

4-	58	"COMMON DEFINITIONS
4-	58	" MNEUMONIC DEFINITIONS
4-	58	" GLOBAL MACRO CALLS
4-	58	" HARDWARE TEST STREAM MACRO CALLS
4-	58	" CMPCA AND CMPCAM MODE DEFINITIONS
4-	58	" SWITCH (SWR) REGISTER BIT DEFINITIONS
4-	58	" SWITCH REGISTER 1 (SWR1) BIT DEFINITIONS
4-	58	" CONSOLE ADAPTER REGISTER DEFINITIONS
4-	58	" ID BUS REGISTER DEFINITIONS
4-	58	" LSI-11 VECTOR DEFINITIONS
4-	58	" MISCELLANEOUS DEFINITIONS
4-	58	" MODULE AND BUS NAME ASSIGNMENTS
4-	58	" LSI-11 REGISTER NAME ASSIGNMENTS
4-	58	" FILE NAME CODES
4-	58	" CONSOLE ROUTINE ERROR CODES AND DEFINITIONS
5-	62	"GLOBAL TAGS
8-	69	"HARDWARE MONITOR COMMON TAGS
8-	129	"THE DISPATCH TABLE TO THE EXECUTE SUBROUTINES
9-	237	"HARDWARE MONITOR SUBROUTINES
9-	244	" TYPE ERROR DATA SUBROUTINE
9-	285	" READ V BUS SUBROUTINE
9-	306	" EXPECTED TRAP ROUTINE
9-	323	" UNEXPECTED TRAP ROUTINE
9-	346	" UNEXPECTED INTERRUPT ROUTINE
9-	363	" TYPE PROGRAM NAME AND VERSION
9-	380	" SINGLE INSTRUCTION THE HARDWARE ROUTINE
10-	433	"PROGRAM INITIALIZATION
10-	498	"TEST STREAM INTERPRETER
11-	537	" BLOCK MIC SUBROUTINE
11-	587	" CHECK POINT SUBROUTINE
11-	625	" CLOCK SUBROUTINE
11-	642	" COMPARE CONSOLE ADAPTER REGISTER SUBROUTINE
11-	763	" COMPARE PC SAVE SUBROUTINE
11-	794	" END HARDWARE SUBROUTINE
11-	815	" ENDOLOOP SUBROUTINE
11-	848	" END OVERLAY SUBROUTINE
11-	876	" ERROR LOOP SUBROUTINE
11-	890	" FETCH SUBROUTINE
11-	921	" FLOAT ONE SUBROUTINE
11-	946	" FLOAT ZERO SUBROUTINE
11-	972	" IF ERROR SUBROUTINE
11-	1035	" INITIALIZE SUBROUTINE
11-	1049	" KMUX GENERATE SUBROUTINE
11-	1075	" LOAD CONSOLE ADAPTER REGISTER SUBROUTINE
11-	1100	" LOAD ID REGISTER SUBROUTINE
11-	1130	" LOOP SUBROUTINE
11-	1205	" MASK SUBROUTINE
11-	1220	" MOVE SUBROUTINE
11-	1247	" NEW TEST SUBROUTINE
11-	1327	" NOP SUBROUTINE
11-	1337	" READ ID BUS SUBROUTINE
11-	1348	" REPORT SUBROUTINE
11-	1387	" RESET SUBROUTINE
11-	1399	" SET PSW SUBROUTINE
11-	1411	" SET VECTOR ROUTINE
11-	1423	" SKIP SUBROUTINE
11-	1435	" SKIP IF ERROR SUBROUTINE

Table of contents

- 11- 1453 " SP ADDRESS GENERATE SUBROUTINE
- 11- 1477 " SUBTEST SUBROUTINE
- 11- 1498 " TEST V BUS SUBROUTINE
- 11- 1577 " TYPE WCS SIZE SUBROUTINE

1 000001 LSTFIL=1

```
.TITLE VAX 11/780 MICRO DIAGNOSTIC HARDCORE MONITOR
2  .IDENT /V13.3/
3
4  ;
5  ; COPYRIGHT (C) 1977,1984
6  ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
7  ;
8  ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
9  ; COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
10 ; ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
11 ; MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
12 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
13 ; TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
14 ; REMAIN IN DEC.
15 ;
16 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
17 ; AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
18 ; CORPORATION.
19 ;
20 ; DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
21 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
22 ;
23 ;
24 ;
25 ;++
26 ; FACILITY: Micro Diagnostic Monitor
27 ;
28 ; FUNCTIONAL DESCRIPTION:
29 ;
30 ; This module controls the loading and execution of the Hardcore
31 ; Test Stream.
32 ;
33 ; ENVIRONMENT: Micro Diagnostic Monitor
34 ;
35 ; AUTHOR: Donald W. Monroe, CREATION DATE: 11-Oct-1977
36 ;
37 ; MODIFIED BY:
38 ; 12.1 Don Monroe March 1981
39 ; Added call to 'CHKKEY' function in console.
40 ; 13.0 Don Monroe April 1981
41 ; Added routine to type file name and version.
42 ; 13.1 Don Monroe September 1981
43 ; Moved single instruction routine from monitor to
44 ; this module and added an ENABLE CONTROL C function
45 ; call to the monitor.
46 ; 13.2 Dave Shull December 1982
47 ; Fixed problem where on M8238 (2K of WCS) was installed the
48 ; hardccre diagnostic would only test the first 1K of the 2K
49 ; module.
50 ; 13.3 Dave Shull November 1984
51 ; Change 4megabyte module callout from 72 to 74
52 ;--
```

```

54 .LIST MC,ME
55 .NLIST MD,CND
56 .MCALL EQUATE,CHKKEY
57
58 000000 EQUATE HARDCORE

```

```

.SBTTL "COMMON DEFINITIONS
.SBTTL " MNEUMONIC DEFINITIONS

```

```

;*****

```

```

;+
; FOLLOWING ARE COMMON DEFINITIONS OF MNEUMONICS USED BY ALL OF THE
; PROGRAMS THAT EXECUTE OUT OF THE LSI-11.
;-

```

```

.SBTTL " GLOBAL MACRO CALLS

```

```

;*****

```

```

;+
; THE FOLLOWING ".MCALL'S" ARE GLOBAL MACRO ASSIGNMENTS. THESE MACRO'S ARE
; USED BY ALL 4 MONITORS, THE PARSER, AND THE DIRECTORY FILE.

```

```

; SOME OF THESE MACRO'S ARE DEFINED IN THE CONSOLE MACRO PACKAGE "STRMAC"
; THEREFORE, THAT MACRO FILE MUST BE MERGED WITH THIS MACRO FILE BEFORE
; ASSEMBLY.
;-

```

```

;*****

```

```

.MCALL T$INIT,T$WRIT,T$READ,F$OPEN,F$READ,LOADCO,CONVERT,CONABORT
.MCALL LDCNSL,GETMDM,ENCTRLC
.MCALL STARS,COMTAGS,OPENFILE,READOVR,RETURN,RESET$,ASSEMBLE
.MCALL DONE,RDIDREG,SBCCLOCK,DONEM,FILL,CALLFAILCHAIN,$CODDF
.MCALL MES,TYPES,TYPED,TYPE,TYEMOD,RINGBELL,GETUPC,TYPESECTNO
.MCALL TYPEERR,CALLMICMON,LOADWCS,STSCLOCK,LOADID,MTPS,MFPS,TYPEB

```

```

.SBTTL " HARDCORE TEST STREAM MACRO CALLS

```

```

;*****

```

```

;+
; THE FOLLOWING MACRO CALLS ARE USED EXCLUSIVELY BY THE HARDCORE TEST
; STREAM. THEY ARE ONLY ASSEMBLED IF THE ARGUMENT TO THE "EQUATE"
; MACRO IS NON BLANK
;-

```

```

;*****

```

```

.MCALL NEWTST,$$NEWTST,NEWOVR,FORCEOVR,INITIALIZE,MASK,HEXTST,DUMMY
.MCALL LOOP,$$LOOP,ENDLOOP,ERLOOP,IFERROR,CMPCA,CMPCAM,$$ERRLOOP
.MCALL NOOP,READVB,BLKMIC,CLOCK,TSTVB,LDIDREG,SKIP,SPAGEN,$$SKIP
.MCALL $$$SK,TESTDAT,ENDDAT,CMPPCSV,$$ENDDAT,HEX3,HEX33,MOVE
.MCALL CMPCAD,CMPCMD,READID,CHKPNT,REPORT,FLTONE,FLTZRO,KMXGEN
.MCALL LOADCA,ENDHC,$$ENDHC,FETCH,$$REPORT,VBUSG,$$FETCH
.MCALL HEX1,HEX11,HEX2,HEX22,HEX4,HEX44,HEX5,HEX55,HEX6,HEX66
.MCALL SETVEC,SUBTEST,RESETC,SETPSW,FILL,$DIR,TYPSIZE,SKIPERROR

```

```

000001 $TN=1 ; INIT THE TEST NUMBER FOR THE TEST STREAM
000001 $SN=1 ; INIT THE SECTION NUMBER FOR THE TEST STREAM
000000 $SECTOR=0 ; INIT THE RELATIVE SECTOR NUMBER FOR
      THE DIRECTORY OF THE TEST STREAM

```

```

.SBTTL " CMPCA AND CMPCAM MODE DEFINITIONS

```

```

;*****

```

```

;+
; FOLLOWING ARE THE TWO MODE DEFINITIONS FOR THE "CMPCA", "CMPCAD",
; "CMPCAM", AND "CMPCMD" PSEUDO INSTRUCTIONS.
;
; IF MORE MODES ARE REQUIRED, THE BRANCH TABLE (IN THE HARDWARE MONITOR)
; WILL HAVE TO HAVE MORE ENTRIES PUT IN IT.
;-

```

```

*****
000000    EQ.= 0    ; BEQ
000004    NE.= 4    ; BNE

```

```

;
; SBTTL " SWITCH (SWR) REGISTER BIT DEFINITIONS
;*****

```

```

;+
; FOLLOWING ARE THE DEFINITIONS OF THE 16 BITS OF THE SOFTWARE SWITCH
; REGISTER. BITS<5:0> ARE READ WRITE BY COMMAND FROM THE MICRO MONITOR.
; BITS 7 AND 6 ARE READ AND CLEAR ONLY FROM THE MICRO MONITOR.
;

```

```

; ALL OTHER BITS ARE TRANSPARENT TO THE OPERATOR.
;-

```

```

*****
000001    HALTD= 1    ; HALT ON ERROR DETECTION
000002    HALTI= 2    ; HALT ON ERROR ISOLATION
000004    LOOP= 4     ; LOOP ON ERROR
000010    NER= 10    ; NO ERROR REPORT
000020    BELL= 20    ; BELL ON EVERY 6 ERRORS
000040    ERABT= 40   ; GO TO NEXT TEST AFTER ERROR
000100    LOSS= 100   ; LOOP ON SPECIAL SECTION
000200    LOST= 200   ; LOOP ON SPECIAL TEST
000400    SINST= 400  ; HARDWARE SINGLE INSTRUCTION FLAG
001000    FLPY2= 1000 ; MA780 FLOPPY (MOUNTED) FLAG
002000    FLPY3= 2000 ; FLOPPY 2 (MOUNTED) FLAG
003000    FLPY4= 3000 ; MS780-E FLOPPY (MOUNTED) FLAG
003000    FLPYMSK=3000 ; MASK FOR FLOPPY FIELD
004000    CONT= 4000  ; CONTINUE FLAG
010000    KEYQUE= 10000 ; KEYBOARD ILLEGAL CHARACTER
020000    KEYERR= 20000 ; KEYBOARD ERROR FLAG
040000    CTRLC= 40000 ; CONTROL C FLAG
100000    COM= 100000 ; COMMAND MODE FLAG

```

```

;
; SBTTL " SWITCH REGISTER 1 (SWR1) BIT DEFINITIONS
;*****

```

```

;+
; FOLLOWING ARE THE BIT DEFINITIONS OF SOFTWARE SWITCH REGISTER 1 (SWR1).
; THESE BITS ARE ALL TRANSPARENT TO THE OPERATOR.
;-

```

```

*****
000001    HARDC= 1    ; HARDWARE (EXECUTING) FLAG
000002    RUNFLG= 2    ; DIAGNOSE COMMAND HAS BEEN USED
000004    B1FULL= 4    ; BUFFER 1 FULL FLAG. USED IN THE
; DOUBLE BUFFERED ROUTINE IN THE GO CHAIN
000010    CLKFST= 10   ; SET CLOCK FAST FLAG. SET OR CLEARED BY THE OPERATOR
000020    CLKSLO= 20   ; SET CLOCK SLOW FLAG.

```

" SWITCH REGISTER 1 (SWR1) BIT DEFINITIONS

```

000040 B2INUSE=40 ; BUFFER 2 IN USE FLAG. USED BY THE DOUBLE
; BUFFER ROUTINE IN THE GO CHAIN
000100 DIRERR= 100 ; DIRECTORY ERROR FLAG. SET BY THE "DIRECTORY"
; PROGRAM IF AN ERROR WAS DETECTED
000200 B2FULL= 200 ; BUFFER 2 FULL FLAG. USED BY THE DOUBLE
; BUFFER ROUTINE IN THE GO CHAIN
000400 B1INUSE=400 ; BUFFER 1 IN USE FLAG. USED BY THE DOUBLE
; BUFFER ROUTINE IN THE GO CHAIN
001000 DICMD= 1000 ; DIAGNOSE COMMAND FLAG. SET WHEN A
; "DIAGNOSE" COMMAND IS USED
002000 MIC1FL= 2000 ; GO CHAIN FILE # 1 FLAG. USED BY THE
; DIRECTORY SEARCH PROGRAM
004000 MIC2FL= 4000 ; GO CHAIN FILE # 2 FLAG. USED BY THE
; DIRECTORY SEARCH PROGRAM.
010000 FPA= 10000 ; FPA PRESENT FLAG. SET BY THE GO CHAIN
; MONITOR OR THE COMMAND PARSER.
020000 TSTSPAN=20000 ; A TEST SPAN HAS BEEN SPECIFIED
040000 SCTSPAN=40000 ; A SECTION SPAN HAS BEEN SPECIFIED

```

.SBTTL " CONSOLE ADAPTER REGISTER DEFINITIONS

\*\*\*\*\*8

THE FOLLOWING ARE THE ADDRESS ASSIGNMENTS AND THE BIT DEFINITIONS OF THE CONSOLE ADAPTER REGISTERS.

