9330:	9331:	9332:	9334:	9335:	9336:
U 0978	9337:	9338:	9339:	9340:	9341:	9342:	9343:	9344:
U 0980	9345:	9346:	9347:	9348:	9349:	9350:	9351:	9352:
U 0988	9353:	9503:	9505:	9508:	9510:	9512:	9514:	9515:
U 0990	9516:	9517:	9518:	9519:	9520:	9521:	9522:	9523:
U 0998	9524:	9525:	9526:	9528:	9529:	9530:	9531:	9532:
U 09A0	9534:	9535:	9536:	9537:	9538:	9539:	9540:	9541:
U 09A8	9542:	9543:	9544:	9545:	9546:	9547:	9548:	9549:
U 09B0	9550:	9551:	9552:	9553:	9554:	9555:	9556:	9557:
U 09B8	9559:	9560:	9563:	9564:	9565:	9566:	9567:	9568:
U 09C0	9569:	9570:	9572:	9573:	9574:	9575:	9576:	9578:
U 09C8	9579:	9580:	9581:	9582:	9583:	9584:	9585:	9586:
U 09D0	9587:	9588:	9589:	9590:	9591:	9592:	9593:	9594:
U 09D8	9595:	9596:	9597:	9598:	9599:	9600:	9601:	9602:
U 09E0	9603:	9604:	9605:	9607:	9608:	9610:	9612:	9613:
U 09E8	9614:	9615:	9616:	9617:	9618:	9619:	9620:	9621:
U 09F0	9622:	9623:	9625:	9626:	9627:	9628:	9629:	9631:
U 09F8	9632:	9633:	9634:	9635:	9636:	9637:	9638:	9639:
U 0A00	9640:	9641:	9642:	9643:	9644:	9645:	9646:	9647:
U 0A08	9648:	9649:	9650:	9651:	9652:	9653:	9654:	9655:
U 0A10	9656:	9657:	9658:	9659:	9660:	9662:	9663:	9750:
U 0A18	9752:	9755:	9757:	9759:	9761:	9762:	9763:	9764:
U 0A20	9765:	9766:	9767:	9768:	9769:	9770:	9771:	9772:
U 0A28	9773:	9774:	9775:	9776:	9777:	9778:	9779:	9780:
U 0A30	9781:	9782:	9783:	9784:	9785:	9786:	9787:	9788:
U 0A38	9789:	9790:	9791:	9793:	9794:	9795:	9796:	9798:
U 0A40	9800:	9801:	9802:	9803:	9804:	9805:	9806:	9807:
U 0A48	9809:	9810:	9813:	9814:	9816:	9818:	9819:	9820:
U 0A50	9821:	9822:	9823:	9824:	9825:	9827:	9828:	9831:
U 0A58	9832:	9833:	9834:	9835:	9836:	9837:	9838:	9839:
U 0A60	9840:	9842:	9844:	9845:	9846:	9847:	9848:	9849:
U 0A68	9850:	9851:	9852:	9853:	9854:	9855:	9857:	9858:
U 0A70	9861:	9862:	9864:	9866:	9867:	9868:	9869:	9870:
U 0A78	9871:	9872:	9873:	9874:	9875:	9876:	9877:	9879:
U 0A80	9880:	9883:	9885:	9886:	9887:	9888:	9889:	9890:
U 0A88	9891:	9892:	9893:	9894:	9895:	9896:	9897:	9898:
U 0A90	9900:	9902:	9903:	9904:	9905:	9906:	9907:	9908:
U 0A98	9909:	9910:	9911:	9912:	9913:	9914:	9915:	9916:
U 0AA0	9917:	9919:	9920:	9923:	9924:	9926:	9928:	9929:
U 0AA8	9930:	9931:	9932:	9933:	9934:	9935:	9936:	9937:
U 0AB0	9938:	9939:	9940:	9941:	9942:	9943:	9945:	9946:
U 0AB8	10018:	10020:	10023:	10025:	10027:	10029:	10030:	10031:
U 0AC0	10032:	10033:	10034:	10035:	10036:	10037:	10038:	10040:
U 0AC8	10042:	10043:	10044:	10045:	10046:	10047:	10048:	10050:
U 0AD0	10051:	10054:	10055:	10056:	10057:	10058:	10059:	10060:
U 0AD8	10061:	10062:	10064:	10066:	10067:	10069:	10070:	10071:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0AE0	10072:	10073:	10074:	10076:	10077:	10079:	10081:	10082:
U 0AE8	10083:	10084:	10085:	10086:	10087:	10088:	10089:	10090:
U 0AF0	10091:	10093:	10095:	10096:	10098:	10099:	10100:	10101:
U 0AF8	10102:	10103:	10104:	10105:	10106:	10107:	10109:	10110:
U 0B00	10111:	10120:	10121:	10122:	10123:	10124:	10125:	10126:
U 0B08	10127:	10128:	10130:	10131:	10132:	10133:	10134:	10203:
U 0B10	10205:	10208:	10210:	10212:	10214:	10215:	10216:	10217:
U 0B18	10218:	10219:	10220:	10221:	10222:	10223:	10224:	10225:
U 0B20	10226:	10227:	10228:	10229:	10231:	10233:	10234:	10236:
U 0B28	10237:	10238:	10239:	10240:	10241:	10242:	10243:	10244:
U 0B30	10245:	10246:	10247:	10248:	10250:	10251:	10253:	10254:
U 0B38	10256:	10257:	10364:	10366:	10369:	10371:	10373:	10375:
U 0B40	10376:	10377:	10378:	10379:	10380:	10381:	10382:	10383:
U 0B48	10384:	10385:	10386:	10388:	10389:	10390:	10391:	10392:
U 0B50	10394:	10396:	10397:	10398:	10399:	10400:	10401:	10402:
U 0B58	10403:	10404:	10405:	10406:	10407:	10408:	10409:	10410:
U 0B60	10411:	10413:	10414:	10417:	10418:	10420:	10422:	10423:
U 0B68	10424:	10425:	10426:	10427:	10428:	10429:	10430:	10431:
U 0B70	10432:	10433:	10434:	10435:	10436:	10437:	10439:	10440:
U 0B78	10443:	10445:	10446:	10447:	10526:	10528:	10531:	10533:
U 0B80	10535:	10537:	10538:	10539:	10540:	10541:	10542:	10543:
U 0B88	10544:	10545:	10546:	10547:	10549:	10550:	10551:	10552:
U 0B90	10553:	10554:	10555:	10556:	10558:	10559:	10560:	10561:
U 0B98	10562:	10563:	10565:	10566:	10567:	10568:	10569:	10570:
U 0BA0	10571:	10572:	10573:	10574:	10575:	10576:	10577:	10578:
U 0BA8	10579:	10580:	10581:	10582:	10583:	10584:	10585:	10586:
U 0BB0	10587:	10588:	10589:	10590:	10591:	10592:	10593:	10594:
U 0BB8	10595:	10596:	10597:	10598:	10599:	10600:	10601:	10602:
U 0BC0	10603:	10604:	10605:	10606:	10607:	10608:	10609:	10610:
U 0BC8	10611:	10613:	10614:	10615:	10616:	10617:	10618:	10619:
U 0BD0	10620:	10621:	10622:	10624:	10625:	10626:	10627:	10628:
U 0BD8	10629:	10630:	10631:	10632:	10633:	10634:	10635:	10636:
U 0BE0	10637:	10638:	10639:	10640:	10641:	10642:	10643:	10644:
U 0BE8	10645:	10646:	10647:	10648:	10649:	10650:	10651:	10652:
U 0BF0	10653:	10654:	10655:	10656:	10657:	10658:	10659:	10660:
U 0BF8	10661:	10662:	10663:	10664:	10665:	10666:	10667:	10668:
U 0C00	10669:	10670:	10672:	10673:	10748:	10750:	10753:	10755:
U 0C08	10757:	10759:	10760:	10761:	10762:	10763:	10764:	10765:
U 0C10	10766:	10767:	10768:	10769:	10770:	10771:	10772:	10773:
U 0C18	10774:	10775:	10776:	10777:	10778:	10779:	10780:	10781:
U 0C20	10782:	10783:	10784:	10786:	10787:	10788:	10789:	10790:
U 0C28	10791:	10792:	10793:	10794:	10795:	10796:	10797:	10798:
U 0C30	10799:	10801:	10802:	10803:	10804:	10805:	10806:	10807:
U 0C38	10808:	10809:	10810:	10811:	10812:	10813:	10814:	10815:
U 0C40	10816:	10817:	10818:	10819:	10820:	10821:	10822:	10823:
U 0C48	10824:	10825:	10826:	10827:	10828:	10829:	10830:	10831:
U 0C50	10832:	10833:	10834:	10835:	10836:	10837:	10838:	10839:
U 0C58	10840:	10841:	10842:	10843:	10844:	10845:	10846:	10847:
U 0C60	10848:	10849:	10850:	10851:	10852:	10853:	10854:	10855:
U 0C68	10857:	10858:	10859:	10860:	10861:	10862:	10863:	10864:
U 0C70	10865:	10866:	10867:	10868:	10869:	10870:	10871:	10872:
U 0C78	10873:	10874:	10875:	10876:	10877:	10878:	10879:	10880:
U 0C80	10881:	10882:	10883:	10884:	10885:	10886:	10887:	10888:

;
; Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0C88	10889:	10890:	10891:	10892:	10893:	10894:	10895:	10896:
U 0C90	10897:	10898:	10899:	10900:	10901:	10902:	10903:	10904:
U 0C98	10905:	10906:	10907:	10908:	10909:	10910:	10911:	10912:
U 0CA0	10913:	10914:	10915:	10916:	10917:	10918:	10919:	10920:
U 0CA8	10921:	10922:	10923:	10924:	10925:	10926:	10927:	10928:
U 0CB0	10929:	10930:	10931:	10932:	10933:	10934:	10935:	10937:
U 0CB8	10938:	10939:	10940:	10941:	10942:	10943:	10944:	10945:
U 0CC0	10946:	10947:	10948:	10949:	10950:	10951:	10952:	10953:
U 0CC8	10954:	10955:	10956:	10957:	10958:	10959:	10960:	10961:
U 0CD0	10962:	10963:	10964:	10965:	10966:	10967:	10968:	10969:
U 0CD8	10970:	10971:	10972:	10973:	10974:	10975:	10976:	10977:
U 0CE0	10978:	10979:	10980:	10981:	10982:	10983:	10984:	10985:
U 0CE8	10986:	10987:	10988:	10989:	10990:	10991:	10993:	10994:
U 0CF0	11070:	11072:	11075:	11077:	11079:	11081:	11082:	11083:
U 0CF8	11084:	11085:	11086:	11087:	11088:	11089:	11090:	11091:
U 0D00	11092:	11093:	11094:	11095:	11096:	11097:	11098:	11099:
U 0D08	11100:	11101:	11103:	11104:	11105:	11106:	11107:	11108:
U 0D10	11109:	11110:	11111:	11113:	11114:	11115:	11116:	11117:
U 0D18	11118:	11119:	11120:	11121:	11122:	11124:	11125:	11126:
U 0D20	11127:	11128:	11129:	11130:	11131:	11132:	11133:	11134:
U 0D28	11135:	11136:	11137:	11138:	11139:	11140:	11141:	11142:
U 0D30	11143:	11144:	11145:	11146:	11147:	11148:	11149:	11150:
U 0D38	11151:	11152:	11153:	11154:	11155:	11156:	11157:	11158:
U 0D40	11159:	11160:	11161:	11162:	11163:	11164:	11165:	11166:
U 0D48	11168:	11169:	11170:	11171:	11172:	11173:	11174:	11175:
U 0D50	11176:	11177:	11178:	11179:	11180:	11181:	11182:	11183:
U 0D58	11184:	11185:	11186:	11187:	11188:	11189:	11190:	11191:
U 0D60	11192:	11193:	11194:	11195:	11196:	11197:	11199:	11200:
U 0D68	11201:	11202:	11203:	11204:	11205:	11206:	11207:	11208:
U 0D70	11209:	11210:	11211:	11213:	11214:	11215:	11216:	11217:
U 0D78	11218:	11219:	11220:	11221:	11222:	11223:	11224:	11225:
U 0D80	11226:	11227:	11228:	11229:	11230:	11231:	11232:	11233:
U 0D88	11234:	11235:	11236:	11237:	11238:	11239:	11240:	11241:
U 0D90	11242:	11244:	11245:	11246:	11247:	11248:	11249:	11250:
U 0D98	11251:	11252:	11253:	11254:	11255:	11256:	11257:	11258:
U 0DA0	11259:	11260:	11261:	11262:	11263:	11264:	11265:	11266:
U 0DA8	11267:	11268:	11269:	11270:	11271:	11272:	11442:	11444:
U 0DB0	11447:	11449:	11451:	11453:	11454:	11455:	11456:	11457:
U 0DB8	11458:	11459:	11460:	11461:	11462:	11463:	11464:	11465:
U 0DC0	11466:	11467:	11468:	11469:	11470:	11471:	11472:	11473:
U 0DC8	11474:	11475:	11476:	11477:	11478:	11479:	11480:	11481:
U 0DD0	11482:	11483:	11484:	11485:	11486:	11487:	11488:	11489:
U 0DD8	11490:	11491:	11492:	11493:	11494:	11495:	11496:	11497:
U 0DE0	11498:	11499:	11500:	11501:	11502:	11503:	11504:	11505:
U 0DE8	11506:	11507:	11508:	11509:	11510:	11511:	11512:	11513:
U 0DF0	11514:	11515:	11516:	11517:	11518:	11519:	11520:	11521:
U 0DF8	11522:	11523:	11524:	11525:	11526:	11527:	11528:	11529:
U 0E00	11530:	11531:	11532:	11533:	11534:	11535:	11536:	11537:
U 0E08	11538:	11539:	11540:	11541:	11542:	11543:	11544:	11545:
U 0E10	11546:	11547:	11548:	11549:	11551:	11552:	11653:	11655:
U 0E18	11658:	11660:	11662:	11664:	11666:	11667:	11668:	11669:
U 0E20	11670:	11672:	11673:	11676:	11677:	11678:	11680:	11682:
U 0E28	11683:	11685:	11687:	11689:	11690:	11691:	11692:	11693:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0E30	11695:	11697:	11699:	11701:	11702:	11704:	11706:	11707:
U 0E38	11708:	11709:	11711:	11713:	11714:	11715:	11717:	11719:
U 0E40	11721:	11722:	11724:	11726:	11728:	11729:	11730:	11731:
U 0E48	11732:	11733:	11735:	11737:	11738:	11740:	11741:	11743:
U 0E50	11744:	11745:	11746:	11747:	11748:	11750:	11752:	11754:
U 0E58	11755:	11756:	11757:	11758:	11759:	11761:	11763:	11764:
U 0E60	11766:	11767:	11769:	11773:	11774:	11775:	11776:	11777:
U 0E68	11779:	11780:	11782:	11784:	11785:	11786:	11787:	11789:
U 0E70	11790:	11791:	11792:	11793:	11794:	11796:	11797:	11798:
U 0E78	11799:	11800:	11801:	11802:	11912:	11914:	11917:	11919:
U 0E80	11921:	11923:	11925:	11926:	11927:	11928:	11929:	11931:
U 0E88	11932:	11934:	11935:	11938:	11939:	11940:	11942:	11944:
U 0E90	11945:	11947:	11948:	11949:	11950:	11951:	11952:	11954:
U 0E98	11955:	11956:	11957:	11958:	11960:	11961:	11962:	11963:
U 0EA0	11965:	11966:	11967:	11969:	11970:	11971:	11973:	11974:
U 0EA8	11975:	11977:	11978:	11979:	11980:	11982:	11983:	11984:
U 0EB0	11986:	11987:	11988:	11990:	11991:	11992:	11994:	11995:
U 0EB8	11996:	11998:	11999:	12001:	12002:	12003:	12005:	12007:
U 0EC0	12008:	12009:	12010:	12011:	12012:	12014:	12015:	12016:
U 0EC8	12018:	12019:	12021:	12022:	12024:	12025:	12027:	12028:
U 0ED0	12029:	12031:	12033:	12034:	12035:	12036:	12037:	12038:
U 0ED8	12040:	12041:	12042:	12044:	12045:	12047:	12048:	12050:
U 0EE0	12054:	12055:	12056:	12057:	12058:	12059:	12061:	12062:
U 0EE8	12064:	12066:	12067:	12068:	12069:	12071:	12072:	12073:
U 0EF0	12074:	12075:	12076:	12078:	12079:	12080:	12081:	12082:
U 0EF8	12083:	12084:	12194:	12196:	12199:	12201:	12203:	12205:
U 0F00	12207:	12208:	12209:	12210:	12211:	12213:	12214:	12216:
U 0F08	12217:	12220:	12221:	12222:	12224:	12226:	12227:	12229:
U 0F10	12230:	12231:	12232:	12233:	12234:	12236:	12237:	12238:
U 0F18	12239:	12240:	12241:	12243:	12244:	12245:	12246:	12248:
U 0F20	12249:	12250:	12252:	12253:	12255:	12256:	12257:	12259:
U 0F28	12260:	12261:	12262:	12264:	12265:	12266:	12268:	12269:
U 0F30	12270:	12272:	12273:	12274:	12276:	12277:	12278:	12280:
U 0F38	12281:	12283:	12284:	12285:	12287:	12289:	12290:	12291:
U 0F40	12292:	12293:	12294:	12296:	12297:	12298:	12300:	12301:
U 0F48	12303:	12304:	12306:	12307:	12309:	12310:	12311:	12313:
U 0F50	12315:	12316:	12317:	12318:	12319:	12320:	12322:	12323:
U 0F58	12324:	12326:	12327:	12329:	12330:	12332:	12335:	12336:
U 0F60	12337:	12338:	12339:	12340:	12342:	12343:	12345:	12347:
U 0F68	12348:	12349:	12350:	12352:	12353:	12354:	12355:	12356:
U 0F70	12357:	12359:	12360:	12361:	12362:	12363:	12364:	12365:
U 0F78	12460:	12462:	12465:	12467:	12469:	12471:	12473:	12474:
U 0F80	12475:	12476:	12477:	12479:	12480:	12482:	12483:	12484:
U 0F88	12486:	12488:	12489:	12490:	12492:	12494:	12495:	12496:
U 0F90	12497:	12498:	12500:	12512:	12513:	12514:	12515:	12516:
U 0F98	12517:	12518:	12519:	12521:	12523:	12524:	12525:	12527:
U 0FA0	12528:	12529:	12531:	12532:	12534:	12535:	12536:	12537:
U 0FA8	12538:	12539:	12541:	12542:	12543:	12545:	12546:	12547:
U 0FB0	12549:	12550:	12552:	12553:	12555:	12556:	12558:	12559:
U 0FB8	12560:	12561:	12562:	12563:	12565:	12566:	12567:	12569:
U 0FC0	12570:	12571:	12573:	12574:	12576:	12577:	12579:	12583:
U 0FC8	12584:	12585:	12586:	12587:	12588:	12589:	12590:	12592:
U 0FD0	12594:	12595:	12596:	12597:	12599:	12600:	12601:	12602:

;
Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 0FD8	12603:	12604:	12606:	12607:	12608:	12609:	12610:	12611:
U 0FE0	12612:	12615:	12616:	12617:	12618:	12619:	12620:	12621:
U 0FE8	12622:	12624:	12626:	12627:	12628:	12629:	12631:	12632:
U 0FF0	12633:	12634:	12635:	12636:	12638:	12639:	12640:	12641:
U 0FF8	12642:	12643:	12644:	12806:	12808:	12811:	12813:	12815:
U 1000	12817:	12819:	12820:	12821:	12822:	12824:	12825:	12826:
U 1008	12827:	12829:	12831:	12833:	12835:	12836:	12837:	12838:
U 1010	12839:	12840:	12841:	12843:	12845:	12847:	12848:	12849:
U 1018	12850:	12852:	12853:	12854:	12855:	12856:	12858:	12859:
U 1020	12861:	12862:	12863:	12864:	12865:	12866:	12867:	12869:
U 1028	12870:	12871:	12873:	12874:	12875:	12876:	12878:	12879:
U 1030	12881:	12882:	12883:	12884:	12885:	12886:	12888:	12891:
U 1038	12892:	12893:	12894:	12895:	12896:	12898:	12899:	12901:
U 1040	12902:	12904:	12905:	12906:	12907:	12908:	12910:	12911:
U 1048	12912:	12913:	12914:	12915:	12918:	12919:	12920:	12921:
U 1050	12922:	12923:	12924:	12926:	12927:	12929:	12930:	12931:
U 1058	12932:	12933:	12949:	12950:	12951:	12952:	12953:	12954:
U 1060	12956:	12957:	12958:	12959:	12961:	12962:	12964:	12965:
U 1068	12967:	12968:	12970:	12972:	12974:	12976:	12977:	12978:
U 1070	12979:	12980:	12981:	12983:	12984:	12985:	12986:	12987:
U 1078	12989:	12990:	12991:	12992:	12993:	12995:	12996:	12997:
U 1080	12998:	12999:	13001:	13002:	13003:	13004:	13007:	13008:
U 1088	13010:	13011:	13012:	13013:	13016:	13017:	13020:	13021:
U 1090	13024:	13025:	13029:	13030:	13031:	13032:	13034:	13036:
U 1098	13037:	13038:	13040:	13041:	13042:	13043:	13044:	13045:
U 10A0	13047:	13048:	13050:	13051:	13052:	13055:	13056:	13058:
U 10A8	13061:	13062:	13063:	13065:	13066:	13067:	13068:	13069:
U 10B0	13072:	13073:	13074:	13075:	13076:	13078:	13080:	13081:
U 10B8	13082:	13083:	13084:	13085:	13087:	13090:	13091:	13092:
U 10C0	13093:	13094:	13095:	13097:	13098:	13099:	13101:	13103:
U 10C8	13104:	13106:	13108:	13109:	13110:	13111:	13113:	13114:
U 10D0	13115:	13116:	13117:	13118:	13120:	13121:	13122:	13123:
U 10D8	13124:	13125:	13126:	13129:	13130:	13131:	13132:	13133:
U 10E0	13134:	13136:	13138:	13139:	13141:	13143:	13144:	13145:
U 10E8	13146:	13148:	13149:	13150:	13151:	13152:	13153:	13155:
U 10F0	13156:	13157:	13158:	13159:	13160:	13161:	13164:	13165:
U 10F8	13166:	13167:	13168:	13169:	13171:	13172:	13173:	13175:
U 1100	13177:	13178:	13180:	13181:	13182:	13183:	13185:	13186:
U 1108	13188:	13189:	13190:	13191:	13192:	13193:	13196:	13197:
U 1110	13198:	13199:	13201:	13202:	13204:	13205:	13206:	13208:
U 1118	13210:	13211:	13213:	13214:	13215:	13216:	13218:	13219:
U 1120	13221:	13222:	13223:	13224:	13225:	13226:	13383:	13385:
U 1128	13388:	13390:	13392:	13394:	13396:	13397:	13398:	13399:
U 1130	13400:	13401:	13404:	13405:	13406:	13408:	13409:	13410:
U 1138	13411:	13413:	13414:	13416:	13417:	13418:	13419:	13420:
U 1140	13422:	13424:	13426:	13427:	13428:	13429:	13430:	13431:
U 1148	13433:	13434:	13436:	13437:	13438:	13439:	13440:	13443:
U 1150	13444:	13445:	13446:	13447:	13448:	13450:	13451:	13453:
U 1158	13454:	13455:	13456:	13459:	13460:	13462:	13463:	13464:
U 1160	13465:	13466:	13469:	13470:	13471:	13473:	13475:	13476:
U 1168	13477:	13478:	13479:	13480:	13481:	13482:	13483:	13485:
U 1170	13486:	13488:	13489:	13491:	13492:	13493:	13495:	13496:
U 1178	13498:	13500:	13502:	13503:	13504:	13505:	13507:	13508:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1180	13509:	13511:	13512:	13513:	13514:	13517:	13518:	13519:
U 1188	13520:	13521:	13522:	13524:	13525:	13526:	13527:	13528:
U 1190	13531:	13532:	13534:	13535:	13536:	13537:	13538:	13551:
U 1198	13552:	13553:	13554:	13555:	13556:	13558:	13559:	13560:
U 11A0	13561:	13563:	13564:	13566:	13568:	13570:	13572:	13573:
U 11A8	13574:	13575:	13576:	13577:	13579:	13580:	13581:	13582:
U 11B0	13583:	13586:	13587:	13588:	13589:	13590:	13591:	13592:
U 11B8	13593:	13596:	13597:	13598:	13599:	13602:	13603:	13604:
U 11C0	13605:	13608:	13610:	13613:	13614:	13615:	13616:	13617:
U 11C8	13618:	13619:	13620:	13622:	13623:	13625:	13628:	13629:
U 11D0	13630:	13631:	13633:	13634:	13635:	13636:	13637:	13639:
U 11D8	13640:	13641:	13643:	13645:	13646:	13647:	13648:	13649:
U 11E0	13651:	13653:	13654:	13655:	13656:	13658:	13659:	13660:
U 11E8	13661:	13662:	13663:	13665:	13666:	13667:	13668:	13669:
U 11F0	13670:	13671:	13674:	13675:	13676:	13677:	13678:	13679:
U 11F8	13680:	13681:	13683:	13684:	13685:	13686:	13688:	13689:
U 1200	13691:	13692:	13693:	13694:	13695:	13696:	13828:	13830:
U 1208	13833:	13835:	13837:	13839:	13841:	13842:	13843:	13844:
U 1210	13845:	13846:	13847:	13848:	13850:	13851:	13854:	13855:
U 1218	13857:	13858:	13859:	13860:	13861:	13863:	13865:	13867:
U 1220	13868:	13869:	13870:	13872:	13873:	13874:	13876:	13877:
U 1228	13878:	13879:	13882:	13883:	13884:	13885:	13887:	13888:
U 1230	13889:	13890:	13891:	13892:	13895:	13896:	13898:	13899:
U 1238	13900:	13901:	13902:	13905:	13906:	13907:	13909:	13911:
U 1240	13912:	13913:	13914:	13915:	13916:	13917:	13919:	13920:
U 1248	13922:	13923:	13925:	13926:	13927:	13929:	13930:	13931:
U 1250	13932:	13933:	13935:	13936:	13937:	13939:	13940:	13942:
U 1258	13943:	13944:	13945:	13946:	13949:	13950:	13951:	13952:
U 1260	13953:	13954:	13956:	13957:	13958:	13959:	13962:	13963:
U 1268	13964:	13965:	13966:	13967:	13968:	13981:	13982:	13983:
U 1270	13984:	13985:	13986:	13988:	13989:	13990:	13991:	13993:
U 1278	13994:	13996:	13998:	13999:	14000:	14001:	14002:	14003:
U 1280	14005:	14006:	14007:	14008:	14009:	14012:	14013:	14014:
U 1283	14015:	14016:	14017:	14018:	14019:	14022:	14023:	14024:
U 1290	14025:	14026:	14029:	14030:	14031:	14032:	14035:	14037:
U 1298	14040:	14041:	14042:	14043:	14044:	14045:	14046:	14047:
U 12A0	14049:	14050:	14052:	14055:	14056:	14057:	14058:	14059:
U 12A8	14060:	14062:	14064:	14065:	14067:	14069:	14070:	14071:
U 12B0	14072:	14074:	14075:	14076:	14077:	14078:	14079:	14081:
U 12B8	14082:	14083:	14084:	14085:	14086:	14087:	14090:	14091:
U 12C0	14092:	14093:	14094:	14095:	14096:	14097:	14099:	14100:
U 12C8	14101:	14102:	14104:	14105:	14107:	14108:	14109:	14110:
U 12D0	14111:	14112:	14188:	14190:	14193:	14195:	14197:	14199:
U 12D8	14201:	14202:	14203:	14204:	14205:	14207:	14208:	14210:
U 12E0	14211:	14213:	14214:	14215:	14217:	14218:	14221:	14223:
U 12E8	14224:	14225:	14226:	14227:	14228:	14230:	14231:	14232:
U 12F0	14233:	14234:	14235:	14236:	14237:	14239:	14241:	14242:
U 12F8	14243:	14244:	14246:	14247:	14248:	14249:	14250:	14251:
U 1300	14253:	14254:	14255:	14256:	14257:	14258:	14260:	14261:
U 1308	14263:	14264:	14266:	14267:	14268:	14269:	14270:	14271:
U 1310	14272:	14274:	14275:	14276:	14277:	14278:	14279:	14280:
U 1318	14281:	14282:	14283:	14284:	14285:	14286:	14287:	14289:
U 1320	14291:	14292:	14293:	14294:	14296:	14297:	14298:	14299:

;
Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1328	14300:	14301:	14303:	14304:	14305:	14306:	14307:	14308:
U 1330	14383:	14385:	14388:	14390:	14392:	14394:	14396:	14397:
U 1338	14398:	14399:	14400:	14402:	14403:	14405:	14406:	14408:
U 1340	14409:	14410:	14412:	14414:	14416:	14418:	14419:	14420:
U 1348	14421:	14422:	14423:	14425:	14426:	14427:	14428:	14429:
U 1350	14430:	14431:	14432:	14434:	14436:	14437:	14438:	14439:
U 1358	14441:	14442:	14443:	14444:	14445:	14446:	14448:	14449:
U 1360	14450:	14451:	14452:	14453:	14455:	14456:	14458:	14459:
U 1368	14461:	14462:	14463:	14464:	14465:	14466:	14467:	14469:
U 1370	14470:	14471:	14472:	14473:	14474:	14475:	14476:	14477:
U 1378	14478:	14479:	14480:	14481:	14482:	14484:	14486:	14487:
U 1380	14488:	14489:	14491:	14492:	14493:	14494:	14495:	14496:
U 1388	14498:	14499:	14500:	14501:	14502:	14503:	14575:	14577:
U 1390	14580:	14582:	14584:	14586:	14588:	14589:	14590:	14591:
U 1398	14592:	14594:	14595:	14597:	14598:	14600:	14601:	14602:
U 13A0	14604:	14605:	14608:	14610:	14611:	14612:	14613:	14614:
U 13A8	14615:	14617:	14618:	14619:	14620:	14621:	14622:	14623:
U 13B0	14624:	14625:	14626:	14627:	14628:	14629:	14631:	14633:
U 13B8	14634:	14635:	14636:	14638:	14639:	14640:	14641:	14642:
U 13C0	14643:	14645:	14646:	14647:	14648:	14649:	14650:	14652:
U 13C8	14653:	14654:	14655:	14656:	14657:	14658:	14660:	14662:
U 13D0	14663:	14664:	14665:	14667:	14668:	14669:	14670:	14671:
U 13D8	14672:	14674:	14675:	14676:	14677:	14678:	14679:	14681:
U 13E0	14682:	14684:	14685:	14757:	14759:	14762:	14764:	14766:
U 13E8	14768:	14770:	14771:	14772:	14773:	14774:	14776:	14777:
U 13F0	14779:	14780:	14782:	14783:	14784:	14786:	14788:	14790:
U 13F8	14792:	14793:	14794:	14795:	14796:	14797:	14799:	14800:
U 1400	14801:	14802:	14803:	14804:	14805:	14806:	14807:	14808:
U 1408	14809:	14810:	14811:	14813:	14815:	14816:	14817:	14818:
U 1410	14820:	14821:	14822:	14823:	14824:	14825:	14827:	14828:
U 1418	14829:	14830:	14831:	14832:	14834:	14835:	14836:	14837:
U 1420	14838:	14839:	14840:	14842:	14844:	14845:	14846:	14847:
U 1428	14849:	14850:	14851:	14852:	14853:	14854:	14856:	14857:
U 1430	14858:	14859:	14860:	14861:	14863:	14864:	14866:	14867:
U 1438	14958:	14960:	14963:	14965:	14967:	14969:	14971:	14972:
U 1440	14973:	14974:	14975:	14976:	14977:	14978:	14980:	14981:
U 1448	14982:	14983:	14985:	14986:	14988:	14989:	14991:	14992:
U 1450	14994:	14995:	14997:	14998:	15000:	15002:	15003:	15004:
U 1458	15005:	15006:	15007:	15008:	15009:	15011:	15012:	15013:
U 1460	15014:	15015:	15018:	15019:	15020:	15021:	15022:	15023:
U 1468	15024:	15026:	15027:	15028:	15030:	15031:	15032:	15033:
U 1470	15036:	15037:	15038:	15039:	15040:	15041:	15043:	15045:
U 1478	15047:	15048:	15049:	15050:	15051:	15053:	15054:	15056:
U 1480	15058:	15060:	15062:	15063:	15064:	15065:	15066:	15067:
U 1488	15068:	15070:	15071:	15073:	15076:	15077:	15078:	15079:
U 1490	15080:	15081:	15082:	15083:	15084:	15085:	15086:	15087:
U 1498	15088:	15089:	15090:	15091:	15093:	15095:	15096:	15097:
U 14A0	15098:	15100:	15101:	15103:	15104:	15105:	15106:	15107:
U 14A8	15108:	15185:	15187:	15190:	15192:	15194:	15196:	15198:
U 14B0	15199:	15200:	15201:	15202:	15203:	15204:	15205:	15207:
U 14B8	15208:	15209:	15210:	15212:	15213:	15215:	15216:	15218:
U 14C0	15220:	15221:	15222:	15223:	15224:	15225:	15227:	15228:
U 14C8	15229:	15230:	15231:	15234:	15235:	15236:	15237:	15238:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 14D0	15240:	15241:	15242:	15243:	15244:	15246:	15248:	15251:
U 14D8	15252:	15253:	15254:	15255:	15256:	15257:	15259:	15261:
U 14E0	15262:	15263:	15264:	15266:	15267:	15269:	15270:	15271:
U 14E8	15272:	15273:	15274:	15342:	15344:	15347:	15349:	15351:
U 14F0	15353:	15355:	15356:	15357:	15358:	15359:	15360:	15361:
U 14F8	15362:	15364:	15365:	15367:	15368:	15370:	15371:	15373:
U 1500	15374:	15376:	15377:	15379:	15381:	15382:	15383:	15385:
U 1508	15386:	15388:	15389:	15390:	15391:	15393:	15396:	15397:
U 1510	15398:	15399:	15400:	15401:	15402:	15403:	15404:	15405:
U 1518	15406:	15407:	15408:	15410:	15412:	15413:	15414:	15415:
U 1520	15417:	15418:	15420:	15421:	15422:	15423:	15424:	15425:
U 1528	15499:	15501:	15504:	15506:	15508:	15510:	15512:	15513:
U 1530	15514:	15515:	15516:	15517:	15518:	15519:	15521:	15522:
U 1538	15524:	15525:	15527:	15528:	15530:	15531:	15533:	15534:
U 1540	15536:	15538:	15539:	15540:	15541:	15542:	15544:	15545:
U 1548	15546:	15547:	15549:	15551:	15554:	15555:	15556:	15557:
U 1550	15558:	15559:	15560:	15561:	15562:	15563:	15564:	15565:
U 1558	15566:	15568:	15570:	15571:	15572:	15573:	15575:	15576:
U 1560	15578:	15579:	15580:	15581:	15582:	15583:	15661:	15663:
U 1568	15666:	15668:	15670:	15672:	15674:	15675:	15676:	15677:
U 1570	15678:	15679:	15680:	15681:	15683:	15684:	15685:	15686:
U 1578	15688:	15689:	15691:	15692:	15694:	15695:	15697:	15699:
U 1580	15700:	15701:	15702:	15703:	15704:	15705:	15706:	15708:
U 1588	15709:	15710:	15711:	15712:	15714:	15715:	15716:	15717:
U 1590	15718:	15719:	15720:	15721:	15722:	15723:	15725:	15726:
U 1598	15727:	15728:	15730:	15733:	15734:	15735:	15736:	15737:
U 15A0	15738:	15740:	15741:	15742:	15743:	15745:	15747:	15748:
U 15A8	15749:	15750:	15752:	15753:	15755:	15756:	15757:	15758:
U 15B0	15759:	15760:	15842:	15844:	15847:	15849:	15851:	15853:
U 15B8	15855:	15856:	15857:	15858:	15859:	15860:	15861:	15862:
U 15C0	15864:	15865:	15866:	15867:	15869:	15870:	15872:	15873:
U 15C8	15875:	15876:	15878:	15879:	15881:	15882:	15884:	15886:
U 15D0	15887:	15888:	15889:	15890:	15891:	15892:	15893:	15895:
U 15D8	15896:	15897:	15898:	15899:	15901:	15902:	15903:	15904:
U 15E0	15905:	15906:	15907:	15908:	15909:	15910:	15912:	15913:
U 15E8	15914:	15915:	15916:	15918:	15921:	15922:	15923:	15924:
U 15F0	15925:	15926:	15927:	15928:	15929:	15930:	15931:	15932:
U 15F8	15933:	15934:	15935:	15936:	15938:	15940:	15941:	15942:
U 1600	15943:	15945:	15946:	15948:	15949:	15950:	15951:	15952:
U 1608	15953:	16015:	16017:	16020:	16022:	16024:	16026:	16028:
U 1610	16029:	16030:	16031:	16032:	16033:	16034:	16035:	16037:
U 1618	16038:	16040:	16041:	16043:	16044:	16046:	16048:	16049:
U 1620	16050:	16051:	16052:	16054:	16055:	16056:	16058:	16061:
U 1628	16062:	16063:	16064:	16066:	16067:	16068:	16070:	16072:
U 1630	16073:	16074:	16075:	16077:	16078:	16080:	16081:	16082:
U 1638	16083:	16084:	16085:	16144:	16146:	16149:	16151:	16153:
U 1640	16155:	16156:	16157:	16158:	16159:	16160:	16161:	16162:
U 1648	16163:	16164:	16165:	16166:	16167:	16168:	16169:	16171:
U 1650	16172:	16173:	16174:	16175:	16176:	16177:	16178:	16179:
U 1658	16180:	16181:	16182:	16183:	16184:	16186:	16187:	16188:
U 1660	16189:	16190:	16191:	16192:	16193:	16194:	16195:	16196:
U 1668	16197:	16198:	16199:	16201:	16202:	16488:	16490:	16493:
U 1670	16495:	16497:	16499:	16500:	16501:	16502:	16503:	16504:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1678	16505:	16506:	16507:	16508:	16509:	16510:	16511:	16512:
U 1680	16513:	16514:	16515:	16516:	16517:	16518:	16519:	16520:
U 1688	16521:	16522:	16523:	16524:	16525:	16526:	16527:	16528:
U 1690	16529:	16530:	16531:	16533:	16534:	16535:	16536:	16537:
U 1698	16538:	16539:	16540:	16541:	16542:	16543:	16544:	16545:
U 16A0	16546:	16547:	16548:	16549:	16550:	16551:	16552:	16553:
U 16A8	16554:	16555:	16556:	16557:	16558:	16559:	16560:	16561:
U 16B0	16562:	16563:	16564:	16565:	16566:	16567:	16569:	16570:
U 16B8	16571:	16572:	16573:	16574:	16575:	16576:	16577:	16578:
U 16C0	16579:	16580:	16581:	16582:	16583:	16584:	16585:	16586:
U 16C8	16587:	16588:	16589:	16590:	16591:	16592:	16593:	16594:
U 16D0	16595:	16596:	16597:	16598:	16599:	16600:	16601:	16602:
U 16D8	16603:	16605:	16606:	16607:	16608:	16609:	16610:	16611:
U 16E0	16612:	16613:	16614:	16615:	16616:	16617:	16618:	16619:
U 16E8	16620:	16621:	16622:	16623:	16624:	16625:	16626:	16627:
U 16F0	16628:	16629:	16630:	16631:	16632:	16633:	16634:	16635:
U 16F8	16636:	16637:	16638:	16639:	16640:	16641:	16642:	16643:
U 1700	16644:	16645:	16646:	16647:	16648:	16649:	16650:	16651:
U 1708	16652:	16653:	16654:	16655:	16656:	16657:	16658:	16659:
U 1710	16660:	16661:	16662:	16663:	16664:	16665:	16666:	16667:
U 1718	16668:	16669:	16671:	16672:	16674:	16676:	16678:	16679:
U 1720	16680:	16681:	16682:	16683:	16684:	16685:	16686:	16687:
U 1728	16688:	16689:	16690:	16691:	16692:	16693:	16694:	16695:
U 1730	16696:	16697:	16698:	16699:	16700:	16701:	16702:	16704:
U 1738	16705:	16706:	16707:	16708:	16709:	16710:	16711:	16712:
U 1740	16713:	16714:	16715:	16716:	16717:	16718:	16719:	16720:
U 1748	16721:	16723:	16724:	16725:	16726:	16727:	16728:	16729:
U 1750	16730:	16731:	16732:	16733:	16734:	16735:	16736:	16737:
U 1758	16738:	16739:	16740:	16741:	16742:	16743:	16744:	16745:
U 1760	16746:	16747:	16748:	16749:	16750:	16751:	16752:	16753:
U 1768	16754:	16755:	16756:	16757:	16758:	16759:	16760:	16761:
U 1770	16762:	16763:	16764:	16765:	16766:	16767:	16768:	16769:
U 1778	16770:	16771:	16772:	16773:	16774:	16775:	16776:	16777:
U 1780	16778:	16779:	16780:	16781:	16782:	16783:	16784:	16785:
U 1788	16786:	16787:	16788:	16789:	16791:	16792:	16793:	16794:
U 1790	16795:	16796:	16797:	16798:	16799:	16800:	16801:	16802:
U 1798	16803:	16806:	16807:	16808:	16809:	16810:	16811:	16812:
U 17A0	16813:	16814:	16816:	16817:	16818:	16819:	16820:	16821:
U 17A8	16822:	16823:	16824:	16825:	16826:	16827:	16828:	16829:
U 17B0	16830:	16831:	16832:	16833:	16834:	16835:	16836:	16837:
U 17B8	16839:	16840:	16841:	16842:	16843:	16844:	16845:	16846:
U 17C0	16847:	16848:	16849:	16850:	16851:	16852:	16853:	16854:
U 17C8	16855:	16856:	16857:	16858:	16859:	16860:	16861:	16862:
U 17D0	16863:	16864:	16865:	16866:	16867:	16868:	16869:	16870:
U 17D8	16871:	16872:	16873:	16874:	16875:	16876:	16877:	16878:
U 17E0	16879:	16880:	16881:	16882:	16883:	16884:	16885:	16886:
U 17E8	16888:	16890:	16892:	16893:	16894:	16895:	16896:	16897:
U 17F0	16898:	16899:	16900:	16901:	16902:	16903:	16904:	16905:
U 17F8	16906:	16907:	16908:	16909:	16910:	16911:	16912:	16913:
U 1800	16914:	16915:	16916:	16917:	16918:	16919:	16920:	16921:
U 1808	16922:	16923:	16924:	16925:	16926:	16927:	16928:	16929:
U 1810	16930:	16931:	16932:	16933:	16934:	16935:	16936:	16937:
U 1818	16938:	16939:	16940:	16941:	16942:	16944:	16945:	16946:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1820	16947:	16948:	16949:	16950:	16951:	16952:	16953:	16954:
U 1828	16955:	16956:	16957:	16958:	16959:	16960:	16961:	16962:
U 1830	16963:	16964:	16965:	16966:	16967:	16968:	16969:	16970:
U 1838	16971:	16972:	16973:	16974:	16975:	16977:	16978:	16979:
U 1840	16980:	16981:	16982:	16983:	16984:	16985:	16986:	16988:
U 1848	16990:	16992:	16993:	16994:	16995:	16996:	16997:	16998:
U 1850	16999:	17000:	17001:	17002:	17003:	17005:	17006:	17007:
U 1858	17008:	17009:	17010:	17011:	17012:	17013:	17014:	17015:
U 1860	17016:	17017:	17018:	17019:	17020:	17022:	17024:	17026:
U 1868	17027:	17028:	17029:	17030:	17031:	17032:	17033:	17034:
U 1870	17035:	17036:	17037:	17038:	17039:	17040:	17041:	17042:
U 1878	17043:	17044:	17045:	17046:	17047:	17048:	17049:	17050:
U 1880	17051:	17052:	17053:	17054:	17055:	17056:	17057:	17058:
U 1888	17059:	17060:	17061:	17062:	17063:	17064:	17065:	17066:
U 1890	17067:	17068:	17069:	17070:	17071:	17072:	17074:	17076:
U 1898	17078:	17079:	17080:	17081:	17082:	17083:	17084:	17085:
U 18A0	17086:	17087:	17088:	17089:	17090:	17091:	17092:	17093:
U 18A8	17094:	17095:	17096:	17097:	17098:	17099:	17318:	17320:
U 18B0	17323:	17325:	17327:	17329:	17330:	17331:	17332:	17333:
U 18B8	17334:	17335:	17336:	17337:	17338:	17339:	17340:	17341:
U 18C0	17342:	17343:	17344:	17345:	17346:	17347:	17348:	17349:
U 18C8	17350:	17351:	17352:	17353:	17354:	17355:	17356:	17357:
U 18D0	17358:	17359:	17360:	17361:	17362:	17363:	17364:	17365:
U 18D8	17366:	17367:	17368:	17369:	17370:	17371:	17372:	17373:
U 18E0	17374:	17375:	17376:	17377:	17378:	17379:	17380:	17381:
U 18E8	17382:	17383:	17384:	17385:	17386:	17387:	17388:	17389:
U 18F0	17390:	17391:	17392:	17393:	17394:	17395:	17396:	17397:
U 18F8	17398:	17399:	17400:	17401:	17402:	17403:	17404:	17405:
U 1900	17406:	17407:	17408:	17409:	17410:	17411:	17412:	17413:
U 1908	17414:	17415:	17416:	17417:	17418:	17419:	17420:	17421:
U 1910	17422:	17423:	17424:	17425:	17426:	17427:	17428:	17429:
U 1918	17430:	17431:	17432:	17433:	17434:	17435:	17436:	17437:
U 1920	17438:	17439:	17440:	17441:	17442:	17443:	17444:	17445:
U 1928	17446:	17447:	17448:	17449:	17450:	17451:	17452:	17453:
U 1930	17454:	17455:	17456:	17457:	17458:	17459:	17460:	17461:
U 1938	17462:	17463:	17464:	17465:	17466:	17467:	17468:	17469:
U 1940	17470:	17471:	17472:	17473:	17474:	17475:	17476:	17477:
U 1948	17478:	17479:	17480:	17481:	17482:	17483:	17484:	17485:
U 1950	17486:	17487:	17488:	17489:	17490:	17491:	17492:	17493:
U 1958	17494:	17495:	17496:	17497:	17498:	17499:	17500:	17501:
U 1960	17502:	17503:	17504:	17505:	17506:	17507:	17508:	17509:
U 1968	17510:	17511:	17512:	17513:	17514:	17515:	17516:	17517:
U 1970	17518:	17519:	17520:	17521:	17522:	17523:	17524:	17525:
U 1978	17526:	17527:	17528:	17529:	17530:	17531:	17532:	17533:
U 1980	17534:	17535:	17536:	17537:	17538:	17539:	17540:	17541:
U 1988	17542:	17543:	17544:	17545:	17546:	17547:	17548:	17549:
U 1990	17550:	17551:	17552:	17553:	17554:	17555:	17556:	17557:
U 1998	17558:	17559:	17560:	17679:	17681:	17684:	17686:	17688:
U 19A0	17690:	17691:	17692:	17693:	17694:	17695:	17696:	17697:
U 19A8	17698:	17699:	17700:	17701:	17702:	17703:	17704:	17705:
U 19B0	17706:	17707:	17708:	17709:	17710:	17711:	17712:	17713:
U 19B8	17714:	17715:	17716:	17717:	17718:	17719:	17720:	17721:
U 19C0	17722:	17723:	17724:	17725:	17726:	17727:	17728:	17729:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 19C8	17730:	17731:	17732:	17733:	17734:	17735:	17736:	17737:
U 19D0	17738:	17739:	17740:	17741:	17742:	17743:	17744:	17745:
U 19D8	17746:	17747:	17748:	17749:	17750:	17751:	17752:	17753:
U 19E0	17754:	17755:	17756:	17757:	17758:	17759:	17760:	17761:
U 19E8	17762:	17763:	17764:	17765:	17766:	17767:	17768:	17769:
U 19F0	17770:	17771:	17772:	17773:	17774:	17775:	17776:	17777:
U 19F8	17778:	17779:	17780:	17781:	17782:	17783:	17784:	17785:
U 1A00	17786:	17787:	17788:	17789:	17790:	17791:	17792:	17793:
U 1A08	17794:	17795:	17796:	17797:	17798:	17799:	17800:	17801:
U 1A10	17802:	17803:	17804:	17805:	17806:	17807:	17808:	17809:
U 1A18	17810:	17811:	17812:	17813:	17814:	17815:	17816:	17817:
U 1A20	17818:	17819:	17820:	17821:	17822:	17823:	17824:	17825:
U 1A28	17826:	17827:	17828:	17829:	17830:	17831:	17832:	17833:
U 1A30	17834:	17835:	17836:	17837:	17838:	17839:	17840.	17841:
U 1A38	17842:	17843:	17844:	17845:	17846:	17847:	17848:	17849:
U 1A40	17850:	17851:	17852:	17853:	17854:	17855:	17856:	17857:
U 1A48	17858:	17859:	17860:	17861:	17862:	17863:	17864:	17865:
U 1A50	17866:	17867:	17868:	17869:	17870:	17871:	17872:	17873:
U 1A58	17874:	17875:	17876:	17877:	17878:	17879:	17880:	17881:
U 1A60	17882:	17883:	17884:	17885:	17886:	17887:	17967:	17969:
U 1A68	17972:	17974:	17976:	17978:	17979:	17980:	17981:	17982:
U 1A70	17983:	17984:	17985:	17986:	17987:	17988:	17989:	17990:
U 1A78	17991:	17992:	17993:	17994:	17995:	17996:	17997:	17998:
U 1A80	17999:	18000:	18001:	18002:	18003:	18004:	18005:	18006:
U 1A88	18007:	18008:	18009:	18010:	18011:	18012:	18013:	18014:
U 1A90	18015:	18016:	18017:	18018:	18019:	18020:	18021:	18022:
U 1A98	18023:	18024:	18025:	18026:	18027:	18028:	18029:	18030:
U 1AA0	18031:	18032:	18033:	18034:	18035:	18036:	18037:	18038:
U 1AA8	18039:	18040:	18041:	18042:	18043:	18044:	18045:	18046:
U 1AB0	18047:	18048:	18049:	18050:	18051:	18052:	18053:	18054:
U 1AB8	18055:	18056:	18057:	18058:	18059:	18060:	18061:	18062:
U 1AC0	18063:	18064:	18065:	18066:	18067:	18068:	18069:	18070:
U 1AC8	18071:	18072:	18073:	18074:	18075:	18076:	18077:	18078:
U 1AD0	18079:	18080:	18081:	18082:	18083:	18084:	18085:	18086:
U 1AD8	18087:	18088:	18089:	18090:	18091:	18092:	18093:	18094:
U 1AE0	18095:	18096:	18097:	18098:	18099:	18100:	18101:	18102:
U 1AE8	18103:	18104:	18105:	18106:	18107:	18108:	18109:	18215:
U 1AF0	18217:	18220:	18222:	18224:	18226:	18227:	18228:	18229:
U 1AF8	18230:	18231:	18232:	18233:	18234:	18235:	18236:	18237:
U 1B00	18238:	18239:	18240:	18241:	18242:	18243:	18244:	18245:
U 1B08	18246:	18247:	18248:	18249:	18250:	18251:	18252:	18253:
U 1B10	18254:	18255:	18256:	18258:	18259:	18260:	18261:	18262:
U 1B18	18265:	18266:	18267:	18268:	18269:	18270:	18271:	18274:
U 1B20	18275:	18276:	18277:	18278:	18287:	18289:	18290:	18291:
U 1B28	18292:	18294:	18295:	18296:	18297:	18299:	18300:	18301:
U 1B30	18302:	18303:	18304:	18368:	18370:	18373:	18375:	18377:
U 1B38	18379:	18380:	18381:	18382:	18383:	18384:	18385:	18386:
U 1B40	18387:	18388:	18389:	18390:	18391:	18392:	18393:	18394:
U 1B48	18395:	18396:	18397:	18398:	18399:	18401:	18402:	18403:
U 1B50	18404:	18405:	18406:	18407:	18408:	18478:	18480:	18483:
U 1B58	18485:	18487:	18489:	18490:	18491:	18492:	18493:	18494:
U 1B60	18496:	18497:	18498:	18499:	18500:	18501:	18502:	18503:
U 1B68	18504:	18505:	18506:	18507:	18508:	18509:	18510:	18511:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1B70	18512:	18513:	18514:	18515:	18516:	18517:	18518:	18519:
U 1B78	18520:	18521:	18522:	18523:	18524:	18619:	18621:	18624:
U 1B80	18626:	18628:	18630:	18631:	18632:	18633:	18634:	18635:
U 1B88	18636:	18637:	18638:	18639:	18640:	18641:	18642:	18644:
U 1B90	18645:	18646:	18648:	18649:	18650:	18651:	18652:	18654:
U 1B98	18656:	18657:	18658:	18659:	18661:	18662:	18664:	18665:
U 1BA0	18666:	18667:	18668:	18669:	18670:	18671:	18673:	18674:
U 1BA8	18675:	18677:	18678:	18679:	18680:	18681:	18682:	18683:
U 1BB0	18684:	18685:	18686:	18687:	18688:	18689:	18690:	18692:
U 1BB8	18693:	18694:	18695:	18696:	18697:	18699:	18700:	18701:
U 1BC0	18702:	18703:	18704:	18705:	18791:	18793:	18796:	18798:
U 1BC8	18800:	18802:	18803:	18804:	18805:	18806:	18807:	18808:
U 1BD0	18809:	18811:	18812:	18813:	18814:	18815:	18816:	18817:
U 1BD8	18818:	18819:	18820:	18821:	18822:	18823:	18824:	18825:
U 1BE0	18826:	18827:	18828:	18829:	18830:	18831:	18832:	18833:
U 1BE8	18834:	18835:	18836:	18837:	18838:	18839:	18840:	18841:
U 1BF0	18842:	18843:	18844:	18845:	18846:	18847:	18848:	18849:
U 1BF8	18850:	18851:	18852:	18853:	18854:	18855:	18856:	18857:
U 1C00	18858:	18859:	18860:	18861:	18862:	18863:	18864:	18865:
U 1C08	18866:	18867:	18868:	18869:	18870:	18871:	18872:	18873:
U 1C10	18874:	18875:	18876:	18877:	18878:	18879:	18880:	18881:
U 1C18	18882:	18883:	18958:	18960:	18963:	18965:	18967:	18969:
U 1C20	18970:	18971:	18972:	18973:	18974:	18975:	18976:	18977:
U 1C28	18978:	18979:	18980:	18981:	18982:	18983:	18984:	18985:
U 1C30	18986:	18987:	18988:	18989:	18990:	18991:	18992:	18993:
U 1C38	18994:	18995:	18996:	18997:	18998:	18999:	19000:	19001:
U 1C40	19002:	19003:	19004:	19005:	19006:	19007:	19008:	19009:
U 1C48	19010:	19011:	19012:	19013:	19014:	19015:	19016:	19017:
U 1C50	19018:	19019:	19020:	19021:	19022:	19023:	19024:	19025:
U 1C58	19026:	19027:	19028:	19029:	19030:	19031:	19032:	19033:
U 1C60	19034:	19035:	19115:	19117:	19120:	19122:	19124:	19126:
U 1C68	19127:	19128:	19129:	19130:	19131:	19132:	19133:	19134:
U 1C70	19135:	19137:	19138:	19139:	19140:	19141:	19142:	19143:
U 1C78	19144:	19145:	19146:	19147:	19148:	19149:	19150:	19151:
U 1C80	19153:	19154:	19156:	19157:	19158:	19159:	19160:	19161:
U 1C88	19162:	19163:	19164:	19165:	19166:	19167:	19168:	19169:
U 1C90	19170:	19171:	19172:	19173:	19174:	19175:	19176:	19177:
U 1C98	19178:	19179:	19180:	19181:	19182:	19183:	19184:	19185:
U 1CA0	19186:	19187:	19188:	19189:	19190:	19191:	19192:	19193:
U 1CA8	19194:	19195:	19196:	19197:	19198:	19199:	19200:	19201:
U 1CB0	19202:	19203:	19204:	19205:	19206:	19207:	19208:	19209:
U 1CB8	19210:	19211:	19212:	19213:	19214:	19215:	19216:	19217:
U 1CC0	19218:	19275:	19277:	19280:	19282:	19284:	19286:	19287:
U 1CC8	19288:	19289:	19290:	19291:	19292:	19293:	19294:	19295:
U 1CD0	19296:	19297:	19298:	19299:	19300:	19301:	19302:	19303:
U 1CD8	19304:	19305:	19306:	19307:	19308:	19309:	19310:	19311:
U 1CE0	19312:	19313:	19314:	19315:	19316:	19317:	19318:	19319:
U 1CE8	19320:	19321:	19322:	19323:	19324:	19394:	19396:	19399:
U 1CF0	19401:	19403:	19405:	19406:	19407:	19408:	19409:	19410:
U 1CF8	19411:	19412:	19413:	19414:	19416:	19417:	19418:	19419:
U 1D00	19420:	19421:	19422:	19423:	19424:	19425:	19426:	19427:
U 1D08	19428:	19429:	19430:	19432:	19433:	19434:	19435:	19436:
U 1D10	19437:	19438:	19439:	19440:	19441:	19442:	19443:	19444:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1D18	19445:	19446:	19447:	19448:	19449:	19450:	19451:	19452:
U 1D20	19453:	19454:	19455:	19456:	19457:	19458:	19459:	19460:
U 1D28	19461:	19462:	19463:	19464:	19465:	19466:	19467:	19468:
U 1D30	19469:	19470:	19471:	19472:	19473:	19474:	19475:	19476:
U 1D38	19477:	19478:	19479:	19480:	19481:	19482:	19483:	19484:
U 1D40	19485:	19486:	19487:	19488:	19489:	19490:	19491:	19492:
U 1D48	19493:	19494:	19495:	19496:	19552:	19554:	19557:	19559:
U 1D50	19561:	19563:	19564:	19565:	19566:	19567:	19568:	19569:
U 1D58	19570:	19571:	19572:	19573:	19574:	19575:	19576:	19577:
U 1D60	19579:	19580:	19581:	19582:	19583:	19585:	19586:	19587:
U 1D68	19588:	19589:	19590:	19591:	19592:	19593:	19594:	19595:
U 1D70	19596:	19597:	19598:	19599:	19601:	19602:	19603:	19604:
U 1D78	19605:	19606:	19607:	19608:	19609:	19610:	19611:	19702:
U 1D80	19704:	19707:	19709:	19711:	19713:	19714:	19715:	19716:
U 1D88	19717:	19718:	19719:	19720:	19721:	19722:	19723:	19724:
U 1D90	19725:	19726:	19727:	19728:	19729:	19730:	19731:	19732:
U 1D98	19733:	19734:	19735:	19736:	19737:	19738:	19739:	19740:
U 1DA0	19741:	19742:	19743:	19744:	19745:	19746:	19747:	19748:
U 1DA8	19749:	19750:	19751:	19752:	19753:	19754:	19755:	19756:
U 1DB0	19757:	19758:	19759:	19760:	19761:	19762:	19763:	19764:
U 1DB8	19765:	19766:	19767:	19768:	19769:	19770:	19771:	19772:
U 1DC0	19773:	19774:	19775:	19776:	19777:	19778:	19779:	19780:
U 1DC8	19781:	19782:	19783:	19784:	19785:	19786:	19787:	19788:
U 1DD0	19789:	19841:	19843:	19846:	19848:	19850:	19852:	19853:
U 1DD8	19854:	19855:	19856:	19857:	19858:	19859:	19860:	19861:
U 1DE0	19862:	19863:	19864:	19865:	19866:	19867:	19868:	19869:
U 1DE8	19870:	19871:	19872:	19873:	19874:	19875:	19876:	19877:
U 1DF0	19878:	19879:	19880:	19881:	19882:	19883:	19884:	19885:
U 1DF8	19886:	19887:	19888:	19889:	19890:	19891:	19892:	19893:
U 1E00	19894:	19895:	19896:	19953:	19955:	19958:	19960:	19962:
U 1E08	19964:	19965:	19966:	19967:	19968:	19969:	19970:	19971:
U 1E10	19972:	19973:	19974:	19975:	19976:	19977:	19978:	19979:
U 1E18	19980:	19981:	19982:	19983:	19984:	19985:	19986:	19987:
U 1E20	19988:	19989:	19990:	19991:	19992:	19993:	19994:	19995:
U 1E28	19996:	19997:	19998:	19999:	20000:	20001:	20002:	20003:
U 1E30	20004:	20005:	20076:	20078:	20081:	20083:	20085:	20087:
U 1E38	20088:	20089:	20090:	20091:	20092:	20093:	20094:	20095:
U 1E40	20096:	20097:	20098:	20099:	20100:	20101:	20102:	20103:
U 1E48	20104:	20105:	20106:	20107:	20108:	20109:	20110:	20111:
U 1E50	20112:	20113:	20114:	20115:	20116:	20117:	20118:	20119:
U 1E58	20120:	20121:	20122:	20123:	20124:	20125:	20126:	20127:
U 1E60	20128:	20129:	20130:	20131:	20132:	20133:	20134:	20135:
U 1E68	20136:	20137:	20138:	20139:	20140:	20141:	20142:	20143:
U 1E70	20144:	20145:	20146:	20147:	20148:	20149:	20150:	20151:
U 1E78	20152:	20153:	20154:	20155:	20156:	20157:	20158:	20159:
U 1E80	20160:	20161:	20162:	20163:	20164:	20165:	20166:	20167:
U 1E88	20168:	20169:	20170:	20171:	20172:	20173:	20174:	20175:
U 1E90	20176:	20177:	20178:	20179:	20180:	20181:	20182:	20183:
U 1E98	20184:	20185:	20186:	20187:	20188:	20189:	20190:	20191:
U 1EA0	20192:	20193:	20194:	20195:	20196:	20197:	20198:	20199:
U 1EA8	20200:	20201:	20202:	20203:	20204:	20205:	20206:	20207:
U 1EB0	20208:	20209:	20210:	20211:	20212:	20213:	20214:	20215:
U 1EB8	20216:	20217:	20218:	20219:	20220:	20221:	20222:	20223:

;
Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 1EC0	20224:	20225:	20226:	20227:	20228:	20229:	20230:	20231:
U 1EC8	20321:	20323:	20326:	20328:	20330:	20332:	20333:	20334:
U 1ED0	20335:	20336:	20337:	20338:	20339:	20340:	20341:	20342:
U 1ED8	20343:	20344:	20345:	20346:	20347:	20348:	20349:	20350:
U 1EE0	20351:	20352:	20353:	20354:	20355:	20356:	20357:	20358:
U 1EE8	20359:	20360:	20361:	20362:	20363:	20364:	20365:	20366:
U 1EF0	20367:	20368:	20369:	20370:	20371:	20372:	20373:	20374:
U 1EF8	20375:	20376:	20377:	20378:	20379:	20380:	20381:	20382:
U 1F00	20383:	20384:	20385:	20386:	20387:	20388:	20389:	20390:
U 1F08	20391:	20392:	20393:	20394:	20395:	20396:	20397:	20398:
U 1F10	20399:	20400:	20401:	20402:	20403:	20404:	20405:	20406:
U 1F18	20407:	20408:	20409:	20410:	20411:	20412:	20413:	20414:
U 1F20	20415:	20416:	20417:	20418:	20419:	20420:	20421:	20422:
U 1F28	20423:	20424:	20425:	20426:	20427:	20428:	20429:	20430:
U 1F30	20431:	20432:	20433:	20434:	20435:	20436:	20437:	20438:
U 1F38	20439:	20440:	20441:	20442:	20443:	20444:	20445:	20446:
U 1F40	20447:	20448:	20449:	20450:	20451:	20452:	20453:	20454:
U 1F48	20455:	20456:	20457:	20458:	20459:	20460:	20461:	20462:
U 1F50	20463:	20464:	20465:	20466:	20467:	20468:	20469:	20470:
U 1F58	20471:	20472:	20473:	20474:	20475:	20476:	20477:	20478:
U 1F60	20479:	20480:	20481:	20482:	20483:	20484:	20485:	20486:
U 1F68	20487:	20488:	20489:	20490:	20491:	20492:	20493:	20494:
U 1F70	20495:	20496:	20497:	20498:	20499:	20500:	20501:	20502:
U 1F78	20503:	20504:	20505:	20506:	20570:	20572:	20575:	20577:
U 1F80	20579:	20581:	20582:	20583:	20584:	20585:	20586:	20587:
U 1F88	20588:	20589:	20590:	20591:	20592:	20593:	20594:	20595:
U 1F90	20596:	20597:	20598:	20599:	20600:	20601:	20602:	20603:
U 1F98	20604:	20605:	20606:	20607:	20608:	20609:	20610:	20611:
U 1FA0	20612:	20613:	20614:	20615:	20616:	20617:	20687:	20689:
U 1FA8	20692:	20694:	20696:	20698:	20699:	20700:	20701:	20702:
U 1FB0	20703:	20704:	20705:	20706:	20707:	20708:	20709:	20710:
U 1FB8	20711:	20712:	20713:	20714:	20715:	20716:	20717:	20718:
U 1FC0	20719:	20720:	20721:	20722:	20723:	20724:	20725:	20726:
U 1FC8	20727:	20728:	20729:	20730:	20731:	20732:	20733:	20734:
U 1FD0	20735:	20736:	20737:	20738:	20739:	20740:	20741:	20742:
U 1FD8	20743:	20744:	20745:	20746:	20747:	20748:	20749:	20750:
U 1FE0	20751:	20752:	20753:	20754:	20755:	20756:	20757:	20758:
U 1FE8	20759:	20760:	20761:	20762:	20763:	20764:	20765:	20766:
U 1FF0	20767:	20768:	20769:	20770:	20771:	20772:	20773:	20774:
U 1FF8	20775:	20776:	20777:	20778:	20779:	20780:	20781:	20782:
U 2000	20783:	20784:	20785:	20786:	20787:	20788:	20789:	20790:
U 2008	20791:	20792:	20793:	20794:	20795:	20796:	20797:	20798:
U 2010	20799:	20800:	20801:	20802:	20803:	20804:	20805:	20806:
U 2018	20807:	20808:	20809:	20810:	20811:	20812:	20813:	20814:
U 2020	20815:	20816:	20817:	20818:	20819:	20820:	20821:	20822:
U 2028	20823:	20824:	20825:	20826:	20827:	20828:	20829:	20830:
U 2030	20831:	20891:	20893:	20896:	20898:	20900:	20902:	20903:
U 2038	20904:	20905:	20906:	20907:	20908:	20909:	20910:	20911:
U 2040	20912:	20913:	20914:	20915:	20916:	20917:	20918:	20919:
U 2048	20920:	20921:	20922:	20923:	20924:	20925:	20926:	20927:
U 2050	20928:	20929:	20930:	20931:	20932:	20933:	20934:	20935:
U 2058	20936:	20937:	20938:	21002:	21004:	21007:	21009:	21011:
U 2060	21013:	21014:	21015:	21016:	21017:	21018:	21019:	21020:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2068	21021:	21022:	21023:	21024:	21025:	21026:	21027:	21028:
U 2070	21029:	21030:	21031:	21032:	21033:	21034:	21035:	21036:
U 2078	21037:	21038:	21039:	21040:	21041:	21042:	21043:	21044:
U 2080	21045:	21046:	21047:	21048:	21049:	21050:	21051:	21052:
U 2088	21053:	21054:	21055:	21056:	21057:	21058:	21059:	21060:
U 2090	21061:	21062:	21063:	21064:	21065:	21066:	21067:	21068:
U 2098	21069:	21070:	21071:	21072:	21073:	21074:	21075:	21076:
U 20A0	21077:	21078:	21079:	21080:	21081:	21082:	21083:	21084:
U 20A8	21085:	21086:	21087:	21088:	21089:	21090:	21091:	21092:
U 20B0	21093:	21204:	21206:	21209:	21211:	21213:	21215:	21216:
U 20B8	21217:	21218:	21219:	21220:	21221:	21222:	21223:	21224:
U 20C0	21225:	21226:	21227:	21228:	21229:	21230:	21231:	21232:
U 20C8	21233:	21234:	21235:	21236:	21237:	21238:	21239:	21240:
U 20D0	21241:	21242:	21243:	21245:	21246:	21247:	21248:	21249:
U 20D8	21250:	21251:	21252:	21253:	21254:	21255:	21256:	21257:
U 20E0	21258:	21259:	21260:	21261:	21262:	21263:	21264:	21265:
U 20E8	21266:	21267:	21268:	21269:	21270:	21271:	21272:	21273:
U 20F0	21274:	21275:	21276:	21277:	21278:	21279:	21280:	21281:
U 20F8	21282:	21283:	21284:	21285:	21286:	21287:	21288:	21289:
U 2100	21290:	21291:	21292:	21293:	21294:	21295:	21296:	21297:
U 2108	21298:	21299:	21300:	21301:	21302:	21303:	21304:	21305:
U 2110	21306:	21307:	21308:	21309:	21310:	21311:	21312:	21313:
U 2118	21314:	21315:	21316:	21317:	21318:	21319:	21320:	21321:
U 2120	21322:	21323:	21324:	21325:	21326:	21327:	21328:	21329:
U 2128	21330:	21331:	21332:	21333:	21334:	21342:	21343:	21344:
U 2130	21345:	21346:	21347:	21348:	21349:	21350:	21351:	21352:
U 2138	21353:	21354:	21355:	21356:	21357:	21358:	21359:	21360:
U 2140	21361:	21362:	21363:	21364:	21365:	21366:	21367:	21368:
U 2148	21369:	21370:	21371:	21372:	21373:	21374:	21375:	21376:
U 2150	21377:	21378:	21379:	21380:	21381:	21382:	21383:	21384:
U 2158	21385:	21386:	21387:	21388:	21389:	21390:	21391:	21392:
U 2160	21393:	21394:	21395:	21396:	21397:	21398:	21399:	21400:
U 2168	21401:	21402:	21403:	21458:	21460:	21463:	21465:	21467:
U 2170	21469:	21470:	21471:	21472:	21473:	21474:	21475:	21476:
U 2178	21477:	21478:	21479:	21480:	21481:	21482:	21483:	21484:
U 2180	21485:	21486:	21487:	21488:	21489:	21490:	21491:	21492:
U 2188	21493:	21494:	21495:	21496:	21497:	21498:	21499:	21500:
U 2190	21501:	21502:	21503:	21504:	21505:	21506:	21507:	21508:
U 2198	21509:	21510:	21511:	21512:	21513:	21514:	21515:	21516:
U 21A0	21517:	21518:	21519:	21520:	21521:	21522:	21523:	21524:
U 21A8	21525:	21526:	21527:	21528:	21529:	21530:	21531:	21532:
U 21B0	21533:	21534:	21535:	21536:	21537:	21538:	21539:	21540:
U 21B8	21541:	21542:	21543:	21544:	21545:	21546:	21547:	21548:
U 21C0	21549:	21550:	21551:	21552:	21553:	21554:	21555:	21556:
U 21C8	21557:	21558:	21559:	21560:	21682:	21684:	21687:	21689:
U 21D0	21691:	21693:	21694:	21695:	21696:	21697:	21698:	21699:
U 21D8	21700:	21701:	21702:	21703:	21704:	21705:	21706:	21707:
U 21E0	21708:	21709:	21710:	21711:	21712:	21713:	21714:	21715:
U 21E8	21716:	21717:	21718:	21719:	21720:	21721:	21722:	21723:
U 21F0	21724:	21725:	21726:	21727:	21728:	21729:	21730:	21731:
U 21F8	21732:	21733:	21734:	21735:	21736:	21737:	21738:	21739:
U 2200	21740:	21741:	21742:	21743:	21744:	21745:	21746:	21747:
U 2208	21748:	21749:	21750:	21751:	21752:	21753:	21754:	21755:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2210	21756:	21757:	21758:	21759:	21760:	21761:	21762:	21763:
U 2218	21764:	21765:	21766:	21767:	21768:	21769:	21770:	21771:
U 2220	21772:	21773:	21774:	21775:	21776:	21777:	21778:	21779:
U 2228	21780:	21781:	21782:	21783:	21784:	21785:	21786:	21787:
U 2230	21788:	21789:	21790:	21791:	21792:	21793:	21794:	21795:
U 2238	21796:	21797:	21798:	21799:	21800:	21801:	21802:	21803:
U 2240	21804:	21805:	21806:	21807:	21808:	21809:	21810:	21811:
U 2248	21812:	21813:	21814:	21815:	21816:	21817:	21818:	21819:
U 2250	21820:	21821:	21822:	21823:	21824:	21825:	21826:	21827:
U 2258	21828:	21829:	21830:	21831:	21832:	21833:	21834:	21835:
U 2260	21836:	21837:	21838:	21839:	21840:	21841:	21842:	21843:
U 2268	21844:	21845:	21846:	21847:	21848:	21849:	21850:	21851:
U 2270	21852:	21853:	21854:	21855:	21856:	21857:	21858:	21859:
U 2278	21860:	21861:	21862:	21863:	21864:	21865:	21866:	21867:
U 2280	21868:	21869:	21870:	21871:	21872:	21873:	21874:	21875:
U 2288	21876:	21877:	21878:	21879:	21880:	21881:	21882:	21883:
U 2290	21884:	21885:	21886:	21887:	21888:	21889:	21890:	21891:
U 2298	21892:	21893:	21894:	21895:	21896:	21897:	21898:	21899:
U 22A0	21900:	21901:	21902:	21903:	21904:	21905:	21906:	21907:
U 22A8	21908:	21909:	21910:	21911:	21912:	21913:	21914:	21915:
U 22B0	21916:	21917:	21918:	21919:	21920:	21921:	21922:	21923:
U 22B8	21924:	21925:	21926:	21927:	21928:	21929:	21930:	21931:
U 22C0	21932:	21933:	21934:	21935:	21936:	21937:	21938:	21939:
U 22C8	21940:	21941:	21942:	21943:	21944:	21945:	21946:	21947:
U 22D0	21948:	21949:	21950:	21951:	21952:	21953:	21954:	21955:
U 22D8	21956:	21957:	22083:	22085:	22088:	22090:	22092:	22094:
U 22E0	22095:	22096:	22097:	22098:	22099:	22100:	22101:	22102:
U 22E8	22103:	22104:	22105:	22106:	22107:	22108:	22109:	22110:
U 22F0	22111:	22112:	22113:	22114:	22115:	22116:	22117:	22118:
U 22F8	22119:	22120:	22121:	22122:	22123:	22124:	22125:	22126:
U 2300	22127:	22128:	22129:	22130:	22131:	22132:	22133:	22134:
U 2308	22135:	22136:	22137:	22138:	22139:	22140:	22141:	22142:
U 2310	22143:	22144:	22145:	22146:	22147:	22148:	22149:	22150:
U 2318	22151:	22152:	22153:	22154:	22155:	22156:	22157:	22158:
U 2320	22159:	22160:	22161:	22162:	22163:	22164:	22165:	22166:
U 2328	22167:	22168:	22169:	22170:	22171:	22172:	22173:	22174:
U 2330	22175:	22176:	22177:	22178:	22179:	22180:	22181:	22182:
U 2338	22183:	22184:	22185:	22186:	22187:	22188:	22189:	22190:
U 2340	22191:	22192:	22193:	22194:	22195:	22196:	22197:	22198:
U 2348	22199:	22200:	22201:	22202:	22203:	22204:	22205:	22206:
U 2350	22207:	22208:	22209:	22210:	22211:	22212:	22213:	22214:
U 2358	22215:	22216:	22217:	22218:	22219:	22220:	22221:	22222:
U 2360	22223:	22224:	22225:	22226:	22227:	22228:	22229:	22230:
U 2368	22231:	22232:	22233:	22234:	22235:	22236:	22237:	22238:
U 2370	22239:	22240:	22241:	22242:	22243:	22244:	22245:	22246:
U 2378	22247:	22248:	22249:	22250:	22251:	22252:	22253:	22254:
U 2380	22255:	22256:	22257:	22258:	22259:	22260:	22261:	22262:
U 2388	22263:	22264:	22265:	22266:	22267:	22268:	22269:	22270:
U 2390	22271:	22272:	22273:	22274:	22275:	22276:	22277:	22278:
U 2398	22279:	22280:	22281:	22282:	22283:	22284:	22285:	22286:
U 23A0	22287:	22288:	22289:	22290:	22291:	22292:	22293:	22294:
U 23A8	22295:	22296:	22297:	22298:	22299:	22300:	22301:	22302:
U 23B0	22303:	22304:	22305:	22306:	22307:	22308:	22309:	22310:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 23B8	22311:	22312:	22313:	22314:	22315:	22316:	22317:	22318:
U 23C0	22319:	22320:	22321:	22322:	22323:	22324:	22325:	22326:
U 23C8	22327:	22328:	22329:	22330:	22331:	22332:	22333:	22334:
U 23D0	22335:	22336:	22337:	22338:	22339:	22340:	22341:	22342:
U 23D8	22343:	22344:	22345:	22346:	22347:	22348:	22349:	22350:
U 23E0	22351:	22352:	22353:	22354:	22355:	22356:	22357:	22358:
U 23E8	22359:	22360:	22361:	22362:	22363:	22364:	22365:	22366:
U 23F0	22367:	22368:	22369:	22370:	22371:	22372:	22373:	22374:
U 23F8	22375:	22376:	22377:	22378:	22379:	22380:	22381:	22382:
U 2400	22383:	22384:	22385:	22386:	22387:	22388:	22389:	22390:
U 2408	22391:	22392:	22393:	22394:	22395:	22396:	22397:	22398:
U 2410	22399:	22400:	22401:	22402:	22403:	22404:	22405:	22406:
U 2418	22407:	22408:	22409:	22410:	22411:	22412:	22526:	22528:
U 2420	22531:	22533:	22535:	22537:	22538:	22539:	22540:	22541:
U 2428	22542:	22543:	22544:	22545:	22546:	22547:	22548:	22549:
U 2430	22550:	22551:	22552:	22553:	22554:	22555:	22556:	22557:
U 2438	22558:	22559:	22560:	22561:	22562:	22563:	22564:	22565:
U 2440	22566:	22567:	22568:	22569:	22570:	22571:	22572:	22573:
U 2448	22574:	22575:	22576:	22577:	22578:	22579:	22580:	22581:
U 2450	22582:	22583:	22584:	22585:	22586:	22587:	22588:	22589:
U 2458	22590:	22591:	22592:	22593:	22594:	22595:	22596:	22597:
U 2460	22598:	22599:	22600:	22601:	22602:	22603:	22604:	22605:
U 2468	22606:	22607:	22608:	22609:	22610:	22611:	22612:	22613:
U 2470	22614:	22615:	22616:	22617:	22618:	22619:	22620:	22621:
U 2478	22622:	22623:	22624:	22625:	22626:	22627:	22628:	22629:
U 2480	22630:	22631:	22632:	22633:	22634:	22635:	22636:	22637:
U 2488	22638:	22639:	22640:	22641:	22642:	22643:	22644:	22645:
U 2490	22646:	22647:	22648:	22649:	22650:	22651:	22652:	22653:
U 2498	22654:	22655:	22656:	22657:	22658:	22659:	22660:	22661:
U 24A0	22662:	22663:	22664:	22665:	22666:	22667:	22668:	22669:
U 24A8	22670:	22671:	22672:	22673:	22674:	22675:	22676:	22677:
U 24B0	22678:	22679:	22680:	22681:	22682:	22683:	22684:	22685:
U 24B8	22686:	22687:	22688:	22689:	22690:	22691:	22692:	22693:
U 24C0	22694:	22695:	22696:	22697:	22698:	22699:	22700:	22701:
U 24C8	22702:	22703:	22704:	22705:	22706:	22707:	22708:	22709:
U 24D0	22710:	22711:	22712:	22713:	22714:	22715:	22716:	22717:
U 24D8	22718:	22719:	22720:	22721:	22722:	22723:	22724:	22725:
U 24E0	22726:	22727:	22728:	22729:	22730:	22731:	22732:	22733:
U 24E8	22734:	22735:	22736:	22737:	22738:	22739:	22740:	22741:
U 24F0	22742:	22743:	22744:	22745:	22746:	22747:	22748:	22749:
U 24F8	22750:	22751:	22752:	22753:	22754:	22755:	22756:	22757:
U 2500	22758:	22759:	22760:	22761:	22762:	22763:	22764:	22765:
U 2508	22766:	22767:	22768:	22769:	22770:	22771:	22772:	22773:
U 2510	22774:	22775:	22776:	22777:	22778:	22779:	22780:	22781:
U 2518	22782:	22783:	22784:	22785:	22786:	22787:	22788:	22789:
U 2520	22790:	22791:	22792:	22793:	22794:	22795:	22796:	22797:
U 2528	22798:	22799:	22800:	22801:	22802:	22803:	22804:	22805:
U 2530	22806:	22807:	22808:	22809:	22810:	22811:	22812:	22813:
U 2538	22814:	22815:	22816:	22817:	22818:	22819:	22820:	22821:
U 2540	22822:	22823:	22824:	22825:	22826:	22827:	22828:	22829:
U 2548	22830:	22831:	22832:	22833:	22834:	22835:	22836:	22837:
U 2550	22838:	22839:	22840:	22841:	22842:	22843:	22844:	22845:
U 2558	22846:	22847:	22848:	22849:	22850:	22851:	22852:	22853:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2560	22854:	22855:	22856:	22857:	22858:	22859:	22860:	22861:
U 2568	22862:	22863:	22864:	22865:	22866:	22867:	22868:	22869:
U 2570	22870:	22871:	22872:	22873:	22874:	22875:	22876:	22877:
U 2578	22878:	22879:	22880:	22881:	22882:	22883:	22884:	22885:
U 2580	22886:	22887:	22888:	22889:	22890:	22891:	22892:	22893:
U 2588	22894:	22895:	22896:	22897:	22898:	22899:	22900:	22901:
U 2590	22902:	22903:	22904:	22905:	22906:	22907:	22908:	22909:
U 2598	22910:	22911:	22912:	22913:	22914:	22915:	22916:	22917:
U 25A0	22918:	22919:	22920:	22921:	22922:	22923:	22924:	22925:
U 25A8	22926:	22927:	22928:	23057:	23059:	23062:	23064:	23066:
U 25B0	23069:	23070:	23071:	23072:	23073:	23074:	23075:	23076:
U 25B8	23077:	23078:	23079:	23080:	23081:	23082:	23083:	23084:
U 25C0	23085:	23086:	23087:	23088:	23089:	23090:	23092:	23093:
U 25C8	23098:	23099:	23100:	23101:	23102:	23107:	23108:	23109:
U 25D0	23110:	23111:	23112:	23113:	23114:	23115:	23116:	23117:
U 25D8	23118:	23119:	23120:	23121:	23122:	23123:	23124:	23125:
U 25E0	23126:	23127:	23128:	23130:	23131:	23132:	23133:	23134:
U 25E8	23135:	23136:	23137:	23138:	23139:	23140:	23141:	23142:
U 25F0	23143:	23144:	23145:	23146:	23147:	23148:	23150:	23151:
U 25F8	23152:	23153:	23154:	23155:	23156:	23157:	23158:	23159:
U 2600	23160:	23161:	23162:	23163:	23164:	23165:	23166:	23167:
U 2608	23169:	23170:	23171:	23172:	23173:	23174:	23175:	23176:
U 2610	23177:	23178:	23179:	23180:	23181:	23182:	23183:	23184:
U 2618	23185:	23186:	23188:	23189:	23190:	23191:	23192:	23193:
U 2620	23194:	23195:	23196:	23197:	23198:	23199:	23200:	23201:
U 2628	23202:	23203:	23204:	23205:	23206:	23208:	23209:	23210:
U 2630	23211:	23212:	23213:	23214:	23215:	23216:	23217:	23218:
U 2638	23219:	23220:	23221:	23222:	23223:	23224:	23225:	23226:
U 2640	23227:	23228:	23230:	23231:	23232:	23233:	23234:	23235:
U 2648	23236:	23237:	23238:	23239:	23240:	23241:	23242:	23243:
U 2650	23244:	23245:	23246:	23247:	23248:	23250:	23251:	23252:
U 2658	23253:	23254:	23255:	23256:	23257:	23258:	23259:	23260:
U 2660	23261:	23262:	23263:	23264:	23265:	23266:	23267:	23268:
U 2668	23269:	23271:	23272:	23273:	23274:	23275:	23276:	23277:
U 2670	23278:	23279:	23280:	23281:	23282:	23283:	23284:	23285:
U 2678	23286:	23288:	23289:	23290:	23291:	23292:	23293:	23294:
U 2680	23295:	23296:	23297:	23298:	23299:	23301:	23302:	23303:
U 2688	23304:	23305:	23306:	23307:	23308:	23310:	23316:	23317:
U 2690	23318:	23319:	23320:	23321:	23322:	23323:	23324:	23326:
U 2698	23327:	23328:	23330:	23331:	23332:	23333:	23334:	23335:
U 26A0	23336:	23337:	23338:	23340:	23341:	23342:	23343:	23345:
U 26A8	23348:	23349:	23350:	23351:	23352:	23353:	23354:	23355:
U 26B0	23356:	23357:	23358:	23359:	23360:	23361:	23362:	23363:
U 26B8	23364:	23365:	23366:	23372:	23373:	23374:	23375:	23376:
U 26C0	23377:	23378:	23379:	23380:	23381:	23382:	23383:	23384:
U 26C8	23385:	23386:	23387:	23388:	23389:	23390:	23396:	23397:
U 26D0	23398:	23399:	23400:	23401:	23402:	23403:	23404:	23405:
U 26D8	23406:	23407:	23408:	23409:	23410:	23411:	23417:	23418:
U 26E0	23419:	23420:	23421:	23422:	23423:	23424:	23425:	23426:
U 26E8	23427:	23428:	23429:	23435:	23436:	23437:	23438:	23439:
U 26F0	23440:	23441:	23442:	23446:	23447:	23448:	23449:	23450:
U 26F8	23451:	23674:	23676:	23679:	23681:	23683:	23685:	23686:
U 2700	23687:	23688:	23689:	23690:	23691:	23692:	23693:	23694:

;
; Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2708	23695:	23696:	23697:	23698:	23699:	23700:	23701:	23702:
U 2710	23703:	23704:	23705:	23706:	23707:	23708:	23709:	23710:
U 2718	23711:	23712:	23713:	23714:	23715:	23716:	23717:	23718:
U 2720	23719:	23720:	23721:	23722:	23723:	23724:	23725:	23726:
U 2728	23727:	23728:	23729:	23730:	23731:	23732:	23733:	23734:
U 2730	23735:	23736:	23737:	23738:	23739:	23740:	23741:	23742:
U 2738	23743:	23744:	23745:	23746:	23747:	23748:	23749:	23750:
U 2740	23751:	23752:	23753:	23754:	23755:	23756:	23757:	23758:
U 2748	23759:	23760:	23761:	23762:	23763:	23764:	23765:	23766:
U 2750	23767:	23768:	23769:	23770:	23771:	23772:	23773:	23774:
U 2758	23775:	23776:	23777:	23778:	23779:	23780:	23781:	23782:
U 2760	23783:	23784:	23785:	23786:	23787:	23788:	23789:	23790:
U 2768	23791:	23792:	23793:	23794:	23795:	23796:	23797:	23798:
U 2770	23799:	23800:	23801:	23802:	23803:	23804:	23805:	23806:
U 2778	23807:	23808:	23809:	23810:	23811:	23812:	23813:	23814:
U 2780	23815:	23816:	23817:	23818:	23819:	23820:	23821:	23822:
U 2788	23823:	23824:	23825:	23826:	23827:	23828:	23829:	23830:
U 2790	23831:	23832:	23833:	23834:	23835:	23836:	23837:	23838:
U 2798	23839:	23840:	23841:	23842:	23843:	23844:	23845:	23846:
U 27A0	23847:	23848:	23849:	23850:	23851:	23852:	23853:	23854:
U 27A8	23855:	23856:	23857:	23858:	23859:	23860:	23861:	23862:
U 27B0	23863:	23864:	23865:	23866:	23867:	23868:	23869:	23870:
U 27B8	23871:	23872:	23873:	23874:	23875:	23876:	23877:	23878:
U 27C0	23879:	23880:	23881:	23882:	23883:	23884:	23885:	23886:
U 27C8	23887:	23888:	23889:	23890:	23891:	23892:	23893:	23894:
U 27D0	23895:	23896:	23897:	23898:	23899:	23900:	23901:	23902:
U 27D8	23903:	23904:	23905:	23906:	23907:	23908:	23909:	23910:
U 27E0	23911:	23912:	23913:	23914:	23915:	23916:	23917:	23918:
U 27E8	23919:	23920:	23921:	23922:	23923:	23924:	23925:	23926:
U 27F0	23927:	23928:	23929:	23930:	23931:	23932:	23933:	23934:
U 27F8	23935:	23936:	23937:	23938:	23939:	23940:	23941:	23942:
U 2800	23943:	23944:	23945:	23946:	23947:	23948:	23949:	23950:
U 2808	23951:	23952:	23953:	23954:	23955:	23956:	23957:	23958:
U 2810	23959:	23960:	23961:	23962:	23963:	23964:	23965:	23966:
U 2818	23967:	23968:	23969:	23970:	23971:	23972:	23973:	23974:
U 2820	23975:	23976:	23977:	23978:	23979:	23980:	23981:	23982:
U 2828	23983:	23984:	23985:	23986:	23987:	23988:	23989:	23990:
U 2830	23991:	23992:	23993:	23994:	23995:	23996:	23997:	23998:
U 2838	23999:	24000:	24001:	24002:	24003:	24004:	24005:	24006:
U 2840	24007:	24008:	24009:	24010:	24011:	24012:	24013:	24014:
U 2848	24015:	24016:	24017:	24018:	24019:	24020:	24021:	24022:
U 2850	24023:	24024:	24025:	24026:	24027:	24028:	24029:	24030:
U 2858	24031:	24032:	24033:	24034:	24035:	24036:	24037:	24038:
U 2860	24039:	24040:	24041:	24042:	24043:	24044:	24045:	24046:
U 2868	24047:	24048:	24049:	24050:	24051:	24052:	24053:	24054:
U 2870	24055:	24056:	24057:	24058:	24059:	24060:	24061:	24062:
U 2878	24063:	24064:	24065:	24066:	24067:	24068:	24069:	24070:
U 2880	24071:	24072:	24073:	24074:	24075:	24076:	24077:	24078:
U 2888	24079:	24080:	24081:	24082:	24083:	24084:	24085:	24086:
U 2890	24087:	24088:	24089:	24090:	24091:	24092:	24093:	24094:
U 2898	24095:	24096:	24097:	24098:	24099:	24100:	24101:	24102:
U 28A0	24103:	24104:	24105:	24106:	24107:	24108:	24109:	24110:
U 28A8	24111:	24112:	24113:	24114:	24115:	24116:	24117:	24118:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 28B0	24119:	24120:	24121:	24122:	24123:	24124:	24125:	24126:
U 28B8	24127:	24128:	24129:	24130:	24131:	24132:	24133:	24134:
U 28C0	24135:	24136:	24137:	24138:	24139:	24140:	24141:	24142:
U 28C8	24143:	24144:	24145:	24146:	24147:	24148:	24149:	24150:
U 28D0	24151:	24152:	24153:	24154:	24155:	24156:	24157:	24158:
U 28D8	24159:	24160:	24161:	24162:	24163:	24164:	24165:	24166:
U 28E0	24167:	24168:	24169:	24170:	24171:	24172:	24173:	24174:
U 28E8	24175:	24176:	24177:	24178:	24179:	24180:	24181:	24182:
U 28F0	24183:	24184:	24185:	24186:	24187:	24188:	24189:	24190:
U 28F8	24191:	24192:	24193:	24194:	24195:	24196:	24197:	24198:
U 2900	24199:	24200:	24201:	24202:	24203:	24204:	24205:	24206:
U 2908	24207:	24208:	24209:	24210:	24211:	24212:	24213:	24214:
U 2910	24215:	24216:	24217:	24218:	24219:	24220:	24221:	24222:
U 2918	24223:	24224:	24225:	24226:	24227:	24228:	24229:	24230:
U 2920	24231:	24232:	24233:	24234:	24235:	24236:	24237:	24238:
U 2928	24239:	24240:	24241:	24242:	24243:	24244:	24245:	24246:
U 2930	24247:	24248:	24249:	24250:	24251:	24252:	24253:	24254:
U 2938	24255:	24256:	24257:	24258:	24259:	24260:	24261:	24262:
U 2940	24263:	24264:	24265:	24266:	24267:	24268:	24269:	24270:
U 2948	24271:	24272:	24273:	24274:	24275:	24276:	24277:	24278:
U 2950	24279:	24280:	24281:	24282:	24283:	24284:	24285:	24286:
U 2958	24287:	24288:	24289:	24290:	24291:	24292:	24293:	24294:
U 2960	24295:	24296:	24297:	24298:	24299:	24300:	24301:	24302:
U 2968	24303:	24304:	24305:	24306:	24307:	24308:	24309:	24310:
U 2970	24311:	24312:	24313:	24314:	24315:	24316:	24317:	24318:
U 2978	24319:	24320:	24321:	24322:	24323:	24324:	24325:	24326:
U 2980	24327:	24328:	24329:	24330:	24331:	24332:	24333:	24334:
U 2988	24335:	24336:	24337:	24338:	24339:	24340:	24341:	24342:
U 2990	24343:	24344:	24345:	24346:	24347:	24348:	24349:	24350:
U 2998	24351:	24352:	24353:	24354:	24355:	24356:	24357:	24358:
U 29A0	24359:	24360:	24361:	24362:	24363:	24364:	24365:	24366:
U 29A8	24367:	24368:	24369:	24370:	24371:	24372:	24373:	24374:
U 29B0	24375:	24376:	24377:	24378:	24379:	24380:	24381:	24382:
U 29B8	24383:	24384:	24385:	24386:	24387:	24388:	24389:	24390:
U 29C0	24391:	24392:	24393:	24394:	24395:	24396:	24397:	24398:
U 29C8	24399:	24400:	24401:	24402:	24403:	24404:	24405:	24406:
U 29D0	24407:	24408:	24409:	24410:	24411:	24412:	24413:	24414:
U 29D8	24415:	24416:	24417:	24418:	24419:	24420:	24421:	24422:
U 29E0	24423:	24424:	24425:	24426:	24427:	24428:	24429:	24430:
U 29E8	24431:	24432:	24433:	24434:	24435:	24436:	24437:	24438:
U 29F0	24439:	24440:	24441:	24442:	24443:	24444:	24445:	24446:
U 29F8	24447:	24448:	24449:	24450:	24451:	24452:	24453:	24454:
U 2A00	24455:	24456:	24457:	24458:	24459:	24460:	24461:	24462:
U 2A08	24463:	24464:	24465:	24466:	24467:	24468:	24469:	24470:
U 2A10	24471:	24472:	24473:	24474:	24475:	24476:	24477:	24478:
U 2A18	24479:	24480:	24481:	24482:	24483:	24484:	24485:	24486:
U 2A20	24487:	24488:	24489:	24490:	24491:	24492:	24493:	24494:
U 2A28	24495:	24496:	24497:	24498:	24499:	24500:	24501:	24502:
U 2A30	24503:	24504:	24505:	24506:	24507:	24508:	24509:	24510:
U 2A38	24511:	24512:	24513:	24514:	24515:	24516:	24517:	24518:
U 2A40	24519:	24520:	24521:	24522:	24523:	24524:	24525:	24526:
U 2A48	24527:	24528:	24529:	24530:	24531:	24532:	24533:	24534:
U 2A50	24535:	24536:	24537:	24538:	24539:	24540:	24541:	24542:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2A58	24543:	24544:	24545:	24546:	24547:	24548:	24549:	24550:
U 2A60	24551:	24552:	24553:	24554:	24555:	24556:	24557:	24558:
U 2A68	24559:	24560:	24561:	24562:	24563:	24564:	24565:	24566:
U 2A70	24567:	24568:	24569:	24570:	24571:	24572:	24573:	24574:
U 2A78	24575:	24576:	24577:	24578:	24579:	24580:	24581:	24582:
U 2A80	24583:	24584:	24585:	24586:	24587:	24588:	24589:	24590:
U 2A88	24591:	24592:	24593:	24594:	24595:	24596:	24597:	24598:
U 2A90	24599:	24600:	24601:	24602:	24603:	24604:	24605:	24606:
U 2A98	24607:	24608:	24609:	24610:	24611:	24612:	24613:	24614:
U 2AA0	24615:	24616:	24617:	24618:	24619:	24620:	24621:	24622:
U 2AA8	24623:	24624:	24625:	24626:	24627:	24628:	24629:	24630:
U 2AB0	24631:	24632:	24633:	24634:	24635:	24636:	24637:	24638:
U 2AB8	24639:	24640:	24641:	24642:	24643:	24644:	24645:	24646:
U 2AC0	24647:	24648:	24649:	24650:	24651:	24652:	24653:	24654:
U 2AC8	24655:	24656:	24657:	24658:	24659:	24660:	24661:	24662:
U 2AD0	24663:	24664:	24665:	24666:	24667:	24668:	24669:	24670:
U 2AD8	24671:	24672:	24673:	24674:	24675:	24676:	24677:	24678:
U 2AE0	24679:	24680:	24681:	24682:	24683:	24684:	24685:	24686:
U 2AE8	24687:	24688:	24689:	24690:	24691:	24692:	24693:	24694:
U 2AF0	24695:	24696:	24697:	24698:	24699:	24700:	24701:	24702:
U 2AF8	24703:	24704:	24705:	24706:	24707:	24708:	24709:	24710:
U 2B00	24711:	24712:	24713:	24714:	24715:	24716:	24717:	24718:
U 2B08	24719:	24720:	24721:	24722:	24723:	24724:	24725:	24726:
U 2B10	24727:	24728:	24729:	24730:	24731:	24732:	24733:	24734:
U 2B18	24735:	24736:	24737:	24738:	24739:	24740:	24741:	24742:
U 2B20	24743:	24744:	24745:	24746:	24747:	24748:	24749:	24750:
U 2B28	24751:	24752:	24753:	24754:	24755:	24756:	24757:	24758:
U 2B30	24759:	24760:	24761:	24762:	24763:	24764:	24765:	24766:
U 2B38	24767:	24768:	24769:	24770:	24771:	24772:	24773:	24774:
U 2B40	24775:	24776:	24777:	24778:	24779:	24780:	24781:	24782:
U 2B48	24783:	24784:	24785:	24786:	24787:	24788:	24789:	24790:
U 2B50	24791:	24792:	24793:	24794:	24795:	24796:	24797:	24798:
U 2B58	24799:	24800:	24801:	24802:	24803:	24804:	24805:	24806:
U 2B60	24807:	24808:	24809:	24810:	24811:	24812:	24813:	24814:
U 2B68	24815:	24816:	24817:	24818:	24819:	24820:	24821:	24822:
U 2B70	24823:	24824:	24825:	24826:	24827:	24828:	24829:	24830:
U 2B78	24831:	24832:	24833:	24834:	24835:	24836:	24837:	24838:
U 2B80	24839:	24840:	24841:	24842:	24843:	24844:	24845:	24846:
U 2B88	24847:	24848:	24849:	24850:	24851:	24852:	24853:	24854:
U 2B90	24855:	24856:	24857:	24858:	24859:	24860:	24861:	24862:
U 2B98	24863:	24864:	24865:	24866:	24867:	24868:	24869:	24870:
U 23A0	24871:	24872:	24873:	24874:	24875:	24876:	24877:	24878:
U 2BA8	24879:	24880:	24881:	24882:	24883:	24884:	24885:	24886:
U 2BB0	24887:	24888:	24889:	24890:	24891:	24892:	24893:	24894:
U 2BB8	24895:	24896:	24897:	24898:	24899:	24900:	24901:	24902:
U 2BC0	24903:	24904:	24905:	24906:	24907:	24908:	24909:	24910:
U 2BC8	24911:	24912:	24913:	24914:	24915:	24916:	24917:	24918:
U 2BD0	24919:	24920:	24921:	24922:	24923:	24924:	24925:	24926:
U 2BD8	24927:	24928:	24929:	24930:	24931:	24932:	24933:	24934:
U 2BE0	24935:	24936:	24937:	24938:	24939:	24940:	24941:	24942:
U 2BE8	24943:	24944:	24945:	24946:	24947:	24948:	24949:	24950:
U 2BF0	24951:	24952:	24953:	24954:	24955:	24956:	24957:	24958:
U 2BF8	24959:	24960:	24961:	24962:	24963:	24964:	24965:	24966:

Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2C00	24967:	24968:	24969:	24970:	24971:	24972:	24973:	24974:
U 2C08	24975:	24976:	24980:	24981:	24982:	24983:	24984:	24985:
U 2C10	24986:	24987:	24988:	24989:	24990:	24991:	24992:	24993:
U 2C18	24994:	24995:	24996:	24997:	24998:	24999:	25000:	25001:
U 2C20	25002:	25003:	25004:	25005:	25006:	25007:	25011:	25012:
U 2C28	25013:	25014:	25015:	25016:	25017:	25018:	25019:	25020:
U 2C30	25021:	25022:	25023:	25024:	25025:	25026:	25027:	25028:
U 2C38	25029:	25030:	25031:	25032:	25033:	25034:	25035:	25036:
U 2C40	25037:	25038:	25039:	25043:	25044:	25045:	25046:	25047:
U 2C48	25048:	25049:	25050:	25051:	25052:	25053:	25054:	25055:
U 2C50	25056:	25057:	25058:	25059:	25060:	25061:	25062:	25063:
U 2C58	25064:	25065:	25066:	25067:	25068:	25069:	25070:	25071:
U 2C60	25072:	25073:	25074:	25075:	25076:	25077:	25078:	25079:
U 2C68	25080:	25081:	25082:	25084:	25085:	25087:	25088:	25089:
U 2C70	25090:	25091:	25092:	25093:	25094:	25095:	25096:	25097:
U 2C78	25098:	25099:	25100:	25101:	25102:	25103:	25104:	25105:
U 2C80	25106:	25107:	25108:	25109:	25110:	25111:	25112:	25113:
U 2C88	25115:	25116:	25118:	25119:	25120:	25121:	25122:	25123:
U 2C90	25124:	25125:	25126:	25127:	25128:	25129:	25130:	25131:
U 2C98	25132:	25133:	25134:	25135:	25136:	25137:	25138:	25139:
U 2CA0	25140:	25141:	25142:	25143:	25144:	25145:	25146:	25147:
U 2CA8	25148:	25150:	25151:	25153:	25154:	25155:	25156:	25157:
U 2CB0	25158:	25159:	25160:	25161:	25162:	25163:	25164:	25165:
U 2CB8	25166:	25167:	25168:	25169:	25170:	25171:	25172:	25173:
U 2CC0	25174:	25175:	25176:	25177:	25178:	25179:	25180:	25181:
U 2CC8	25182:	25184:	25185:	25187:	25188:	25189:	25190:	25191:
U 2CD0	25192:	25193:	25194:	25195:	25196:	25197:	25198:	25199:
U 2CD8	25200:	25201:	25202:	25203:	25204:	25205:	25206:	25207:
U 2CE0	25208:	25209:	25210:	25211:	25212:	25213:	25214:	25215:
U 2CE8	25216:	25217:	25219:	25220:	25222:	25223:	25224:	25225:
U 2CF0	25226:	25227:	25228:	25229:	25230:	25231:	25232:	25233:
U 2CF8	25234:	25235:	25236:	25237:	25238:	25239:	25240:	25241:
U 2D00	25242:	25243:	25244:	25245:	25246:	25247:	25248:	25250:
U 2D08	25251:	25253:	25254:	25255:	25256:	25257:	25258:	25259:
U 2D10	25261:	25267:	25268:	25269:	25270:	25271:	25277:	25278:
U 2D18	25279:	25280:	25281:	25282:	25283:	25289:	25290:	25291:
U 2D20	25292:	25293:	25294:	25295:	25296:	25297:	25298:	25299:
U 2D28	25300:	25301:	25302:	25303:	25304:	25305:	25306:	25307:
U 2D30	25308:	25309:	25310:	25316:	25317:	25318:	25319:	25320:
U 2D38	25321:	25322:	25323:	25328:	25329:	25330:	25331:	25332:
U 2D40	25333:	25334:	25335:	25336:	25337:	25338:	25339:	25340:
U 2D48	25341:	25342:	25397:	25399:	25402:	25404:	25406:	25408:
U 2D50	25409:	25410:	25411:	25412:	25413:	25414:	25415:	25416:
U 2D58	25417:	25418:	25419:	25420:	25421:	25422:	25423:	25424:
U 2D60	25425:	25427:	25428:	25429:	25431:	25432:	25433:	25435:
U 2D68	25436:	25437:	25438:	25441:	25444:	25445:	25446:	25447:
U 2D70	25448:	25449:	25450:	25451:	25452:	25453:	25454:	25455:
U 2D78	25456:	25457:	25458:	25459:	25461:	25463:	25464:	25467:
U 2D80	25468:	25469:	25470:	25471:	25472:	25473:	25474:	25475:
U 2D88	25476:	25477:	25478:	25480:	25481:	25482:	25484:	25485:
U 2D90	25486:	25487:	25488:	25489:	25491:	25492:	25493:	25494:
U 2D98	25495:	25496:	25497:	25498:	25499:	25500:	25501:	25502:
U 2DA0	25503:	25504:	25505:	25506:	25507:	25508:	25509:	25510:

;
Location / Line Number Index

;Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
U 2DA8	25511:	25512:	25513:	25515:	25516:	25518:	25520:	25521:
U 2DB0	25522:	25523:	25524:	25525:	25526:	25527:	25528:	25530:
U 2DB8	25531:	25533:	25534:	25535:	25537:	25538:	25540:	25541:
U 2DC0	25544:	25546:	25547:	25550:	25551:	25552:	25553:	25554:
U 2DC8	25555:	25556.	25557:	25558:	25559:	25560:	25561:	25562:
U 2DD0	25563:	25564:	25565:	25567:	25569:	25570:	25573:	25574:
U 2DD8	25575:	25576:	25577:	25578:	25579:	25580:	25581:	25582:
U 2DE0	25583:	25584:	25586:	25587:	25588:	25590:	25591:	25592:
U 2DE8	25593:	25594:	25595:	25597:	25598:	25599:	25600:	25601:
U 2DF0	25602:	25603:	25604:	25605:	25606:	25607:	25608:	25609:
U 2DF8	25610:	25611:	25612:	25613:	25614:	25615:	25616:	25617:
U 2E00	25618:	25619:	25622:	25624:	25626:	25627:	25628:	25629:
U 2E08	25637:	25638:	25639:	25641:	25642:	25643:	25644:	25645:
U 2E10	25646:	25647:	25649:	25650:	25651:	25652:	25653:	25655:
U 2E18	25656:	25657:	25658:	25659:	25660:	25661:	25662:	25663:
U 2E20	25736:	25738:	25741:	25743:	25745:	25747:	25748:	25749:
U 2E28	25750:	25751:	25752:	25753:	25754:	25755:	25756:	25757:
U 2E30	25758:	25759:	25760:	25761:	25762:	25763:	25765:	25767:
U 2E38	25768:	25769:	25770:	25771:	25772:	25774:	25775:	25777:
U 2E40	25778:	25780:	25781:	25782:	25783:	25784:	25785:	25786:
U 2E48	25789:	25790:	25791:	25792:	25793:	25796:	25797:	25798:
U 2E50	25799:	25802:	25804:	25805:	25806:	25807:	25808:	25809:
U 2E58	25810:	25811:	25812:	25813:	25814:	25815:	25816:	25817:
U 2E60	25818:	25819:	25820:	25821:	25822:	25824:	25825:	25827:
U 2E68	25828:	25830:	25831:	25832:	25833:	25834:	25835:	25836:
U 2E70	25837:	25838:	25841:	25842:	25843:	25844:	25845:	25846:
U 2E78	25849:	25850:	25851:	25852:	25855:	25857:	25858:	25859:
U 2E80	25860:	25861:	25862:	25863:	25864:	25865:	25866:	25867:
U 2E88	25868:	25869:	25870:	25871:	25872:	25873:	25874:	25875:
U 2E90	25877:	25878:	25880:	25881:	25883:	25884:	25885:	25886:
U 2E98	25887:	25888:	25889:	25890:	25893:	25894:	25895:	25896:
U 2EA0	25897:	25898:	25901:	25902:	25903:	25904:	25907:	25909:
U 2EA8	25910:	25911:	25912:	25913:	25914:	25915:	25916:	25917:
U 2EB0	25918:	25919:	25920:	25921:	25922:	25923:	25924:	25925:
U 2EB8	25926:	25927:	25928:	25929:	25930:	25931:	25932:	25933:
U 2EC0	25934:	25935:	25936:	25937:	25938:	25939:	25940:	25941:
U 2EC8	25942:	25944:	25945:	25947:	25948:	25949:	25950:	25951:
U 2ED0	25952:	25953:	25954:	25955:	25956:	25957:	25958:	25959:
U 2ED8	25960:	25961:	25962:	25963:	25964:	25966:	25967:	25968:
U 2EE0	25969:	25971:	25972:	25974:	25976:			

C-code Microword Summary

	Words not in bounds
ENKCE	512
Used	512
Remaining	

Total microwords used in memory C: 512
Total microwords remaining in memory C: 0
Highest address used in memory C: 180 (hex)

;
; U-code Microword Summary

	Words not in bounds
ENKCE	11754
Used	11754
Remaining	

Total microwords used in memory U: 11754
Total microwords remaining in memory U: 0
Highest address used in memory U: 2EE4 (hex)

; The file used in this assembly is:
ENKCE.MIC

Pass 1 warnings: 0 Pass 2 warnings: 0
Pass 1 errors: 0 Pass 2 errors: 0