\*\*\*\*\*

```

173000 ROM0= 173000 ; ROM LOCATION 0
173002 ROM1= 173002 ; ROM LOCATION 2
173004 SPARE1= 173004
173006 IDDATLO=173006 ; LOW 16 BITS OF ID DATA REGISTER
173010 IDDATAHI=173010 ; HIGH 16 BITS OF ID DATA REGISTER
173012 SPARE2= 173012
173014 RXDNE=173014 ; RECEIVER CONTROL AND STATUS REGISTER
173016 TXRDY= 173016 ; TRANSMITTER CONTROL AND STATUS REGISTER
173020 TOIDLO= 173020 ; LO 16 BITS OF TRANSMITTER DATA BUFFER
173022 TOIDHI= 173022 ; HIGH 16 BITS OF TRANSMITTER DATA BUFFER
173024 FMIDLO= 173024 ; LO 16 BITS OF RECEIVER DATA BUFFER
173026 FMIDHI= 173026 ; HIGH 16 BITS OF RECEIVER DATA BUFFER
173030 IDCS= 173030 ; ID BUS CONTROL AND STATUS REGISTER
000200 IDMAINT=200 ; ID MAINTENANCE BIT
000100 IDWRITE=100 ; ID BUS WRITE BIT
100000 IDCYCLE=100000 ; ID BUS CYCLE BIT
173032 CONMCR= 173032 ; MACHINE CONTROL REGISTER
010000 INIT= 10000 ; CPU INITIALIZE BIT
002000 MNTRTN=2000 ; MAINTENANCE RETURN ENABLE BIT
001000 UPC12=1000 ; FORCE UPC 12 BIT
000200 CLRWRD=200 ; ROM NOP BIT
000100 SOMM=100 ; STOP ON MICRO MATCH ENABLE BIT
000040 CLKSTPD=40 ; CLOCK STOPPED BIT
000020 FR1= 20 ; CLOCK FREQUENCY SELECT BIT 1
000010 FR0= 10 ; CLOCK FREQUENCY SELECT BIT 0
000004 STS= 4 ; ENABLE SINGLE TIME STATE BIT
000002 SBC= 2 ; ENABLE SINGLE BUS CYCLE BIT
000001 PROCEED=1 ; CLOCK PROCEED BIT
173034 CONMCS= 173034 ; MACHINE CONTROL AND STATUS REGISTER

```

" CONSOLE ADAPTER REGISTER DEFINITIONS

```

010000    FLPYON=10000 ; FLOPPY ON BIT
001000    CONCM=1000 ; CONSOLE COMMAND MODE BIT
000400    CPURUN=400 ; CPU RUN BIT
000200    CONACK=200 ; CONSOLE ACKNOWLEDGE BIT
000100    RDYIE=100 ; RECEIVER INTERRUPT ENABLE BIT
000040    DNEIE=40 ; TRANSMITTER INTERRUPT ENABLE
:73036    VBCTRL= 173036 ; V BUS CONTROL REGISTER
000200    CCPT0=200 ; TIME STATE CPT0 BIT
000100    CCPT1=100 ; TIME STATE CPT1 BIT
000040    CCPT2=40 ; TIME STATE CPT2 BIT
000020    CCPT3=20 ; TIME STATE CPT2 BIT
000004    SLFTST=4 ; V BUS SELF TEST BIT
000002    VBLOAD=2 ; V BUS LOAD BIT
000001    VBCLK=1 ; V BUS CLOCK BIT

```

:SBTTL " ID BUS REGISTER DEFINITIONS

:\*\*\*\*\*

:+ FOLLOWING ARE THE MNEUMONICS ASSIGNED TO THE ID BUS REGISTERS.

:\*\*\*\*\*

```

000000    IBDAT= 00 ; IBUF DATA
000001    IBTOD= 01 ; TIME OF DAY CLOCK
;
000003    CONID= 03 ; SYSTEM ID
000004    CONRXS= 04 ; CONSOLE RXCS
000005    CONRXD= 05 ; CONSOLE RXDB
000006    CONTXS= 06 ; CONSOLE TXCS
000007    CONTXD= 07 ; CONSOLE TXDB
000010    RWDQ= 10 ; WRITE Q REG, READ D REG
000011    IBNIN= 11 ; NEXT INTERVAL REGISTER
000012    IBCLKS= 12 ; CLOCK CONTROL AND STATUS
000013    IBICT= 13 ; IBUF INTERVAL COUNT
000014    CES= 14
000015    VECT= 15
000016    SIR= 16
000017    PSL= 17
000020    TBDAT= 20 ; TBUF DATA
;
000022    TBER0= 22 ; TBUF ERROR REG 0
000023    TBER1= 23 ; TBUF ERROR REG 1
000024    ACC0= 24 ; ACCELERATOR REG 0
000025    ACC1= 25 ; ACCELERATOR REG 1
000026    ACCMNT= 26 ; ACCELERATOR MAINTENANCE REG
000027    ACCST= 27 ; ACCELERATOR STATUS REGISTER
000030    SBISILO=30 ; SBI SILO
000031    SBIERR= 31 ; SBI ERROR REG
000032    SBITO= 32 ; SBI TIMEOUT ADDRESS
000033    SBIFLT= 33 ; SBI FAULT/STATUS
000034    SBISCM= 34 ; SBI SILO COMAPRATOR
000035    SBIMAT= 35 ; SBI MAINTENANCE
000036    SBICP= 36 ; SBI CACHE PARITY
;
000040    USCSTK=40 ; SEQUENCER MICRO STACK
000041    USCBRK=41 ; SEQUENCER MICRO BREAK
000042    USCADR=42 ; SEQUENCER WCS ADDRESS

```



" ID BUS REGISTER DEFINITIONS

000043 USCDAT=43 ; SEQUENCER WCS DATA
; THE FOLLOWING REGISTERS ARE THE TEMP A AND TEMPB REGISTERS

- 000044 POBR= 44 ;
000045 P1BR= 45
000046 SBR= 46
000050 KSP= 50
000051 ESP= 51
000052 SSP= 52
000053 USP= 53
000054 ISP= 54
000055 FPDA= 55
000056 D.SV= 56
000057 Q.SV= 57
000060 TEMP0= 60
000061 TEMP1= 61
000062 TEMP2= 62
000063 TEMP3= 63
000064 TEMP4= 64
000065 TEMP5= 65
000066 TEMP6= 66
000067 TEMP7= 67
000070 TEMP8= 70
000071 TEMP9= 71
000072 PCBB= 72
000073 SCBB= 73
000074 POLR= 74
000075 P1LR= 75
000076 SLR= 76

.SBTTL " LSI-11 VECTOR DEFINITIONS
\*\*\*\*\*

; THE FOLLOWING MNEUMONICS ARE THE DEFINITIONS FOR THE LSI-11 TRAP
; AND INTERRUPT VECTORS.

- 00J034 TRAPVEC=34 ; "TRAP" INSTRUCTION VECTOR
J00300 TXVEC= 300 ; TRANSMITTER INTERRUPT VECTOR
000304 RXVEC= 304 ; RECEIVER INTERRUPT VECTOR

.SBTTL " MISCELLANEOUS DEFINITIONS
\*\*\*\*\*

; FOLLOWING ARE SOME MISCELLANEOUS DEFINITIONS USED IN THE HARDWARE TESTS.

- 177777 IDREGLO=-1 ; USED AFTER A "READID" PSEUDO INSTRUCTION TO SPECIFY
; THE CONTENTS OF LOCATION "IDDATLO"
; AS THE ARGUMENT
000001 IDREGHI=1 ; USED AFTER A "READID" PSEUDO INSTRUCTION
; TO SPECIFY THE CONTENTS OF LOCATION
; "IDDATHI" AS THE ARGUMENT

```

000034      TPCINIT=34 ; FIRST ADDRESS (RELATIVE) OF EACH TEST
; STREAM OVERLAY.
; NOTE: IF THE LENGTH OF THE DISPATCH
; TABLE IS CHANGED, THIS DEFINITION
; MUST ALSO BE CHANGED.
000004      ITSTPTR=4 ; FIRST ADDRESS (RELATIVE) OF THE TEST TABLE
; IN EACH TEST STREAM OVERLAY.

```

```

000010      RADOCT= 10 ; RADIX OCTAL CODE FOR CONSOLE CONVERT ROUTINE
000020      RADHEX= 20 ; RADIX HEX CODE FOR CONSOLE CONVERT ROUTINE

```

```

;
;
;SBTTL " MODULE AND BUS NAME ASSIGNMENTS
;*****
;+
; THE FOLLOWING DEFINITIONS ARE USED BY THE "TYPMOD" ROUTINE IN THE
; MICRO DIAGNOSTIC MUNITOR.
;-
;*****
;

```

```

000000      CIB= 0
000001      USC= 1
000002      WCS= 2
000003      PCS= 3
000004      DAP= 4
000010      DBP= 10
000005      DCP= 5
000006      DDP= 6
000007      DEP= 7
000011      CEH= 11
000012      ICL= 12
000013      CAM= 13
000014      CDM= 14
000015      TBM= 15
000016      SBL= 16
000017      SBH= 17
000020      IRC= 20
000021      IDP= 21
000022      MSB= 22
000023      MCN= 23
000024      MDT= 24
000025      MAY= 25
000026      CLK= 26
000027      TRS= 27
000030      FNM= 30
000031      FMH= 31
000032      FML= 32
000033      FAD= 33
000034      FCT= 34
000035      MAY5= 35 ; MAY WITH 16K CHIP
000036      MPI= 36
000037      MPC= 37
000040      MPS= 40
000041      MAT= 41
000042      WCS2K= 42 ; 2K WCS MODULE
000043      MSBE= 43 ; MSB for MA780-E
000044      BYL= 44 ; Lower controller

```

" MODULE AND BUS NAME ASSIGNMENTS

```

000045    BYU= 45 ; Upper controller
000046    MAY4= 46 ; 1 Megabyte Array
000047    MAY8= 47 ; 4 Megabyte Array

```

; START OF BUS NAMES

```

000050    CSBUS= 50
000051    IDBUS= 51
000052    VBUS= 52

```

; START OF ADAPTER NAMES

```

000053    UBA= 53
000054    MBA= 54
000055    DRA= 55
000056    CIA= 56

```

```

; THE FOLLOWING OFFSET DEFINITIONS ARE OFFSETS INTO THE RAD50 LIST FOR
; THE ABOVE MODULE NAMES. IF THE LIST IS CHANGED, THESE OFFSETS MUST BE
; CHANGED. THESE OFFSETS ARE USED BY THE TYPE MODULE ROUTINE IN ESKAR.

```

```

000120    BUSOFF= CSBUS * 2
000052    M4KOFF= MAY * 2
000072    M6KOFF= MAY6 * 2
000114    M64KOF= MAY4 * 2
000116    M256KO= MAY8 * 2
000126    ADAOFF= UBA * 2

```

.SBTTL " LSI-11 REGISTER NAME ASSIGNMENTS

```

000000    R0= %0
000001    R1= %1
000002    R2= %2
000003    R3= %3
000004    R4= %4
000005    R5= %5
000006    R6= %6
000007    R7= %7
000006    SP= %6
000007    PC= %7

```

.SBTTL " FILE NAME CODES

\*\*\*\*\*

```

; THE FOLLOWING CODES ARE USED BY THE "OPEN FILE" ROUTINE IN THE
; MICRO DIAGNOSTIC MONITOR.

```

\*\*\*\*\*

```

000000    HCMONITOR=0 ; HARDCORE MONITOR
000002    TESTSTREAM=2 ; HARDCORE TEST STREAM
000004    GOCHAINMONITOR=4 ; GO CHAIN MONITOR
000006    GOCHA1=6 ; GO CHAIN FILE NUMBER 1 (FLOPPY 1)
000010    PARSER=10 ; MICRO DIAGNOSTIC PARSER
000012    GOCHA2=12 ; GO CHAIN FILE NUMBER 2 (FLOPPY 2)

```

" FILE NAME CODES

```

000014 DIRECTORY=14 ; DIRECTORY SEARCH FILE
000016 FAILCHAINMONITOR=16 ; FAIL CHAIN MONITOR
000020 FCHAI1=20 ; FAIL CHAIN FILE NUMBER 1 (FLOPPY 1)
000022 FCHAI2=22 ; FAIL CHAIN FILE NUMBER 2 (FLOPPY 2)
000024 MPGOCH=24 ; MA780 GO CHAIN
000026 MPFC=26 ; MA780 FAIL CHAIN
000030 MSGOCH=30 ; MS780-E GO CHAIN
000032 MSFC=32 ; MS780-E FAIL CHAIN

```

SBTTL " CONSOLE ROUTINE ERROR CODES AND DEFINITIONS

\*\*\*\*\*

```

; THE FOLLOWING ARE ERROR CODE DEFINITIONS AND EMT DEFINITIONS DEFINED
; BY MIKE HARE THAT ARE USED TO COMMUNICATE WITH THE CONSOLE ROUTINES.

```

\*\*\*\*\*

000000 ;CODDF

;FLOPPY AND TERMINAL ERROR CODES

```

000001 $FER=1 ;FLOPPY HARDWARE ERROR
000002 $FNF=2 ;FILE NOT FOUND
000003 $FNR=3 ;FLOPPY QUEUE FULL
000004 $FOR=4 ;FLOPPY SECTOR # OUT OF LEGAL RANGE
000005 $TBSY=5 ;NO NODE FOR REQUEST
000006 $TCTC=6 ;CONTROL-C INPUTTED
000007 $TER=7 ;TERMINAL HARDWARE DETECTED ERROR

```

;USER SERVICE EMT CODE DEFINITIONS

;THESE CODES MUST BE IN SYNC WITH THE EMT SERVICE MODULE

```

000000 TINIT=0
000001 TWRITE=1
000002 TREAD=2
000003 OPENFL=3
000004 READSC=4
000005 WRITSC=5
000006 LOADCN=6
000007 CNVERT=7
000010 RADGET=10
000011 OFNFL1=11
000012 TYF1=12
000013 TYF2=13
000014 LCCANWC=14
000015 RMWRON=15
000016 LCWRON=16
000017 TMERTR=17
000020 R$SET=20
000021 LDCONS=21
000022 MDMTYP=22
000023 CHKSWI=23
000024 TSTMFG=24

```

60 000J00 .BLKB 6370  
62 006370 COMTAGS  
.SBTTL "GLOBAL TAGS

\*\*\*\*\*  
; THE FOLLOWING 128 BYTES ARE THE GLOBAL TAGS USED BY ALL THE MONITORS.  
; THESE TAGS MUST BE LOCATED AT THE END OF THE MICRO DIAGNOSTIC MONITOR  
; AND AT THE BEGINNING OF ALL THE OTHER MONITORS OR FILES THAT USE THESE  
; TAGS.  
; ONCE THE MICRO DIAGNOSTIC MONITOR IS LOADED INTO MEMORY, THESE TAGS ARE  
; NEVER OVERLAYED.  
; THESE TAGS MUST BE EXACTLY 128 BYTES IN LENGTH.  
\*\*\*\*\*

006370 000000 \$PASS: .WORD 0 ; CONTAINS THE CURRENT PASS COUNT  
006372 000000 \$TSTNM: .WORD 0 ; CONTAINS THE CURRENT TEST NUMBER  
006374 000000 ENDSPAN: .WORD 0 ; ENDING TEST OR SECTION NUMBER OF SPAN  
006376 000000 TESNO: .WORD 0 ; CONTAINS THE TEST NUMBER FOR LOST  
006400 000000 SUBTST: .WORD ; CONTAINS THE CURRENT SUBTEST NUMBER  
006402 000000 \$SCTNO: .WORD 0 ; CONTAINS THE CURRENT SECTION NUMBER  
006404 000001 SECTNO: .WORD 1 ; CONTAINS THE SECTION NUMBER FOR LOSS  
006406 000000 \$ERFLG: .WORD 0 ; IS NON ZERO IF AN ERROR HAS BEEN DETECTED  
; IN THE CURRENT TEST  
006410 000000 \$LPADR: .WORD 0 ; CONTAINS THE LOOP ADDRESS  
006412 000000 \$LPERR: .WORD 0 ; CONTAINS THE ERROR LOOP ADDRESS  
006414 000000 \$ERRPC: .WORD 0 ; CONTAINS THE PC OF THE ERROR CALL  
006416 000000 GOODDAT: .WORD 0 ; CONTAINS THE GOOD DATA OF A TEST  
006420 000000 .WORD 0  
006422 000000 BADDAT: .WORD 0 ; CONTAINS THE BAD DATA OF A TEST  
006424 000000 .WORD 0  
006426 000002 SWR: .WORD 2 ; CONTAINS THE CURRENT VALUE OF THE FLAGS  
006430 000000 SWR1: .WORD 0  
006432 000000 TPC: .WORD ; TEST PC FOR HARDWARE TESTS  
006434 020400 RELOC: .WORD END ; END ADDRESS OF HARDWARE  
006436 000000 FILPTR: .WORD ; INDEX FOR RAD50 FILE NAME  
006440 000000 OVRADR: .WORD ; START ADR FOR READ OVERLAY  
006442 000000 OVRBYT: .WORD ; BYTE COUNT FOR READ OVR  
006444 000000 TYPADR: .WORD ; ADDRESS OF DATA FOR TYPE CALLS  
006446 000000 MODADR: .WORD ; ADR OF MODULE STRING  
006450 000000 SRCADR: .WORD ; ADR OF DATA FOR LOAD WCS  
006452 000000 WCSADR: .WORD ; ADR OF WCS FOR LOAD WCS  
006454 000000 WCSCNT: .WORD ; WORD COUNT FOR LOAD WCS  
006456 000000 STSCNT: .WORD ; STS COUNT  
006460 000000 IDADR: .WORD ; DATA POINTER FOR LOAD ID  
006462 000000 IDADR: .WORD ; ADDRESS OF ID REG  
006464 000000 RDIDLC: .WORD ; LG 16 BITS OF READ ID DATA  
006466 000000 RDIDHI: .WORD ; HI 16 BITS  
006470 000000 GOTUPC: .WORD ; RECEIVED UPC FOR GETUPC  
006472 002 015 012 \$CRLF: .BYTE 2,15,12,0 ; ASCII FOR A "CRLF"  
006475 000  
006476 COMSPC: MES <, >,NB  
006502 MSGA: MES <DATA: >,NB  
006510 MSGB: MES <TRACE: >,NB  
006516 MSGC: MES <?KEYBOARD ERROR: >,NB

## "GLOBAL TAGS

```

006534      SIXSPC: MES <      >
006542 000000      KEYCODE: .WORD
006544 000000      $PSW: .WORD
006546 000001      PASCNT: .WORD 1 ; USER SET PASS COUNT
006550 000000      FPYVEC: .WORD ; FLOPPY INTERRUPT VECTOR
006552 000000      LOSLNK: .WORD 0 ; THIS LOCATION IS DEFINED IN THE MICRO
; DIAGNOSTIC MONITOR TO BE THE ADDRESS
; OF THE MICRO DIAGNOSTIC MONITOR LOCAL
; TAGS. IT IS USED BY THE "DIRECTORY"
; PROGRAM.
006554 000000      LOSSEC: .WORD 0
006556 000000      SECTOR: .WORD 0
006560 000000      FPSYNC: .WORD 0 ; THIS WORD CONTAINS THE MICRO ADDRESS
; THAT WAS SPECIFIED IF A "SET FP" COMMAND
; COMMAND HAS BEEN ISSUED. IT IS USED BY THE
; GO CHAIN MONITOR TO SET THE SYNC POINT
; AT EACH NEWTST STATEMENT.
006562 000000      TERMINT: .WORD
006564 000000      MODLNK: .WORD ; THIS LOCATION IS LOADED BY THE
; HARDCORE MONITOR AND THE FAILCHAIN MONITOR
; TO POINT AT THE RADSO LIST OF MODULE NAMES

```

GLOBAL TAGS

```
63  
64 006570 HEAD=  
65 006570 000000 .WORD ; THE BYTE COUNT OF THIS FILE IS PLACED HERE  
66 ; BY THE LINKER  
67 006572 000167 002202 JMP HARDCO ; LINKAGE FROM THE MICRO DIAGNOSTIC MONITOR
```

```

69 .SBTTL "HARDCORE MONITOR COMMON TAGS
70
71 :*****
72 :+
73 : FOLLOWING ARE THE 512 BYTES (4 SECTORS) OF LOCAL VARIABLES FOR THIS
74 : PROGRAM. THESE VARIABLES MUST ALWAYS START AT RELATIVE ADDRESS 0 AND
75 : MUST BE EXACTLY 5.2 BYTES LONG.
76 :-
77 :*****
78
79 006576 000000 TSTPTR: .WORD ; INDEX INTO TEST TABLE
80 006600 000000 $TMP0: .WORD ; TEMPORARY STORAGE
81 006602 000000 $ITEMB: .WORD ; NOT USED
82 006604 000000 LOADAD: .WORD ; LOAD ADDRESS OF THIS OVERLAY
83 006606 000000 $FLAG: .WORD ; USED BY FETCH ROUTINE
84 006610 000000 LPICNT: .WORD ; LOOP COUNT OF THE CURRENT TEST
85 006612 000000 .WORD ; J COUNTER
86 006614 000000 .WORD ; K COUNTER
87 006616 006624' LOOPTBL: .WORD IINDX ; PTR TO I INDEX TABLE
88 006620 006634' .WORD JINDX ; PTR TO J INDEX TABLE
89 006622 006644' .WORD KINDX ; PTR TO K INDEX TABLE
90
91 006624 IINDX: .BLKW 4 ; I INDEX TABLE
92 006634 JINDX: .BLKW 4 ; J INDEX TABLE
93 006644 KINDX: .BLKW 4 ; K INDEX TABLE
94
95 006654 000000 ARG1: .WORD 0 ; ARGUMENT 1 OF CURRENT OP CODE
96 006656 000000 ARG2: .WORD 0 ; ARGUMENT 2
97 006660 000000 ARG3: .WORD 0 ; ARGUMENT 3
98 006662 000000 ARG4: .WORD 0 ; ARGUMENT 4
99 006664 000000 ARG5: .WORD 0 ; ARGUMENT 5
100 006666 000000 ARG6: .WORD 0 ; ARGUMENT 6
101 006670 000000 STADR: .WORD ; PHYSICAL START ADDRESS OF THE TEST STREAM
102 006672 000000 ENDADR: .WORD ; CONTAINS THE LAST ADDRESS+2 OF THE CURRENT OVERLAY
103 006674 000000 MSKFLG: .WORD 0 ; FLAG FOR THE "CMPCAM" ROUTINE
104 006676 000001 DBLFLG: .WORD 1 ; FLAG FOR THE "CMPCA" AND "CMPCAM" ROUTINES
105 006700 000000 LOSTAD: .WORD 0 ; USED TO SAVE THE ADDRESS OF THE LAST
106 ; NEWTST STATEMENT
107 006702 000160 MAXCNT: .WORD 160 ; TOTAL NUMBER OF V BUS BITS IN THE
108 ; LONGEST CHANNEL
109 006704 000000 $CHKFLG: .WORD 0 ; SET BY THE TEST V BUS ROUTINE
110 006706 000000 ERRCON: .WORD 0 ; USED TO SAVE TPC OF INST FOLLOW IFERROR
111 006710 000000 DATTYPE: .WORD 0 ; IF NON ZERO, INDICATES 32 BIT DATA FOR TYPEOUT
115 006712 000000 WCSSIZE: .WORD ; CONTAINS NUMBER OF WCS MODULES
116 006714 000000 ZERO: .WORD 0 ; ZERO WORD FOR TYPEOUT
117 006716 177777 SIZEFLG: .WORD -1 ; TYPEOUT FLAG FOR "TYPESIZE" ROUTINE
118 006720 177777 SPANFLAG: .WORD -1
119 006722 000000 KEYBUF: .WORD
120
121 006724 TWOSPC: MES < > ; ASCII STRING FOR TWO SPACES
122
123 006726 MSG1: MES <NO. OF 1K BANKS OF WCS = >,NB
124 006730 MSG2: MES <?UNEXPECTED TRAP TO 4...TPC= >,NB
125 006776 MSG3: MES <?ILL WCS CONF - DATA REG= >
126 007022 MSG4: MES <?UNEXPECTED INTRPT...TPC= >
127 007046 MSG24: MES < TPC= >,NB
128

```



```

129 .SBTTL "THE DISPATCH TABLE TO THE EXECUTE SUBROUTINES
130
131 007060' TBLHEAD =
132 007060 015016' DISPATCH: $NOOP ; NO OPERATION
133 007062 014506' $NEWTST ; START A NEW TEST
134 007064 014046' $LOOP ; SETUP A LOOP
135 007066 012604' $ENDLOOP ; END A LOOP
136 007070 013030' $ERRLOOP ; SET THE ERROR LOOP TPC
137 007072 013324' $IFERR ; CHECK THE ERROR FLAG
138 007074 012042' $CMPCA ; COMPARE CONSOLE ADAPTER REGISTER (16 BITS)
139 007076 012036' $CPCAM ; COMPARE CONSOLE ADAPTER REGISTER (16 BITS)
140 ; MASKED
141 007100 015160' $RESET ; EXECUTE AN LSI-11 RESET
142 007102 011464' $BLKMIC ; LOAD A BLOCK OF MICRO INSTRUCTIONS
143 007104 015164' $SETPSW ; SET THE PSW
144 007106 012010' $CLOCK ; STEP THE CPU CLOCK
145 007110 015414' $TSTVB ; TEST THE V BUS
146 007112 013730' $LDIDRE ; LOAD AN ID BUS REGISTER
147 007114 012702' $ENDOVR ; END THIS OVERLAY
148 007116 015220' $SKIP ; SKIP SOME INSTRUCTIONS
149 007120 011656' $CHKPNT ; CHECK THE V BUS ERROR FLAG
150 007122 015032' $REPORT ; TYPE THE FAILING MODULES
151 007124 015020' $READID ; READ AN ID BUS REGISTER
152 007126 013174' $FLTONE ; GENERATE A FLOATING ONE PATTERN
153 007130 013246' $FLTZRC ; GENERATE A FLOATING ZERO PATTERN
154 007132 012434' $CMPPCS ; COMPARE THE UPC SAVE REGISTER
155 007134 012534' $ENDHC ; END THE HARDWARE TEST STREAM
156 007136 013654' $LDCA ; LOAD A CONSOLE ADAPTER REGISTER
157 007140 015716' $TYPsize ; TYPE THE SIZE OF THE WCS
158 007142 013040' $FETCH ; FETCH A MICRO INSTRUCTION
159 007144 015174' $SETVEC ; SET AN LSI-11 TRAP VECTOR
160 007146 013552' $INIT ; INITIALIZE THE CPU
161 007150 014370' $MASK ; MASK SOME DATA
162 007152 013604' $KMXGEN ; GENERATE A KMX FIELD OF A MICRO WORD
163 007154 015332' $SUBTEST ; INCREMENT THE SUBTEST NUMBER
164 007156 014420' $MOVE ; MOVE SRC TO DST
165 007160 015262' $SPAGEN ; GENERATE A SPA ADDR FIELD OF A MICRO WORD
166 007162 015236' $SKIPERRR ; SKIP IF ERROR FLAG SET
167 007164' TBLEND =
168 000042 TBLsiz = <TBLEND - TBLHEAD>/2
169
170 007164 VBEUFF: .BLKB 160 ; BUFFER FOR THE V BUS BITS
171
172 007344' ) =
173 000754 ) = Y-W
174 000024 ) = 1000-X
175 007344 FILL Z
176
177 ;*
178 ; THE FOLLOWING LIST OF MODULE NAMES (IN ASCII FORMAT) IS USED RE THE
179 ; "TYPE ERROR" ROUTINE. THE MODULE NAME MUST BE EXACTLY 7 CHARACTERS LONG
180 ; SINCE THIS LIST IS INDEXED TO PICKUP THE APPROPRIATE NAME.
181 ;-
182
183 007370 MODULES: MES <36>, NB ; CIB
184 007372 MES <35>, NB ; USC
185 007374 MES <33>, NB ; WCS

```

```

186 007376      MES <34>,NB ; PCS
187 007400      MES <29>,NB ; DAP
188 007402      MES <28>,NB ; DCP
189 007404      MES <27>,NB ; DDP
190 007406      MES <26>,NB ; DEP
191 007410      MES <25>,NB ; DBP
192 007412      MES <30>,NB ; CEH
193 007414      MES <31>,NB ; ICL
194 007416      MES <20>,NB ; CAM
195 007420      MES <21>,NB ; CDM
196 007422      MES <22>,NB ; TBM
197 007424      MES <18>,NB ; SBL
198 007426      MES <19>,NB ; SBH
199 007430      MES <24>,NB ; IRC
200 007432      MES <23>,NB ; IDP
201 007434      MES <14>,NB ; MSB
202 007436      MES <13>,NB ; MCN
203 007440      MES <12>,NB ; MDT
204 007442      MAY4K: MES <11>,NB ; MAY
205 007444      MES <32>,NB ; CLK
206 007446      MES <37>,NB ; TRS
207 007450      MES <85>,NB ; FNM
208 007452      MES <86>,NB ; FMH
209 007454      MES <87>,NB ; FML
210 007456      MES <88>,NB ; FAD
211 007460      MES <89>,NB ; FCT
212 007462      MAY16K: MES <10>,NB ; MAY 16K CHIP
213 007464      MES <58> ; MPI
214 007466      MES <59> ; MPC
215 007470      MES <61> ; MPS
216 007472      MES <60> ; MAT
217 007474      MES <38> ; WCS 2K
218
219 007476      MES <76> ; MSB FOR MS780-E
220 007500      MES <75> ; BY LOWER
221 007502      MES <75> ; BY UPPER
222 007504      MES <73> ; 1 MEGABYTE ARRAY
223 007506      MES <74> ; 4 MEGABYTE ARRAY
224      ;*
225      ; THE FOLLOWING LIST ARE THE BUS NAMES THAT ARE USED IN THE "TYPE ERROR"
226      ; ROUTINE. THEY, LIKE THE MODULE NAMES, MUST BE 2 BYTES LONG. (THE MES
227      ; MACRO WILL CAUSE THIS TO HAPPEN).
228      ;-
229
230 007510      BUSES: MES <CS>,NB
231 007512      MES <ID>,NB
232 007514      MES <VB>,NB
233 007516      ADAPT: MES <UBA>
234 007522      MES <MBA>
235

```

```

237 .SBTTL "HARDORE MONITOR SUBROUTINES
238 ;*****
239 ;+
240 ; THE FOLLOWING SUBROUTINES ARE USED EXCLUSIVELY BY THIS PROGRAM.
241 ;-
242 ;*****
243
244 .SBTTL " TYPE ERROR DATA SUBROUTINE
245 ;*****
246 ;+
247 ; THIS SUBROUTINE TYPES THE CONTENTS OF "GOODDAT" AND
248 ; "BADDAT" IN EITHER 16 OR 32 BIT FORMAT DEPENDING ON
249 ; THE CONTENTS OF "DATTYPE".
250 ;-
251 ;*****
252
253 007526 TYPDAT: TYPE #MSGA ; TYPE THE DATA IDENTIFIER
254 007544 005767 177140 TST DATTYPE ; 32 BIT FORMAT?
255 007550 001440 BEQ 1$ ; BRANCH IF NO
256 007552 TYPED #GOODDAT,HEX ;
257 007572 TYPE #CRLF,ASCII ;
258 007612 TYPE #SIXSPC ; TYPE SIX SPACES
259 007630 TYPED #BADDAT,HEX ;
260 007650 000437 BR 2$
261 007652 1$: TYPES #GOODDAT,HEX ;
262 007672 TYPE #CRLF,ASCII ;
263 007712 TYPE #SIXSPC ;
264 007730 TYPES #BADDAT,HEX ;
265 007750 2$: TYPE #CRLF,ASCII ;
266 007770 TYPE #SIXSPC ; TYPE SIX SPACES
267 010006 005767 176576 TST LP1CNT ; IS LOOP COUNT BEING USED?
268 010012 001427 BEQ 3$ ; BRANCH IF NO
269 010014 TYPES #LP1CNT,HEX ; TYPE THE LOOP COUNT OF THE TEST
270 010034 TYPE #CRLF,ASCII
271 010054 TYPE #SIXSPC
272 010072 005767 176514 3$: TST LP1CNT+2 ; IS LOOP COUNT BEING USED?
273 010076 001427 BEQ 4$ ; BRANCH IF NO
274 010100 TYPES #LP1CNT+2,HEX
275 010129 TYPE #CRLF,ASCII
276 010140 TYPE #SIXSPC
277 010156 005767 176432 4$: TST LP1CNT+4 ; IS LOOP COUNT BEING USED?
278 010162 001410 BEQ 5$ ; BRANCH IF NO
279 010164 TYPES #LP1CNT+4,HEX
280 010204 5$: TYPE #CRLF,ASCII ;
281 010224 TYPE #MSGB ; TYPE "TRACE:"
282 010242 RETURN
283
284
285 .SBTTL " READ V BUS SUBROUTINE
286 ;*****
287 ;+
288 ; THIS ROUTINE READS THE V BUS INTO A BUFFER STARTING AT
289 ; LOCATION "VBBUFF". IF THE NUMBER OF BITS TO READ IS NOT
290 ; SPECIFIED, THE ENTIRE BUS IS READ.
291 ;-
292 ;*****
293

```

## " READ V BUS SUBROUTINE

```

294 010244 052737 000002 173036 $READVB:BIS #VBLOAD,a#VBCTRL ; LOAD THE V BUS
295 010252 042737 000002 173036 BIC #VBLOAD,a#VBCTRL ;
296 010260 016700 176416 MOV MAXCNT,R0 ; READ THE WHOLE BUS
297 010264 012701 007164 MOV #VBBUFF,R1 ; GET THE ADDRESS OF THE BUFFER
298 010270 066701 176310 ADD LOADAD,R1 ; ADD RELOCATION FACTOR
299 010274 113721 173037 2$: MOVB a#VBCTRL+1,(R1)+ ; LOAD THE BUFFER
300 010300 052737 000001 173036 BIS #VBCLK,a#VBCTRL ; SHIFT THE BUS
301 010306 005300 DEC R0 ; DECREMENT THE LOOP COUNT
302 010310 001371 BNE 2$ ; CONTINUE
303 010312 RETURN

```

304

305

306 .SBTTL " EXPECTED TRAP ROUTINE

307 ;\*\*\*\*\*

308 ;+

```

309 ; THIS IS THE EXPECTED TRAP ROUTINE. IT IS USED BY THE TEST THAT
310 ; CHECKS FOR A Q BUS TIMEOUT FOR CERTAIN CONSOLE ADAPTER REGISTER
311 ; ADDRESSES.

```

312 ;

```

313 ; A "SCTVER N" PSEUDO INSTRUCTION MUST HAVE BEEN EXECUTED PRIOR TO
314 ; EXPECTING THE TRAP. THIS ROUTINE CLEARS THE ERROR FLAGS AND RETURNS.

```

315 ;-

316 ;\*\*\*\*\*

317

318 010314 005067 176066 TRAP: CLR \$ERFLG ; CLEAR THE ERROR FLAG

319 010320 022626 CMP (SP)+,(SP)+ ; CLEAN UP THE STACK

320 010322 RETURN ; RETURN TO NEXT PSEUDO INSTRUCTION

321

322

323 .SBTTL " UNEXPECTED TRAP ROUTINE

324 ;\*\*\*\*\*

325 ;+

```

326 ; THIS IS THE UNEXPECTED TRAP ROUTINE.
327 ; IT IS USED BY THE TEST OF THE CONSOLE ADAPTER REGISTER ADDRESS SPACE.
328 ; THE "NEWTST" OR "SUBTEST" PSEUDO INSTRUCTIONS SET LOCATION 4 TO POINT
329 ; AT THIS ROUTINE. THIS ROUTINE SETS THE ERROR FLAGS AND GOES TO THE
330 ; "IFERROR" ROUTINE WHERE THE ERROR REPORT IS MADE.

```

331 ;-

332 ;\*\*\*\*\*

333

334 010324 CATCH: TYPE #SRLF,ASCII

335 010344 TYPE #MSG2 ; TYPE UNEXPECTED TRAP MESSAGE

336 010362 016767 176044 176210 CATEX: MOV TPC,\$TMP0

337 010370 166767 176040 176202 SUB RELOC,\$TMP0

338 010376 TYPES # \$TMP0

339 010416 TYPE #SRLF,ASCII

340 010436 022626 CMP (SP)+,(SP)+ ; CLEANUP THE STACK

341 010440 012767 000401 175740 MOV #401,\$ERFLG ; SET THE ERROR FLAG

342 010446 012767 177777 176202 MOV #-1,ARG2

343 010454 016767 175752 176224 MOV TPC,ERRCON

344 010462 000157 002764 JMP E2ERR ; GO TO IFERROR ROUTINE

345

346 .SBTTL " UNEXPECTED INTERRUPT ROUTINE

347 ;\*\*\*\*\*

348 ;+

```

349 ; THIS IS THE UNEXPECTED INTERRUPT ROUTINE.
350 ; THIS ROUTINE IS USED TO CATCH UNEXPECTED INTERRUPTS FROM THE CONSOLE

```

" UNEXPECTED INTERRUPT ROUTINE

```

351 ; ADAPTER. IT IS USED IN THE TEST THAT CHECKS THE READY AND DONE INTERRUPTS
352 ; IN THE CONSOLE ADAPTER. THE "NEWTST" AND "SUBTEST" PSEUDO INSTRUCTIONS
353 ; SET THE INTERRUPT VECTORS TO POINT AT THIS ROUTINE. IT TYPES A
354 ; MESSAGE AND TRANSFERS TO THE UNEXPECTED TRAP ROUTINE.
355 ;-
356 ;:*****
357

```

```

358 010466 CATCHI: TYPE #SRLF,ASCII
359 010506 TYPE #MSG4
360 010524 000716 BR CATEX
361

```

```

362
363 .SBTTL " TYPE PROGRAM NAME AND VERSION
364 ;:*****
365 ;+

```

```

366 ; THIS ROUTINE TYPES THE PROGRAM NAME AND VERSION. THE FIRST SECTOR OF
367 ; THE FILE MUST BE IN MEMORY STARTING AT THE ADDRESS POINTED TO BY
368 ; THE CONTENTS OF "RELOC".
369 ;-
370 ;:*****

```

```

371 010526 010046 TYPVER: MOV R0,-(SP) ; SAVE R0
372 010530 016700 175700 MOV RELOC,R0 ; GET POINTER TO ASCII NAME AND VERSION
373 010534 166700 176044 SUB LOADAD,R0 ; DISCARD LOAD ADDRESS
374 010540 TYPE R0,ASCII ; TYPE THE FILE NAME AND VERSION
375 010553 TYPE #SRLF,ASCII ;
376 010576 012600 MOV (SP)+,R0
377 010600 000207 RTS PC ; EXIT
378
379

```

```

380 .SBTTL " SINGLE INSTRUCTION THE HARDCORE ROUTINE
381 ;:*****
382 ;+

```

```

383 ; THIS ROUTINE REQUIRES THE FOLLOWING GLOBAL VARIABLES:
384 ;

```

```

385 ; "TPC" - CONTAINS THE RELOCATED VALUE OF THE HARDCORE TEST PC.
386 ; "RELOC" - CONTAINS THE PHYSICAL START ADDRESS OF THE HARDCORE
387 ; TEST STREAM BUFFER.
388 ;

```

```

389 ; THIS ROUTINE TYPES THE CURRENT VALUE OF THE NON-RELOCATED TEST PC (TPC).
390 ; IT THEN ISSUES A READ REQUEST TO THE KEYBOARD FOR ONE CHARACTER.
391 ; WHEN THE OPERATOR TYPES A CHARACTER, IT IS CHECKED TO SEE IF IT WAS A
392 ; "SPACE (ASCII 40)". IF A SPACE IS TYPED, A CARRIAGE RETURN LINE FEED
393 ; IS TYPED AND EXECUTION IS RETURNED TO THE CALLING SEQUENCE.
394 ;

```

```

395 ; IF ANY OTHER CHARACTER WAS TYPED, A CARRIAGE RETURN LINE FEED IS TYPED,
396 ; THE SINGLE INSTRUCTION FLAG IS CLEARED, AND THE MICRO MONITOR SUBROUTINE
397 ; IS CALLED.
398 ;

```

```

399 ; THIS ROUTINE IS USED TO SINGLE INSTRUCTION
400 ; THE HARDCORE TEST STREAM. IT REQUIRES THAT THE "SINGLE INSTRUCTION"
401 ; FLAG IS SET.
402 ;-
403 ;:*****
404

```

```

405 010602 SGLINST:
406 010602 MFPS $PSW ; SAVE THE PSW
407 010610 MTPS #0 ; SET PSW AT ZERO

```

" SINGLE INSTRUCTION THE HARDWARE ROUTINE

```

408 010616      TYPE #SRLF,ASCII
409 010636      TYPE #MSG24
410 010654 016767 175552 175716  MOV TPC,$TMP0 ; GET VALUE OF TPC
411 010662 166767 175546 175710  SUB RELOC,$TMP0 ; SUBTRACT RELOCATION CONSTANT
412 010670      TYPES #TMP0 ; TYPE IT IN OCTAL
413 010710      TYPE #TWOSPC ; TYPE TWO SPACES
417 010726      T$INIT ; KILL THE CURRENT INPUT REQUEST
418 010730      T$READ #KEYBUF,#1 ; READ ONE CHARACTER
422 010744 122767 000040 175751  CMPB #40,KEYBUF+1 ; WAS IT A SPACE?
423 010752 001405      BEQ 1$ ; BRANCH IF YES
424 010754 042767 000400 175444  BIC #SINST,SWR ; CLEAR THE SINGLE INSTRUCTION FLAG
425 010762      CALLMICMON
426 010764 000401      BR 2$
427 010766      1$: ENCTRLC ; CALL MONITOR TO ENABLE KEYBOARD REQUEST
428 010770      2$: MTPS $PSW ; RESTORE THE PSW
429 010776 000207      RTS PC ; RETURN
430
431

```

KMUX GENERATE SUBROUTINE

```

1050 ;:*****:
1051 ; THIS ROUTINE GENERATES A KMX FIELD IN THE SPECIFIED MICRO INSTRUCTION
1052 ; EQUAL TO THE CURRENT LOOP COUNT MINUS 1.
1053 ;:-
1054 ;:*****:
1055
1056 013604 016700 173044 $KMXGEN:MOV ARG1,R0 ; GET ADDRESS OF MICRO INSTRUCTION
1057 013610 066700 172620 ADD RELOC,R0 ; ADD RELOCATION FACTOR
1058 013614 062700 000006 ADD #6,R0 ; SELECT THE 4TH 16 BIT WORD (KMX FILED STARTS AT BITS8)
1059 013620 016701 173032 MOV ARG2,R1 ; GET INDEX INTO INDEX TABLE POINTER TABLE
1060 013624 066701 172754 ADD LOADAD,R1 ;
1061 013630 017101 006616 MOV @LOOPTB(R1),R1 ; GET THE CURRENT INDEX VALUE
1062 013634 005301 DEC R1 ; ADJUST
1063 013636 000301 SWAB R1 ; PUT IN KMX FILED POSITION
1064 013640 006301 ASL R1 ; ...
1065 013642 006301 ASL R1 ; ...
1066 013644 042710 176000 BIC #176000,(R0) ; CLEAR CURRENT KMX FIELD
1067 013650 050110 BIS R1,(R0) ; INSERT NEW FIELD VALUE
1068 013652 RETURN ; EXIT
1069
1070
1071
1072
1073
1074

```

```

1075 .SBTTL " LOAD CONSOLE ADAPTER REGISTER SUBROUTINE
1076 ;:*****:
1077 ; THIS ROUTINE LOADS THE SPECIFIED CONSOLE ADAPTER REGISTER WITH
1078 ; THE SPECIFIED DATA. THE DATA IS ALWAYS 16 BITS.
1079 ;:-
1080 ;:*****:
1081
1082 013654 016700 172774 $LDCA: MOV ARG1,R0 ; GET THE ADDRESS OF THE CA REGISTER
1083 013660 016701 172772 MOV ARG2,R1 ; GET THE ADDRESS OF THE DATA
1084 013664 016702 172770 MOV ARG3,R2 ; IS THE DATA INDEXED?
1085 013670 100407 BMI 1$ ; BRANCH IF NO
1086 013672 066702 172706 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
1087 013676 017202 006616 MOV @LOOPTB(R2),R2 ; GET THE CURRENT VALUE OF THE INDEX
1088 013702 005302 DEC R2 ; MAKE IT A WORD INDEX
1089 013704 006302 ASL R2 ;
1090 013706 000401 BR 2$ ;
1091 013710 005002 1$: CLR R2 ; NO INDEXING, SO CLEAR INDEX
1092 013712 060201 2$: ADD R2,R1 ; GENERATE THE ADDRESS OF THE DATA
1093 013714 066701 172514 ADD RELOC,R1 ; ADD THE RELOCATION CONSTANT
1094 013720 011110 MOV (R1),(R0) ; LOAD THE REGISTER WITH THE DATA
1095 013722 000240 NOP ; WAIT FOR INTERRUPTS
1096 013724 000240 NOP ; ...
1097 013726 RETURN
1098
1099

```

```

1100 .SBTTL " LOAD ID REGISTER SUBROUTINE
1101 ;:*****:
1102 ; THIS ROUTINE LOADS THE SPECIFIED ID BUS REGISTER WITH THE
1103 ; SPECIFIED DATA. IF THE DATA IS INDEXED, THE INDEX IS MADE
1104 ; ON 32 BITS.
1105 ;:-
1106 ;:*****:

```

```

433 .SBTTL "PROGRAM INITIALIZATION
434 ;*****
435 ;+
436 ; THE FOLLOWING CODE CALCULATES THE LOAD ADDRESS AND SAVES IT IN "LOADAD".
437 ; IT SETS UP THE INTERRUPT AND TRAP VECTORS FOR UNEXPECTED TRAPS.
438 ; IT RELOCATES THE DISPATCH TABLE AND THE LOOP INDEX POINTER TABLES.
439 ; IT THEN OPENS THE TEST STREAM FILE, READS IN THE FIRST OVERLAY, AND
440 ; TYPES THE OVERLAY NUMBER. IT THEN INITIALIZES THE TEST PC (TPC) AND
441 ; THE TEST TABLE POINTER.
442 ;
443 ; EXECUTION THEN TRANSFERES TO THE TEST STREAM INTERPRETER.
444 ;-
445 ;*****
446
447 011000 010700   HARDCO: MOV PC,R0 ; CALCULATE THE OFFSET FROM
448 011002 162700 011002'  SUB #.,R0 ; ABSOLUTE 0
449 011006 010067 175572   MOV R0,LOADAD ; SAVE
450 011012 012767 020400' 175414  MOV #END,RELOC ; INITIALIZE RELOCATION CONSTANT
451 011020 060067 175410   ADD R0,RELOC ; ADJUST THE ADDRESS IN LOCATION RELOC
452 011024 012737 010324' 000004  MOV #CATCH,a#4 ; SET UNEXPECTED TRAP CATCHER
453 011032 066737 175546 000004  ADD LOADAD,a#4
454 011040 012701 010466'   MOV #CATCHI,R1 ; GET ADR OF UNEXPECTE INTRPT CATCHER
455 011044 066701 175534   ADD LOADAD,R1
456 011050 010137 000300   MOV R1,a#300
457 011054 010137 000304   MOV R1,a#304
458 011060 012701 000340   MOV #340,R1
459 011064 005037 000006   CLR a#6 ; SETUP PSW OF TRAP TO 4
460 011070 010137 000302   MOV R1,a#302 ; AND INTRPT VECTORS
461 011074 010137 000306   MOV R1,a#306 ; ...
462 011100 012702 000042   MOV #TBLSIZ,R2 ; ADD RELOCATION FACTOR TO
463 011104 012701 007060'   MOV #DISPAT,R1 ; THE DISPATCH TABLE ENTRIES
464 011110 060001   ADD R0,R1 ; ...
465 011112 060021   5$: ADD R0,(R1)+ ; ...
466 011114 005302   DEC R2 ; ...
467 011116 001375   BNE 5$ ; ...
468 011120 012702 000003   MOV #3,R2 ; ADD RELOCATION FACTOR TO
469 011124 012701 006616'   MOV #LOOPTB,R1 ; THE LOOP INDEX TABLE
470 011130 060001   ADD R0,R1 ; ...
471 011132 060021   6$: ADD R0,(R1)+ ; ...
472 011134 005302   DEC R2 ; ...
473 011136 001375   BNE 6$ ; ...
474 011140 012767 007370' 175416  MOV #MODULES,MODLNK ; INITIALIZE THE MODULE NAME LIST LINK
475 011146 060067 175412   ADD R0,MODLNK ; POINTER
476 011152   OPENFILE TESTSTREAM ; OPEN THE TEST STREAM FILE
477 011164   READOVR RELOC,#256. ; GET DIRECTORY
478 011204 004767 177316   JSR PC,TYPVER ; TYPE THE VERSION NUMBER
479 011210 016767 175220 175452  MOV RELOC,STADR ; INITIALIZE START ADDRESS
480 011216 032767 000300 175202  BIT #LOST+LOSS,SWR ; LOOP ON SPECIAL TEST OR SECTION?
481 011224 001415   BEQ 64$ ; BRANCH IF NO
482 011226   OPENFILE TESTSTREAM ; OPEN THE TEST STREAM FILE
483 011240 066767 175310 175310  ADD LUSSEC,SECTOR ; GENERATE SECTOR ADDRESS OF SPECIAL FUNCTION
484 011246 016767 175124 175116  MOV TESTNO,$STNM ; INIT THE TEST NUMBER
485 011254 005367 175112   DEC $STNM ; ...
486 011260   64$: READOVR RELOC ; READ THE FIRST OVERLAY
487 011300 016767 007076 175074  MOV END+2,$SCTNO ; SAVE THIS SECTION NUMBER
488 011306   TYPESECTNO ; TYPE THE SECTION NUMBER
489 011310 012767 000034 175114  MOV #TPCINIT,TPC ; INITIALIZE THE TPC

```



"PROGRAM INITIALIZATION

```

490 011316 066767 175112 175106 ADD RELOC,TPC ;
491 011324 012767 000004 175244 MOV #ITSTPTR,TSTPTR ; INITIALIZE THE TEST POINTER
492 011332 066767 175076 175236 ADD RELOC,TSTPTR ;
493 011340 032767 060000 175062 BIT #TSTSPAN+SCTSPAN,SWR1 ; WAS A SPAN SPECIFIED?
494 011346 01403 BEQ 66$ ; BRANCH IF NO
495 011350 042767 000300 175050 BIC #LOSS+LOST,SWR ; CLEAR THE LOST AND LOSS FLAGS
496 011356 66$:

```

```

497
498 .SBTTL "TEST STREAM INTERPRETER
499 :*****
500 :+
501 : THE FOLLOWING CODE INTERPRETS THE PSEUDO INSTRUCTIONS IN THE TEST
502 : STREAM. IT FIRST TESTS IF THE "SINGLE INSTRUCTION" FLAG IS SET AND
503 : IF IT IS, A CALL IS MADE TO THE "SINGLE INSTRUCTION ROUTINE" IN THE
504 : MICRO DIAGNOSTIC MONITOR.
505 :
506 : IT THEN PICKS UP THE CURRENT OPCODE AND ARGUMENT COUNT, PUTS THE
507 : ARGUMENTS OF THE OPCODE IN LOCATIONS "ARG1" THRU "ARG6" AND DOES A
508 : SUBROUTINE CALL TO THE ROUTINE SPECIFIED BY THE OPCODE.
509 :-
510 :*****
511

```

```

512 011356 4$: CHKKEY ; CHECK IF KEY SWITCH CHANGE
011356 104023 EMT CHKSWITCH
513 011360 032767 000400 175040 BIT #SINST,SWR ; SINGLE INSTRUCTION MODE SET?
514 011366 001412 BEQ 7$ ; BRANCH IF NO
515 011370 032767 000200 175030 BIT #LOST,SWR ; LOOP ON SPECIAL TEST?
516 011376 001404 BEQ 9$ ; BRANCH IF NO
517 011400 026767 174766 174770 CMP $TSTNM,TESTNO ; ON THE TEST YET?
518 011406 001002 BNE 7$ ; BRANCH IF NO
519 011410 004767 177166 9$: JSR PC,SGLINST ; CALL THE SINGLE INSTRUCTION ROUTINE
520 011414 016700 175012 7$: MOV TPC,R0 ; GET THE CURRENT TPC
521 011420 112001 MOVB (R0)+,R1 ; GET THE OP CODE
522 011422 112002 MOVB (R0)+,R2 ; GET THE NUMBER OF WORD ARGUMENTS
523 011424 001407 BEQ 2$ ; BRANCH IF NO ARGUMENTS
524 011426 012703 006654 MOV #ARG1,R3 ; GET THE ADDRESS OF THE ARGUMENT TABLE
525 011432 066703 175146 ADD LOADAD,R3 ; ADD RELOCATION FACTOR
526 011436 012023 1$: MOV (R0)+,(R3)+ ; PICK UP AN ARGUMENT
527 011440 005302 DEC R2 ; IS THE LOOP DONE?
528 011442 001375 BNE 1$ ; BRANCH IF NO
529 011444 010067 174762 2$: MOV R0,TPC ; UPDATE THE TPC
530 011450 006301 ASL R1 ; MAKE OP CODE A WORD INDEX
531 011452 066701 175126 ADD LOADAD,R1 ; ADD RELOCATION FACTOR
532 011456 004771 007060 JSR PC,DISPAT(R1) ; GO TO THE EXECUTE SUBROUTINE
533 011462 000735 BR 4$ ; CONTINUE
534
535

```

## " BLOCK MIC SUBROUTINE

```

537 .SBTTL " BLOCK MIC SUBROUTINE
538 ;:*****
539 ;+
540 ; THIS ROUTINE MOVES A BLOCK OF MICRO WORDS TO WCS.
541 ;-
542 ;:*****
543
544 011464 016700 175164 $BLKMIC:MOV ARG1,R0 ; GET THE ADDRESS OF THE MICRO INSTRUCTION
545 011470 066700 174740 ADD RELOC,R0 ; ADD THE RELOCATION CONSTANT
546 011474 116701 175162 MOV B ARG4,R1 ; IS THIS ADDRESS INDEXED?
547 011500 100420 BMI 2$ ; BRANCH IF NO
548 011502 066701 175076 ADD LOADAD,R1 ; ADD RELOCATION CONSTANT
549 011506 017101 006616 MOV @LOOPTB(R1),R1 ; GET THE CURRENT VALUE OF THE INDEX
550 011512 016702 175142 MOV ARG3,R2 ; GET THE WORD COUNT OF THE BLOCK
551 011516 010203 MOV R2,R3 ; SAVE IN R3
552 011520 006302 ASL R2 ; CALCULATE THE SIZE OF THE BLOCK IN BYTES
553 011522 060302 ADD R3,R2 ; BY MULTIPLYING THE WORD COUNT BY 14(8)
554 011524 006302 ASL R2 ;
555 011526 006302 ASL R2 ;
556 011530 005003 CLR R3 ; INITIALIZE A WORKING REGISTER
557 011532 005301 1$: DEC R1 ; MULTIPLY THE SIZE OF THE BLOCK BY
558 011534 001403 BEQ 3$ ; THE CURRENT INDEX
559 011536 060203 ADD R2,R3 ;
560 011540 000774 BR 1$ ;
561 011542 005003 2$: CLR R3 ; NO INDEXING SO CLEAR THE INDEX
562 011544 060300 3$: ADD R3,R0 ; GENERATE THE ADR OF THE U INSTRUCTION
563 011546 016701 175104 MOV ARG2,R1 ; GET THE WCS ADDRESS
564 011552 116702 175105 MOV B ARG4+1,R2 ; IS THE ADDRESS INDEXED?
565 011556 100412 BMI 4$ ; BRANCH IF NO
566 011560 066702 175020 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
567 011564 017202 006616 MOV @LOOPTB(R2),R2 ; GET CURRENT VALUE OF INDEX
568 011570 005302 DEC R2
569 011572 005767 175066 TST ARG5 ; IS ADDRESS IN LSI-11 MEMORY?
570 011576 001403 BEQ 5$ ; BRANCH IF NO
571 011600 006302 ASL R2 ; MAKE INDEX A WORD INDEX
572 011602 000401 BR 5$
573 011604 005002 4$: CLR R2 ; NO INDEXING
574 011606 060201 5$: ADD R2,R1 ; INDEX THE WCS ADDRESS
575 011610 005767 175050 TST ARG5 ; IS ADDRESS IN LSI-11 TABLE?
576 011614 001403 BEQ 6$ ; BRANCH IF NO
577 011616 066701 174612 ADD RELOC,R1 ; GET ADDRESS OF TABLE
578 011622 011101 MOV (R1),R1 ; GET THE WCS ADDRESS
579 011624 016702 175030 6$: MOV ARG3,R2 ; PUT THE WORD COUNT IN R2
580 011630 010203 MOV R2,R3 ; MULTIPLY THE WORD COUNT BY 3
581 011632 006302 ASL R2 ;
582 011634 060302 ADD R3,R2 ;
583 011636 LOADWCS R0,R1,R2 ; LOAD THE WCS WITH THE BLOCK
584 011654 RETURN ; EXIT
585
586
587 .SBTTL " CHECK POINT SUBROUTINE
588 ;:*****
589 ;+
590 ; THIS ROUTINE IS USED TO TEST THE RESULT OF A "TSTVB" PSEUDO
591 ; INSTRUCTION. IT PERFORMS THE FOLLOWING FUNCTIONS:
592 ;
593 ; 1) IF THE "NER" FLAG IS CLEAR THE TPC OF THE CALL

```

" CHECK POINT SUBROUTINE

```

594 : WILL BE TYPED FOLLOWED BY A COMMA SPACE. THIS
595 : OUTPUT OCCURS ON THE LINE NAMED "TRACE".
596 :
597 : 2) IF THE V BUS TEST DID NOT FAIL (LOCATION $CHKFLG IS
598 : CLEAR), THE TPC IS SET TO THE "PASS ADDRESS" IF IT
599 : WAS SPECIFIED OTHERWISE THE TPC IS UNCHANGED.
600 :
601 : 3) IF THE V BUS TEST FAILED (LOCATION $CHKFLG IS NON ZERO)
602 : THE TPC IS SET TO THE "FAIL ADDRESS" IF IT WAS
603 : SPECIFIED, OTHERWISE THE TPC IS UNCHANGED.
604 :-
605 : *****
606
607 011656 005767 175022 $CHKPNT:TST $CHKFLG ; V BUS FAILURE?
608 011662 001407 BEQ 2$ ; BRANCH IF NO
609 011664 005767 174766 TST ARG2 ; FAIL ADDRESS SPECIFIED?
610 011670 100415 BMI 4$ ; BRANCH IF NO
611 011672 016767 174760 174532 MOV ARG2,TPC ; SET TPC TO FAIL ADDRESS
612 011700 000406 BR 3$ ; EXIT
613 011702 005767 174746 2$: TST ARG1 ; PASS ADDRESS SPECIFIED?
614 011706 100406 BMI 4$ ; BRANCH IF NO
615 011710 016767 174740 174514 MOV ARG1,TPC ; SET TPC TO PASS ADDRESS
616 011716 066767 174512 174506 3$: ADD RELOC,TPC ; ADD RELOCATION FACTOR TO TPC
617 011724 032767 000010 174474 4$: BIT #NER,SWR ; INHIBIT ERROR REPORT?
618 011732 061025 BNE 1$ ; BRANCH IF YES
619 011734 016767 174472 174636 MOV TPC,$TMP0 ; GET NEXT TPC
620 011742 166767 174466 174630 SUB RELOC,$TMP0 ; SUBTRACT RELOCATION OFFSET
621 011750 TYPES # $TMP0 ; TYPE IT
622 011770 TYPE #COMSPC ; TYPE A " , "
623 012006 1$: RETURN ; EXIT

```

.SBTTL " CLOCK SUBROUTINE

```

624
625 .SBTTL " CLOCK SUBROUTINE
626 : *****
627 :+
628 ; THIS ROUTINE TICKS THE VAX SYSTEM CLOCK IN SINGLE TIME STATE
629 ; MODE 1: NUMBER OF TICKS SPECIFIED BY THE FIRST ARGUMENT
630 ; CF THE "CLOCK" MACRO.
631 :-
632 : *****
633
634 012010 022767 000004 174636 $CLOCK: CMP #4,ARG1 ; DO A SINGLE BUS CYCLE?
635 012016 001002 BNE 1$ ; BRANCH IF NO
636 012020 SBCCLOCK ; TICK THE CLOCK (SINGLE BUS CYCLE)
637 012022 000404 BR 2$ ;
638 012024 1$: STSCLOCK ARG1 ; TICK THE CLOCK (SINGLE TIME STATE)
639 012034 2$: RETURN

```

.SBTTL " COMPARE CONSOLE ADAPTER REGISTER SUBROUTINE

```

640
641 .SBTTL " COMPARE CONSOLE ADAPTER REGISTER SUBROUTINE
642 : *****
643 :+
644 ; THIS ROUTINE IS ENTERED BY FOUR PSEUDO INSTRUCTIONS: CMPCA,
645 ; CMPCAD, CMPCAM AND CMPCMD.
646 :-
647 : *****
648
649
650 012036 005267 174632 $CPCAM: INC MSKFLG ; SET THE MASK FLAG

```

## " COMPARE CONSOLE ADAPTER REGISTER SUBROUTINE

```

651 012042 116700 174606 $CMPCA: MOV B ARG1,R0 ; GET THE MODE VALUE
652 012046 005067 174632 CLR $CHKFLG ; AND THE CHECK ERROR FLAG
653 012052 066700 174526 ADD LOADAD,R0 ; ADD RELOCATION FACTOR
654 012053 116701 174573 MOV B ARG1+1,R1 ; IS THE DATA INDEXED?
655 012062 100407 BMI 1$ ; BRANCH IF NO
656 012064 066701 174514 ADD LOADAD,R1 ; ADD RELOCATION CONSTANT
657 012070 017101 006616 MOV @L0OPTB(R1),R1 ; GET CURRENT INDEX
658 012074 005301 DEC R1 ;
659 012076 006301 ASL R1 ; MAKE IT A WORD INDEX
660 012100 000401 BR 2$ ;
661
662 ;+
663 ; DATA IS NOT INDEXED SO CLEAR THE INDEX VALUE.
664 ; -
665
666 012102 005001 1$: CLR R1 ;
667 012104 016702 174546 2$: MOV ARG2,R2 ; GET THE ADDRESS OF THE CONSOLE
668 ; ADAPTER REGISTER
669 012110 022702 177777 CMP #-1,R2 ; IS DATA IN LSI-11 MEMORY?
670 012114 001004 BNE 12$ ; BRANCH IF LOW WORD ISN'T
671 012116 012702 006464 MOV #RDIDLOW,R2 ; PUT MEMORY ADDRESS IN R2
672 012122 066702 174456 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
673 012126 022702 000001 12$: CMP #1,R2 ; IS HIGH DATA IN LSI-11 MEMORY?
674 012132 001004 BNE 11$ ; BRANCH IF NO
675 012134 012702 006456 MOV #RDIDHI,R2 ; PUT MEMORY ADDRESS IN R2
676 012140 066702 174440 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
677 012144 016703 174510 11$: MOV ARG3,R3 ; GET ADDRESS OF DATA TABLE
678 012150 066703 174260 ADD RELOC,R3 ; ADD THE RELOCATION FACTOR
679 012154 105767 174502 TST B ARG4 ; DOUBLE MODE?
680 012160 100410 BMI 13$ ; BRANCH IF NO
681 012162 005767 174510 TST DBLFLG ; IS THIS FIRST CALL OF THE DOUBLE CALL?
682 012166 001403 BEQ 9$ ; BRANCH IF NO
683 012170 005067 174502 CLR DBLFLG ;
684 012174 000402 BR 13$ ;
685 012176 005267 174474 9$: INC DBLFLG ; SET DOUBLE FLAG TO INDICATE SECOND TIME
686 012202 005767 174466 13$: TST MSKFLG ; IS THIS A MASK CALL?
687 012206 001007 BNE 4$ ; BRANCH IF YES
688
689 ;+
690 ; THE CALL WAS A "CMPCAD". ADJUST THE INDEX VALUE.
691 ; -
692
693 012210 105767 174446 10$: TST B ARG4 ; DOUBLE MODE?
694 012214 100401 BMI 3$ ; BRANCH IF NO
695 012216 006301 ASL R1 ; MAKE THE INDEX A 32 BIT INDEX
696 012220 060103 3$: MOV R1,R3 ; GENERATE THE ADDRESS OF THE DATA
697 012222 021312 CMP (R3),(R2) ; COMPARE THE EXPECTED AND RECEIVED DATA
698 012224 000440 BR 8$ ; GO EXECUTE THE APPROPRIATE BRANCH
699
700 ;+
701 ; THE CALL WAS EITHER A "CMPCAM" OR "CMPCMD". SEE WHICH ONE.
702 ; -
703
704 012226 016704 174432 4$: MOV ARG5,R1 ; GET THE ADDRESS OF THE MASK
705 012232 066704 174176 ADD RELOC,R4 ; ADD THE RELOCATION FACTOR
706 012236 116705 174421 MOV B ARG4+1,R5 ; IS THE MASK INDEXED?
707 012242 100407 BMI 5$ ; BRANCH IF NO

```

```

708 012244 066705 174334 ADD LOADAD,R5 ; ADD THE RELOCATION FACTOR
709 012250 017505 006616 MOV @LOOPTR(R5),R5 ; GET THE CURRENT INDEX
710 012254 005305 DEC R5 ;
711 012256 006305 ASL R5 ; MAKE THE INDEX A WORD INDEX
712 012260 000401 BR 6$ ;
713 012262 005005 5$: CLR R5 ; NO INDEXING SO CLEAR THE INDEX
714 012264 105767 174372 6$: TSTB ARG4 ; IS IT A DOUBLE MODE?
715 012270 100402 BMI 7$ ; BRANCH IF NO
716
717 ;+
718 ; THE INSTRUCTION WAS A "CMPCMD". MAKE INDEX VALUE DOUBLE.
719 ;-
720
721 012272 006301 ASL R1 ; MAKE THE INDEX A 32 BIT INDEX
722 012274 006305 ASL R5 ; ALSO THE MASK INDEX
723 012276 060103 7$: ADD R1,R3 ; GENERATE THE ADDRESS OF THE DATA
724 012300 060504 ADD R5,R4 ; GENERATE THE ADDRESS OF THE MASK
725 012302 011267 174272 MOV (R2), $TMP0 ; PUT THE RECEIVED DATA IN $TMP0
726 012306 012702 006600 MOV # $TMP0,R2 ; GET THE ADDRESS WHERE THE DATA IS
727 012312 066702 174266 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
728 012316 011404 MOV (R4),R4 ; GET THE MASK
729 012320 005104 COM R4 ;
730 012322 040412 BIC R4,(R2) ; MASK THE RECEIVED DATA
731 012324 021312 CMP (R3),(R2) ; MAKE THE COMPARISON OF EXPECTED AND RECEIVED
732 012326 000160 012424 8$: JMP BRTBL(R0) ; GO EXECUTE THE APPROPRIATE BRANCH
733
734 ;+
735 ; THE BRANCH TABLE RETURNS TO EITHER "PASS" OR "FAIL". IF FAIL,
736 ; THE ERROR FLAG IS SET. THE EXPECTED AND RECEIVED DATA IS SAVED
737 ; IN LOCATIONS "GOODDAT" AND "BADDAT" FOR ERROR TYPEOUT.
738 ;-
739
740 012332 012767 000401 174046 FAIL: MOV #401,$ERFLG ; SET THE ERROR FLAG
741 012340 012767 000001 174336 MOV #1,$CHKFLG ; SET THE CHECK FLAG ALSO
742 012346 005067 174322 PASS: CLR MSKFLG ; INITIALIZE THE MASK FLAG
743 012352 105767 174304 TSTB ARG4 ; IS THIS A DOUBLE CALL?
744 012356 100403 BMI 3$ ; BRANCH IF NO
745 012360 005767 174312 TST DBLFLG ; IS THIS FIRST TIME THROUGH?
746 012364 001007 BNE 1$ ; BRANCH IF NO
747 012366 011367 174024 3$: MOV (R3),GOODDAT ; SAVE THE EXPECTED DATA
748 012372 011267 174024 MOV (R2),BADDAT ; SAVE THE RECEIVED DATA
749 012376 005067 174306 CLR DATTYPE ; SET 16 BIT DATA TYPE
750 012402 000407 BR 2$ ;
751 012404 011367 174010 1$: MOV (R3),GOODDAT+2 ; SAVE SECOND WORD OF EXPECTED DATA
752 012410 011267 174010 MOV (R2),BADDAT+2 ; SAVE SECOND WORD OF RECEIVED DATA
753 012414 012767 000001 174266 MOV #1,DATTYPE ; SET 32 BIT DATA TYPE
754 012422 2$: RETURN ; EXIT
755
756
757 012424 001750 BRTBL: BEQ PASS
758 012426 000741 BR FAIL
759 012430 001346 BNE PASS
760 012432 000737 BR FAIL
761
762
763 .SBTTL COMPARE PC SAVE SUBROUTINE
764 ;:.....

```

" COMPARE PC SAVE SUBROUTINE

```

765      ;*
766      ; THIS ROUTINE COMPARES THE SPECIFIED EXPECTED DATA WITH THE
767      ; CONTENTS OF THE MICRO PC SAVE REGISTER. IF THEY ARE NOT EQUAL,
768      ; THE ERROR FLAG "$ERFLG" IS SET.
769      ;-
770      ;*****
771      ;*****
772
773 012434 016700 174214 $CMPPCS:MOV ARG1,R0 ; GET THE ADDRESS OF THE EXPECTED DATA
774 012440 066700 173770 ADD RELOC,R0 ; ADD THE RELOCATION FACTOR
775 012444 016701 174206 MOV ARG2,R1 ; IS IT INDEXED?
776 012450 100407 BMI 1$ ; BRANCH IF NO
777 012452 066701 174126 ADD LOADAD,R1 ; ADD RELOCAT'ON CONSTANT
778 012456 017101 006616 MOV @LOOPTR(R1),R1 ; GET THE CURRENT VALUE OF THE INDEX
779 012462 005301 DEC R1
780 012464 006301 ASL R1 ; MAKE IT A WORD INDEX
781 012466 000401 BR 2$ ;
782 012470 005001 1$: CLR R1 ; NO INDEXING SO CLEAR THE INDEX
783 012472 060100 2$: ADD R1,R0 ; GENERATE THE ADDRESS OF THE EXP DATA
784 012474 011067 173716 MOV (R0),GOODDAT ; SAVE EXPECTED VALUE OF UPC SAVE
785 012500 GETUPC ; GET THE CURRENT VALUE OF THE UPC
786 012502 016767 173712 MOV GOTUPC,BADDAT ; SAVE IT
787 012510 005067 174174 CLR DATTYPE ; SET 16 BIT DATA TYPE
788 012514 026767 173676 CMP GOODDAT,BADDAT ; IS EXPECTED SAME AS RECEIVED?
789 012522 001403 BEQ 5$ ; BRANCH IF YES
790 012524 012767 000401 173654 4$: MOV #401,$ERFLG ; SET THE ERROR FLAG
791 012532 5$: RETURN ; EXIT

```

```

792
793
794 .SBTTL " END HARDCORE SUBROUTINE
795 ;*****
796 ; THIS ROUTINE FIRST CHECKS TO SEE IF A SECTION IS BEING LOOPED
797 ; IF IT IS AND THE SECTION IS BACKWARD, EXECUTION IS
798 ; TRANSFERRED TO THE END OVERLAY ROUTINE. IF THE SECTION IS FORWARD
799 ; OR LOOP ON SPECIAL SECTION IS NOT SET, THE HARDCORE TESTS ARE
800 ; TERMINATED AND EXECUTION RETURNS TO THE MICRO DIAGNOSTIC MONITOR.
801 ;-
802 ;*****

```

```

803
804 012534 032767 000100 173664 $ENDHC: BIT #LOSS,SWR ; LOOP ON SPECIAL SECTION?
805 012542 001010 BNE 1$ ; BRANCH IF YES
806 012544 032767 040000 173656 BIT #SCTSPAN,SWR1 ; SPAN SPECIFIED?
807 012552 001406 BEQ 2$ ; BRANCH IF NO
808 012554 026767 173614 173620 CMP ENDSPAN,$SCTNO ; /
809 012562 001002 BNE 2$ ; /
810 012564 000167 000112 1$: JMP $ENDOVR ; LET END OVERLAY ROUTINE FIGURE IT OUT
811 012570 005726 2$: TST (SP)+ ; POP RETURN ADDRESS FROM STACK
812 012572 DONE ; RETURN

```

```

813
814
815 .SBTTL " ENDL0OP SUBROUTINE
816 ;*****
817 ;*
818 ; THIS ROUTINE FIRST ADDS THE INCREMENT VALUE OF THE LOOP
819 ; TO THE CURRENT LOOP VALUE. IT THEN CHECKS TO SEE IF THE LOOP
820 ; IS FINISHED BY COMPARING THE CURRENT LOOP VALUE WITH THE
821 ; END LOOP VALUE. IF THE LOOP IS NOT FINISHED, THE TPC IS REPLACED

```

ENDLOOP SUBROUTINE

```

822 ; WITH THE CONTENTS OF THE LOOP TPC.
823 :-
824 ;:*****
825
826 012604 016700 174044 $ENDL0P:MOV ARG1,R0 ; GET INDEX INTO LOOP TABLE POINTER TABLE
827 012610 010002 MOV R0,R2 ; SAVE 10 INDEX LPICNT
828 012612 066700 173766 ADD LOADAD,R0 ; ADD RELOCATION CONSTANT
829 012616 016000 006616 MOV LOOPTB(R0),R0 ; GET THE ADDRESS OF THE LOOP TABLE
830 012622 011001 MOV (R0),R1 ; GET THE CURRENT VALUE
831 012624 066001 000004 ADD 4(R0),R1 ; ADD INDEX VALUE
832 012630 010110 MOV R1,(R0) ; SAVE NEW CURRENT LOOP VALUE
833 012632 062702 006610 ADD #LPICNT,R2 ; GENERATE ADDRESS OF WORD TO SAVE THE COUNT
834 012636 067702 173742 ADD LOADAD,R2 ;
835 012642 010112 MOV R1,(R2) ; UPDATE THE LOOP COUNT THAT GETS TYPED
836 012644 005760 000004 TST 4(R0) ; IS INCREMENT POSITIVE OR NEGATIVE?
837 012650 100003 BPL 1$ ; BRANCH IF POSITIVE
838 012652 026010 000002 CMP 2(R0),(R0) ; IS LAST .GT. FIRST?
839 012656 000402 BR 2$
840 012660 021060 000002 1$: CMP (R0),2(R0) ; IS FIRST .GT. LAST?
841 012664 003004 2$: BGT 3$ ; BRANCH IF LOOP DONE
842 012666 016067 000006 173536 MOV 6(R0),TPC ; SET LOOP ADDRESS IN TPC
843 012674 000401 BR 4$
844 012676 005012 3$: CLR (R2) ; CLEAR THE LOOP COUNT FOR TYPEOUT
845 012700 4$: RETURN

```

```

846
847
848 .SBTTL " END RELAY SUBROUTINE
849 ;:*****
850 ;:
851 ; THIS ROUTINE FIRST CHECKS THE LOOP ON SPECIAL SECTION (LOSS)
852 ; FLAG AND IF SET, EITHER THE CURRENT SECTION
853 ; IS LOOPED ON, THE NEXT SECTION IS READ INTO MEMORY, OR THE
854 ; FILE IS REOPENED. IF NOT SET, THE NEXT SECTION IS READ INTO MEMORY.
855 :-
856 ;:*****

```

```

857
858 012702 032767 000100 173516 $END0VR:BIT #LOSS,SWR ; LOOP ON SPECIAL SECTION?
859 012710 001404 BEQ 1$ ; BRANCH IF NO
860 012712 026767 173464 173464 CMP $SCTNO,SECTNO ; IN THE CORRECT SECTION?
861 012720 001426 BEQ 2$ ; BRANCH IF YES
862 012722 032767 040000 173500 1$: BIT #SCTSPAN,SWR1 ; WAS A SECTION SPAN SPECIFIED?
863 012730 001406 BEQ 3$ ; BRANCH IF NO
864 012732 026767 173436 173442 CMP ENDSPAN,$SCTNO ; FINISHED LAST SECTION?
865 012740 001002 BNE 3$ ; BRANCH IF NO
866 012742 4$: CALLMICMON ; DONE
867 012744 000776 BR 4$ ; DON'T ALLOW CONTINUE
868 012746 3$: READ0VR RELOC ; GET THE THE NEXT OVERLAY
869 012766 016767 005410 173406 MOV END+2,$SCTNO ; SAVE THE CURRENT SECTION NUMBER
870 012774 TYPESECTNO
871 012776 012767 000034 173426 2$: MOV #TPCINIT,TPC ; INITIALIZE THE TPC
872 013004 066767 173424 173420 ADD RELOC,TPC ; ADD THE RELOCATION CONSTANT
873 013012 012767 000004 173556 MOV #ITSTPTR,TSTPTR ; INITIALIZE THE TEST POINTER
874 013020 066767 173410 173550 ADD RELOC,TSTPTR ; ADD THE RELOCATION CONSTANT
875 013026 RETURN ; EXIT

```

```

876
877
878 .SBTTL " ERROR LOOP SUBROUTINE

```

## " ERROR LOOP SUBROUTINE

```

879      ;:*****
880      ;:
881      ; THIS ROUTINE SAVES THE CURRENT TPC IN LOCATION "$LPERR" TO
882      ; MINIMIZE THE SIZE OF THE ERROR LOOP.
883      ;:-
884      ;:*****
885
886 013030 016767 173376 173354 $ERRLOP:MOV TPC,$LPERR ; SAVE THE TPC FOR LOOPING
887 013036      RETURN
888
889
890      .SBTTL " FETCH SUBROUTINE
891      ;:*****
892      ;:
893      ; THIS ROUTINE FETCHES THE MICRO INSTRUCTION AT THE SPECIFIED
894      ; ADDRESS BY DOING A MAINTENANCE RETURN TO THAT ADDRESS.
895      ;:-
896      ;:*****
897 013040 116700 173612 $FETCH:MOV B ARG2,R0 ; IS ADDRESS INDEXED?
898 013044 100412      BMI 1$ ; BRANCH IF NO
899 013046 066700 173532      ADD LOADAD,R0 ; ADD RELOCATION FACTOR
900 013052 017000 006616      MOV @LOOPTB(R0),R0 ; GET CURRENT INDEX VALUE
901 013056 005300      DEC R0 ;
902 013060 105767 173573      TSTB ARG2+1 ; IS ADDRESS ALPHA?
903 013064 100403      BMI 2$ ; BRANCH IF NO
904 013066 006300      ASL R0 ; MAKE INDEX A WORD INDEX
905 013070 000401      BR 2$ ;
906 013072 005000      1$: CLR R0 ; NO INDEXING SO CLEAR INDEX
907 013074 060067 173554      2$: ADD R0,ARG1 ; INDEX THE ADDRESS
908 013100 105767 173553      TSTB ARG2+1 ; IS ADDRESS ALPHA?
909 013104 100406      BMI 3$ ; BRANCH IF NO
910 013106 066767 173322 173540      ADD RELOC,ARG1 ; ADD RELOCATION FACTOR
911 013114 017767 173534 173532      MOV @ARG1,ARG1 ; GET THE WCS ADDRESS
912 013122 052737 002200 173032 3$: BIS #MNTRTN+CLRUWRD,@#CONMCR ; SET MAINTENANCE RETURN
913 013130      LOADID #ARG1,#USCSTK ; PUT THE ADDRESS ON THE MICRO STACK
914 013154 005767 173500      TST ARG3 ; CLEAR ROM NOP?
915 013160 001403      BEQ 4$ ; BRANCH IF NO
916 013162 042737 000200 173032      BIC #CLRUWRD,@#CONMCR ; CLEAR ROM NOP
917 013170      4$: SBCCLOCK ; POP THE USTACK INTO THE PC SAVE
918 013172      RETURN
919
920
921      .SBTTL " FLOAT ONE SUBROUTINE
922      ;:*****
923      ;:
924      ; THIS ROUTINE PLACES A ONE(1) IN THE BIT POSITION OF THE SPECIFIED
925      ; DATA WORD (32 BITS), ACCORDING TO THE CURRENT VALUE OF THE INDEX.
926      ;:-
927      ;:*****
928
929 013174 016700 173456 $FLTONE:MOV ARG2,R0 ; GET INDEX INTO INDEX TABLE POINTER TBL
930 013200 066700 173400      ADD LOADAD,R0 ; ADD RELOCATION FACTOR
931 013204 017000 006616      MOV @LOOPTB(R0),R0 ; GET THE CURRENT VALUE OF THE INDEX
932 013210 012702 000001      MOV #1,R2 ; INITIALIZE THE DATA
933 013214 005001      CLR R1 ;
934 013216 005300      1$: DEC R0 ; CHECK THE LOOP COUNT
935 013220 001403      BEQ 2$ ; BRANCH IF DONE

```



```

936 013222 006302    ASL R2 ; SHIFT THE ONE(1) BIT
937 013224 006101    ROL R1 ; THRGUGH 32 BITS
938 013226 000773    BR 1$ ; CONTINUE
939 013230 016703 173420 2$: MOV ARG1,R3 ; GET THE ADDRESS OF THE DATA
940 013234 066703 173174    ADD RELOC,R3 ; ADD THE RELOCATION CONSTANT
941 013240 010223    MOV R2,(R3)+ ; PUT THE PATTERN IN THE DATA WORDS
942 013242 010113    MOV R1,(R3) ;
943 013244    RETURN ; EXIT
944
945
946 .SBTTL " FLOAT ZERO SUBROUTINE
947 ;:*****
948 ;+
949 ; THIS ROUTINE FLOATS A ZERO(0) THROUGH A FIELD OF ONES(1). THE
950 ; DATA WORD MUST BE 32 BITS WIDE.
951 ;-
952 ;:*****
953
954 013246 016700 173404 $FLTZR0:MOV ARG2,R0 ; GET INDEX INTO INDEX TBL POINTER TBL
955 013252 066700 173326    ADD LOADAD,R0 ; ADD RELOCATION FACTOR
956 013256 017000 006616'  MOV @LOOPTB(R0),R0 ; GET THE CURRENT VALUE OF THE INDEX
957 013262 012702 177776    MOV #177776,R2 ; INITIALIZE THE FIELD OF 1'S
958 013266 012701 177777    MOV #-1,R1 ;
959 013272 005300    1$: DEC R0 ; CHECK THE LOOP COUNT
960 013274 001404    BEQ 2$ ; BRANCH IF DONE
961 013276 000261    SEC
962 013300 006102    ROL R2 ; SHIFT THE ZERO BY A BIT POSITION
963 013302 006101    ROL R1 ; 32 BITS WORTH
964 013304 000772    BR 1$ ; CONTINUE
965 013306 016703 173342 2$: MOV ARG1,R3 ; GET THE ADDRESS OF THE DATA
966 013312 066703 173116    ADD RELOC,R3 ; ADD TEH RELOCATION CONSTANT
967 013316 010223    MOV R2,(R3)+ ; SAVE THE FIELD OF 1'S
968 013320 010113    MOV R1,(R3) ;
969 013322    RETURN ; EXIT
970
971
972 .SBTTL " IF ERROR SUBROUTINE
973 ;:*****
974 ;+
975 ; THIS ROUTINE DOES THE FOLLOWING:
976 ;
977 ; 1) IT CHECKS THE ERROR FLAG AND IF NOT SET THE INTERPRETER
978 ; WILL CONTINUE SEQUENTIAL EXECUTION.
979 ; THE FOLLOWING ONLY OCCUR IF THE ERROR FLAG IS SET
980 ; 2) IF THE "BELL" FLAG IS SET, THE BELL WILL BE RUNG.
981 ; 3) IF THE "NER" FLAG IS NOT SET, THE ERROR MESSAGE WILL BE
982 ; TYPED.
983 ; 4) IF THE "HALTD" FLAG IS SET, EXECUTION WILL GO TO THE DIAGNOSTIC
984 ; MONITOR FOR OPERATOR INTERVENTION.
985 ; 5) IF THE "LOOP" FLAG IS SET, THE RETURN TPC WILL BE SET TO THE
986 ; ADDRESS IN LOCATION "$LPERR".
987 ; 6) IF AN ISOLATION ROUTINE IS SPECIFIED, (DETERMINED BY THE
988 ; SECOND ARGUMENT OF THE "IFERR" MACRO) THE TPC WILL BE SET
989 ; TO THE SPECIFIED ISOLATION ROUTINE ADDRESS.
990 ; 7) IF NONE OF THE ABOVE CONDITIONS ARE SATISFIED, THE INTERPRETER
991 ; WILL CONTINUE SEQUENTIAL EXECUTION.
992 ;-

```

IF ERROR SUBROUTINE

```

993      ;:*****
994
995 013324 105767 173056 $IFERR: TSTB $ERFLG ; ANY ERRORS?
996 013330 001505      BEQ ENDERR ; BRANCH IF NO
997 013332 032767 040000 173066 BIT #CTRLC,SWR ; CONTROL C FLAG SET?
998 013340 001401      DEQ 50$ ; BRANCH IF NO
999 013342      CALLMICMON
1000 013344 105767 173037 50$: TSTB $ERFLG+1 ; WAS THERE AN ERROR THIS LOOP?
1001 013350 001447      BEQ CHKLOP ; BRANCH IF NO
1002 013352 016767 173054 173326 MOV TPC,ERRCON ; SAVE THE TEST PC FOR ERROR CONTINUE
1003 013360 032767 000020 173040 BIT #BELL,SWR ; BELL ON ERROR?
1004 013366 001401      BEQ 1$ ; BRANCH IF NO
1005 013370      RINGBELL ; TYPE A BELL IF IT'S TIME
1006 013372 016767 173256 173202 1$: MOV ARG1,$ITEMB ; GET THE MESSAGE NUMBER
1007 013400 016767 173026 173006 MOV TPC,$ERRPC ; GET THE ERROR TPC
1008 013406 162767 000006 173000 SUB #6,$ERRPC ; BACK IT UP TO THE CALL
1009 013414 166767 173014 172772 SUB RELOC,$ERRPC ; SUBTRACT THE RELOCATION CONSTANT
1010 013422 032767 000010 172776 BIT #NER,SWR ; INHIBIT ERROR TYPEOUT?
1011 013430 001004      BNE 6$ ; BRANCH IF YES
1012 013432      TYPEERR ; GO TYPE THE ERROR MESSAGE
1013 013436 004767 174064      JSR PC,TYPDAT ; TYPE THE ERROR DATA
1014 013442 032767 000001 172756 6$: BIT #HALTD,SWR ; HALT ON ERROR DETECTION?
1015 013450 001407      BEQ CHKLOP ; BRANCH IF NO
1016 013452      E2ERR: MFPS $PSW ; SAVE THE PSW
1017 013460      CALLMICMON ; GO TO THE MICRO DIAGNOSTIC MONITOR
1018 013462      MTPS $PSW ; RESTORE THE PSW
1019 013470 032767 000004 172730 CHKLOP: BIT #LOOP,SWR ; LOOP ON ERROR?
1020 013476 001410      BEQ 4$ ; BRANCH IF NO
1021 013500 026767 172726 173200 CMP TPC,ERRCON ; AT THE CORRECT ERROR CALL?
1022 013506 001016      BNE ENDERR ; BRANCH IF NO
1023 013510 016767 172676 172714 MOV $LPERR,TPC ; SET THE TPC TO THE ERROR LOOP ADDRESS
1024 013516 000412      BR ENDERR ;
1025 013520 105767 172663 4$: TSTB $ERFLG+1 ; WAS THERE AN ERROR THIS LOOP?
1026 013524 001407      BEQ ENDERR ; BRANCH IF NO
1027 013526 016700 173124      MOV ARG2,R0 ; JUMP TO ISOLATION ROUTINE?
1028 013532 100404      BMI ENDERR ; BRANCH IF NO
1029 013534 066700 172674      ADD RELOC,R0 ; ADD RELOCATION CONSTANT TO TAG ADDRESS
1030 013540 010067 172666      MOV R0,TPC ; SET THE TPC TO THE TAG ADDRESS
1031 013544 105067 172437      ENDERR: CLR $ERFLG+1
1032 013550      RETURN

```

```

1033
1034
1035 .SBTTL INITIALIZE SUBROUTINE
1036 ;:*****
1037 ; THIS ROUTINE INITIALIZES THE STAR CPU BY SETTING THE INIT BIT
1038 ; IN THE MCR REGISTER.
1039 ;
1040 ;:*****

```

```

1041
1042 013552 052737 010202 173032 $INIT: BIS #INIT,CLRWRD+SBC,a#CONMCR
1043 013560 042737 010000 173032 BIC #INIT,a#CONMCR
1044 013566 052737 000001 173032 BIS #PROCEED,a#CONMCR ; ENSURE CLOCK IN OPT0
1045 013574      MTPS #0
1046 013602      RETURN

```

```

1047
1048
1049 .SBTTL KMX GENERATE SUBROUTINE

```

LOAD ID REGISTER SUBROUTINE

```

1107
1108 013730 116700 172723 $LDIDRE:MOVB ARG2+1,R0 ; IS DATA INDEXED?
1109 013734 100420 BMI 1$ ; BRANCH IF NO
1110 013736 066700 172642 ADD LOADAD,R0 ; ADD RELOCATION FACTOR
1111 013742 017000 006616' MOV @LOOPTB(R0),R0 ; GET CURRENT INDEX VALUE
1112 013746 005300 DEC R0 ; MAKE IT A WORD INDEX
1113 013750 006300 ASL R0 ; ...
1114 013752 126727 172700 000040 CMPB ARG2,#USCSTK ; IS THE ID REGISTER ONLY 16 BITS WIDE?
1115 013760 002404 BLT 3$ ; BRANCH IF NO
1116 013762 126727 172670 000042 CMPB ARG2,#USCADR ; IS THE ID REGISTER ONLY 16 BITS WIDE?
1117 013770 003403 BLT 2$ ; BRANCH IF YES
1118 013772 006300 3$: ASL R0 ; MAKE IT A LONG WORD INDEX
1119 013774 000401 BR 2$ ;
1120 013776 005900 1$: CLR R0 ; NO INDEXING SO CLEAR INDEX
1121 014000 066700 172650 2$: ADD ARG1,R0 ; GENERATE THE ADDRESS OF THE DATA
1122 014004 066700 172424 ADD RELOC,R0 ; ADD THE RELOCATION FACTOR
1123 014010 166700 172570 SUB LOADAD,R0 ; LOADID MACRO ADDS LOADAD AGAIN
1124 014014 116701 172636 MOVB ARG2,R1 ; GET ID BUS REGISTER ADDRESS
1125 014020 042701 177400 BIC #177400,R1 ;
1126 014024 LOADID R0,R1 ; LOAD THE REGISTER
1127 014044 RETURN
1128
1129
1130 .SBTTL " LOOP SUBROUTINE
1131 ;:*****
1132 ;:
1133 ; THIS ROUTINE INITIALIZES THE LOOP TABLE WITH THE FIRST VALUE,
1134 ; THE LAST VALUE, THE INCREMENT VALUE, AND THE LOOP ADDRESS.
1135 ;:-
1136 ;:*****
1137
1138 014046 116700 172602 $LOOP: MOVB ARG1,R0 ; GET INDEX INTO LOOP TABLE POINTER TABLE
1139 014052 010002 MOV R0,R2 ; SAVE IN R2
1140 014054 066700 172524 ADD LOADAD,R0 ; ADD RELOCATION CONSTANT
1141 014060 016090 006616' MOV LOOPTB(R0),R0 ; GET THE ADDRESS OF THE LOOP TABLE
1142 014064 016720 172570 MOV ARG3,(R0)+ ; PUT FIRST VALUE OF INDEX IN TABLE
1143 014070 062702 006610' ADD #LP1CNT,R2 ; GENERATE ADDRESS OF WORD TO SAVE THE COUNT
1144 014074 066702 172504 ADD LOADAD,R2 ; ...
1145 014100 016712 172554 MOV ARG3,(R2) ; INITIALIZE THE LOOP COUNT FOR TYPEOUT
1146 014104 105767 172545 TSTB ARG1+1 ; IS LAST VALUE NUMERIC?
1147 014110 100007 BPL 2$ ; BRANCH IF YES
1148 014112 016701 172540 MOV ARG2,R1 ; GET INDEX INTO INDEX TABLE FOR LAST VALUE
1149 014116 066701 172462 ADD LOADAD,R1 ; ADD RELOCATION FACTOR
1150 014122 017167 006616' 172526 MOV @LOOPTB(R1),ARG2 ; GET VALUE OF THE LAST VALUE OF THE LOOP
1151 014130 016720 172522 2$: MOV ARG2,(R0)+ ; PUT LAST VALUE OF INDEX IN TABLE.
1152 014134 005767 172522 TST ARG4 ; IS LOOP SIZE DEPENDENT?
1153 014140 001477 BEQ 60$ ; BRANCH IF NO
1154 014142 RDIDREG #USCDAT
1155 014152 016705 172306 MOV RDIDLO,R5 ; GET CONTENTS OF ID REG
1156 014156 012704 000001 MOV #1,R4 ; INIT THE SIZE TO 1K
1157 014162 032705 000040 BIT #40,R5 ; MORE THAN 1K?
1158 014166 001411 BEQ 75$ ; BRANCH IF NO
1159 014170 005204 INC R4 ; COUNT THE 6TH K
1160 014172 032705 000100 BIT #100,R5 ; IS THERE A 7TH K?
1161 014176 001405 BEQ 75$ ; BRANCH IF NO
1162 014200 005204 INC R4 ; COUNT IT
1163 014202 032705 000200 BIT #200,R5 ; 8TH K?

```

```

1164 014206 001401    BEQ 75$ ; BRANCH IF NO
1165 014210 005204    INC R4 ; COUNT IT
1166 014212 010467    172474 75$: MOV R4,WCSSIZE ; SAVE THE SIZE
1167
1168 014216 012705    000003 70$: MOV #3,R5 ; INIT VALUE FOR WCS SIZE EQL 1K
1169 014222 022767    000004 172462 CMP #4,WCSSIZE ; ARE THERE 4K OF WCS?
1170 014230 001002    BNE 10$ ; BRANCH IF NO
1171 014232 005005    CLR R5
1172 014234 000415    BR 50$
1173 014236 022767    000003 172446 10$: CMP #3,WCSSIZE ; 3 K WCS?
1174 014244 001003    BNE 20$ ; BRANCH IF NO
1175 014246 012705    000001 MOV #1,R5 ;
1176 014252 000406    BR 50$
1177 014254 022767    000002 172430 20$: CMP #2,WCSSIZE ; 2 K?
1178 014262 001002    BNE 50$ ; BRANCH IF ONLY ONE K
1179 014264 012705    000002 MOV #2,R5
1180
1181 014270 010002    50$: MOV R0,R2 ; GET BASE ADDRESS OF TABLE
1182 014272 026060    177774 177776 CMP -4(R0),-2(R0) ; IS FIRST LESS THAN LAST?
1183 014300 002404    BLT 5$ ; BRANCH IF YES
1184 014302 162702    000004 SUB #4,R2 ; GET ADDRESS OF VALUE
1185 014306 011204    MOV (R2),R4 ; GET FIRST VALUE
1186 014310 000403    BR 3$
1187 014312 162702    000002 5$: SUB #2,R2 ; GET ADDRESS OF LAST
1188 014316 011204    MOV (R2),R4 ; GET LAST VALUE
1189 014320 010403    3$: MOV R4,R3 ; SAVE MAX VALUE
1190 014322 006204    ASR R4 ; DIVIDE BY 4
1191 014324 006204    ASR R4 ;
1192 014326 005305    30$: DEC R5 ; CHECK IF DONE ADJUSTING LOOP VALUE
1193 014330 100402    BMI 40$ ; BRANCH IF YES
1194 014332 160403    SUB R4,R3 ; ADJUST VALUE
1195 014334 000774    BR 30$ ; CONTINUE
1196 014336 010312    40$: MOV R3,(R2) ; STORE THE NEW VALUE
1197 014340 012720    000001 60$: MOV #1,(R0)+ ; SET POSITIVE INCREMENT IN TABLE
1198 014344 026060    177772 177774 CMP -6(R0),-4(R0) ; IS FIRST LESS THAN LAST?
1199 014352 002403    BLT 1$ ; BRANCH IF YES
1200 014354 012760    177777 177776 MOV #-1,-2(R0) ; PUT NEGATIVE INCREMENT IN TABLE
1201 014362 016710    172044 1$: MOV TPC,(R0) ; PUT LOOP TPC IN TABLE
1202 014366 RETURN
1203
1204
1205 .SBTTL " MASK SUBROUTINE
1206 ;:.....;
1207 ; THIS ROUTINE MASKS THE SPECIFIED LOCATION WITH THE SPECIFIED
1208 ; MASK.
1209 ;:
1210 ;:.....;
1211
1212 014370 066767 172040 172256 $MASK: ADD RELOC,ARG1 ; GENERATE THE ADDRESS OF THE DATA
1213 014376 066767 172032 172252 ADD RELOC,ARG2 ; GENERATE THE ADDRESS OF THE MASK
1214 014404 017700 172246 MOV #ARG2,R0 ; GET THE MASK
1215 014410 005100 COM R0 ;
1216 014412 040077 172236 BIC R0,#ARG1 ; MASK THE DATA
1217 014416 RETURN
1218
1219
1220 .SBTTL " MOVE SUBROUTINE

```

MOVE SUBROUTINE

```

1221 :*****
1222 :*
1223 : THIS ROUTINE MOVES 16 BITS OF DATA POINTED TO BY ARG1 (INDEXED BY
1224 : ARG2) INTO THE LOCATION POINTED TO BY ARG3.
1225 :
1226 :*****
1227 :
1228 014420 065767 172010 172226 $MOVE: ADD RELOC,ARG1 ; RELOCATE SRC POINTER
1229 014426 116700 172224 MOVB ARG2,R0 ; IS SRC DATA INDEXED?
1230 014432 100413 BMI 1$ ; BRANCH IF NO
1231 014434 066700 172144 ADD LOADAD,R0 ; ADD RELOCATION FACTOR
1232 014440 017000 006616 MOV @LOOPTR(R0),R0 ; GET CURRENT INDEX VALUE
1233 014444 005300 DEC R0
1234 014446 006300 ASL R0 ; MAKE INDEX A WORD INDEX
1235 014450 105767 172203 TSTB ARG2+1 ; IS DATA TYPE LONG?
1236 014454 001403 BEQ 2$ ; BRANCH IF NO
1237 014456 006300 ASL R0 ; MAKE INDEX A 32 BIT INDEX
1238 014460 000401 BR 2$
1239 014462 005000 1$: CLR R0
1240 014464 060067 172164 2$: ADD R0,ARG1 ; OFFSET SRC ADDRESS
1241 014470 066767 171740 172162 ADD RELOC,ARG3 ; RELOCATE DST POINTER
1242 014476 017777 172152 172154 MOV @ARG1,@ARG3 ; MOVE THE SRC TO THE DST
1243 014504 RETURN
1244
1245
1246

```

SBTTL NEW TEST SUBROUTINE

```

1247 :*****
1248 :*
1249 : THIS ROUTINE PERFORMS THE FOLLOWING FUNCTIONS:
1250 :
1251 : 1) CHECKS THE LOOP ON SPECIAL TEST FLAG. IF SET, IT DETERMINES
1252 : IF THE SPECIFIED TEST IS FORWARD OR BACKWARD FROM THE CURRENT
1253 : TEST. IF FORWARD, THE TPC IS SET TO THE NEXT NEWTST STATEMENT.
1254 : IF BACKWARD, THE FILE IS CLOSED AND REOPENED AND THE TPC
1255 : IS SET TO THE NEWTST STATEMENT OF THE FIRST TEST.
1256 :
1257 :*****
1258 :*
1259 :
1260 014506 032767 040000 171712 $NEWTST:BIT #CTRLC,SWR ; CONTROL C FLAG SET?
1261 014514 001404 BEQ 12$ ; BRANCH IF NO
1262 014516 042767 040000 171702 BIC #CTRLC,SWR ;
1263 014524 CALLMICMON ; GO TO THE MICRO DIAGNOSTIC MONITOR
1264 014526 005067 171646 12$: CLR SUBTST ; INITIALIZE THE SUBTEST
1265 014532 032767 000200 171666 BIT #LOST,SWR ; LOOP ON SPECIAL TEST?
1266 014540 001440 BEQ 8$ ; BRANCH IF NO
1267 014542 026767 171624 171626 CMP $TSTNM,TESTNO ; ON THE TEST YET?
1268 014550 001404 BEQ 4$ ; BRANCH IF YES
1269 :
1270 :*
1271 : GET THE ADDRESS OF THE NEXT NEWTST STATEMENT AND GO TO IT
1272 :
1273 :
1274 014552 017767 172020 171630 6$: MOV @TSTPTR,$LPADR ; GET ADDRESS OF NEXT NEWTST
1275 014560 062767 000002 172010 ADD #2,TSTPTR ; INCREMENT THE TEST POINTER
1276 014566 066767 171642 171614 ADD RELOC,$LPADR ; ADD RELOCATION CONSTANT TO NEWTST ADR
1277 014574 005067 171606 CLR $ERFLG

```

NEW TEST SUBROUTINE

```

1278 014600 005067 172100 CLR $CHKFLG
1279 014604 000464 BR 3$ ; EXIT
1280
1281 ;+
1282 ; WE FOUND THE CORRECT TEST SO SET THE LOOP ADDRESSES AND START LOOPING.
1283 ; -
1284
1285 014606 016767 172066 171574 4$: MOV LOSTAD,$LPADR ; SET LOOP ADDRESS
1286 014614 016767 172060 171570 MOV LOSTAD,$LPERR ; SET THE DEFAULT ERROR LOOP ADDRESS
1287 014622 032767 004000 171576 BIT #CONT,SWR ; IS CONTINUE FLAG SET?
1288 014630 001460 BEQ 5$ ; BRANCH IF NO
1289 014632 042767 004200 171566 BIC #CONT+LOST,SWR ; CLEAR LOST FLAG
1290 014640 000454 BR 5$ ; EXIT
1291
1292 014642 032767 004000 71556 8$: BIT #CONT,SWR ; CONTINUE FLAG SET?
1293 014650 001403 BEQ 22$ ; BRANCH IF NO
1294 014652 042767 004100 171546 BIC #CONT+LOSS,SWR ; CLEAR LOSS FLAG
1295 014660 032767 020000 171542 22$: BIT #TSTSPAN,SWR1 ; WAS A TEST SPAN SPECIFIED?
1296 014666 001416 BEQ 11$ ; BRANCH IF NO
1297 014670 026767 171476 171500 CMP $TSTNM,TESTNO ; STARTED THE FIRST TEST YET?
1298 014676 002725 BLT 6$ ; BRANCH IF NO
1299 014700 001003 BNE 20$ ; BRANCH IF PAST IT
1300 014702 005267 172012 INC SPANFLAG ; EXECUTED FIRST TEST YET?
1301 014706 001737 BEQ 4$ ; BRANCH IF NO
1302 014710 026767 171460 171454 20$: CMP ENDSpan,$TSTNM ; COMPLETED LAST TEST YET?
1303 014716 001002 BNE 11$ ; BRANCH IF NO
1304 014720 21$: CALLMICMON ; DONE
1305 014722 000776 BR 21$ ; DON'T ALLOW CONTINUE
1306
1307 ;+
1308 ; WERE NOT LOOPING ON SPECIAL TEST SO SET THE
1309 ; LOOP ADDRESSES TO THIS TEST.
1310 ; -
1311
1312 014724 005067 171456 11$: CLR $ERFLG ; INITIALIZE THE ERROR FLAG
1313 014730 005067 171750 CLR $CHKFLG ; INITIALIZE THE VBUS CHECK FLAG
1314 014734 016767 171472 171446 MOV TPC,$LPADR ; SET THE LOOP ADDRESS
1315 014742 016767 171464 171442 MOV TPC,$LPERR ; SET THE DEFAULT ERROR LOOP ADDRESS
1316 014750 062767 000002 171620 ADD #2,TSTPTR ; INCREMENT THE TEST TABLE POINTER
1317 014756 016767 171450 171714 3$: MOV TPC,LOSTAD ; SAVE THIS ADDRESS INCASE LOOP ON SPECIAL
1318 ; TEST IS ASCERTED.
1319 014764 016767 171664 171400 MOV ARG1,$TSTNM ; UPDATE THE TEST NUMBER
1320 014772 016767 171412 171432 5$: MOV $LPADR,TPC ; SET THE TEST PC
1321 015000 005067 171604 7$: CLR LP1CNT
1322 015004 005067 171602 CLR LP1CNT+2
1323 015010 005067 171600 CLR LP1CNT+4
1324 015014 RETURN ; EXIT
1325
1326
1327 .SBTTL " NOP SUBROUTINE
1328 ; .....
1329 ;+
1330 ; THIS ROUTINE DOES NOTHING.
1331 ; -
1332 ; .....
1333
1334 015016 $NOOP: RETURN

```

" NOP SUBROUTINE

1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391

.SBTTL " READ ID BUS SUBROUTINE  
:\*\*\*\*\*  
: THIS ROUTINE READS THE SPECIFIED ID BUS REGISTER AND SAVES  
: IT IN LOCATIONS "IDLOW" AND "IDHIGH".  
:-  
:\*\*\*\*\*

015020 \$READID:RDIDREG ARG1 ; READ THE SPECIFIED REGISTER  
015030 RETURN

.SBTTL " REPORT SUBROUTINE  
:\*\*\*\*\*  
: THIS ROUTINE IS USED TO TYPE THE NAMES OF THE FAILING MODULES  
: WHEN DIAGNOSIS IS FINISHED. IT PERFORMS THE FOLLOWING FUNCTIONS:  
: 1) IF THE "NER" FLAG IS CLEAR, THE SPECIFIED LIST OF  
: MODULE NAMES IS TYPED, OTHERWISE NOTHING IS TYPED.  
: 2) IF THE "HALTI" FLAG IS SET, EXECUTION WILL RETURN  
: TO THE DIAGNOSTIC MONITOR.  
: 3) IF THE "LOOP" FLAG IS SET, THE TPC WILL BE  
: SET TO THE CONTENTS OF "\$LPERR", OTHERWISE IT IS  
: UNCHANGED.  
:-  
:\*\*\*\*\*

015032 032767 J0010 171366 \$REPORT:BIT #NER,SWR ; NO ERROR REPORT?  
015040 001007 BNE 3\$ ; BRANCH IF YES  
015042 TYPEMOD #ARG1 ; GO TYPE THE MODULE NAMES  
015060 032767 000002 171340 3\$: BIT #HALTI,SWR ; HALT ON ISOLATION?  
015066 001401 BEQ 4\$ ; BRANCH IF NO  
015070 CALLMICMON ; GO TO THE MICRO DIAGNOSTIC MONITOR  
015072 032767 000004 171326 4\$: BIT #LOOP,SWR ; LOOP ON ERROR?  
015100 001404 BEQ 5\$ ; BRANCH IF NO  
015102 016767 171304 171322 MOV \$LPERR,TPC ; SET TPC TO ERROR LOOP ADDRESS  
015110 000422 BR 6\$ ; EXIT  
015112 032767 000040 171306 5\$: BIT #ERABT,SWR ; IS THE ERROR ABORT FLAG SET?  
015120 001010 BNE 7\$ ; BRANCH IF YES  
015122 016767 171560 171302 MOV ERRCON,TPC ; SET THE TPC  
015130 005067 171252 CLR \$ERFLG ; CLEAR THE ERROR FLAG  
015134 005067 171544 CLR \$CHKFLG ; AND THE VBUS ERROR FLAG  
015140 000406 BR 6\$  
015142 017767 171430 171262 7\$: MOV @TSTPTR,TPC ; SET TPC TO ADDRESS OF NEXT "NEWTST"  
015150 066767 171260 171254 ADD RELUC,TPC ; ADD RELOCATION FACTOR  
015156 6\$: RETURN ; EXIT

.SBTTL " RESET SUBROUTINE  
:\*\*\*\*\*  
: THIS ROUTINE EXECUTES AN LSI-11 RESET INSTRUCTION  
:-

## " RESET SUBROUTINE

```

1392 ;*****
1393
1394 015160 $RESET: RESET$
015160 104020 EMT R$SET
1395 015162 RETURN
1396
1397
1398
1399 .SBTTL " SET PSW SUBROUTINE
1400 ;*****
1401 ;+
1402 ; THIS ROUTINE IS USED TO SET A PRIORITY LEVEL IN THE LSI-11 PROCESSOR
1403 ; STATUS WORD.
1404 ;-
1405 ;*****
1406
1407 015164 $SETPSW:MTPS ARG1 ; SET THE LEVEL
1408 015172 RETURN
1409
1410
1411 .SBTTL " SET VECTOR ROUTINE
1412 ;*****
1413 ; THIS ROUTINE LOADS THE ADDRESS SPECIFIED BY ARG1 WITH THE ADDRESS
1414 ; OF THE EXPECTED TRAP ROUTINE.
1415 ;
1416 ;*****
1417 015174 012777 010314 171452 $SETVEC:MOV #TRAP,@ARG1 ; PUT TRAP ADDRESS IN IT
1418 015202 066777 171376 171444 ADD LOADAD,@ARG1 ; ADD RELOCATION CONSTANT
1419 015210 012767 000401 171170 MOV #401,$ERFLG ; SET THE ERROR FLAG
1420 015216 RETURN
1421
1422
1423 .SBTTL " SKIP SUBROUTINE
1424 ;*****
1425 ;+
1426 ; THIS ROUTINE SETS THE TPC TO THE SPECIFIED ADDRESS.
1427 ;-
1428 ;*****
1429
1430 015220 016767 171430 171204 $SKIP: MOV ARG1,TPC ; GET THE ADDRESS TO SKIP TO
1431 015226 066767 171202 171176 ADD RELOC,TPC ; ADD THE RELOCATION CONSTANT
1432 015234 RETURN
1433
1434
1435 .SBTTL " SKIP IF ERROR SUBROUTINE
1436 ;*****
1437 ;+
1438 ; THIS ROUTINE SETS THE TPC TO THE SPECIFIED ADDRESS IF THE
1439 ; ERROR FLAG IS SET.
1440 ;-
1441 ;*****
1442
1443 015236 $SKIPERROR:
1444 015236 105767 171144 TSTB $ERFLG ; IS THE ERROR FLAG SET?
1445 015242 001406 BEQ 1$ ; BRANCH IF NO
1446 015244 016767 171404 171160 MOV ARG1,TPC ; GET THE ADDRESS TO SKIP TO
1447 015252 066767 171156 171152 ADD RELOC,TPC ; ADD THE RELOCATION CONSTANT

```



1448 015260 1\$: RETURN

1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459

.SBTTL " SP ADDRESS GENERATE SUBROUTINE  
;\*\*\*\*\*;  
; THIS ROUTINE GENERATES A SPA FIELD IN THE SPECIFIED MICRO INSTRUCTION  
; EQUAL TO THE CURRENT LOOP COUNT MINUS 1.  
;-  
;\*\*\*\*\*

1460 015262 016700 171366 \$SPAGEN:MOV ARG1,R0 ; GET ADDRESS OF MICRO INSTRUCTION  
1461 015266 066700 171142 ADD RELOC,R0 ; ADD RELOCATION FACTOR  
1462 015272 062700 000004 ADD #4,R0 ; SELECT THE 3TH 16 BIT WORD (KMX FILED STARTS AT BIT35)  
1463 015276 016701 171354 MOV ARG2,R1 ; GET INDEX INTO INDEX TABLE POINTER TABLE  
1464 015302 066701 171276 ADD LOADAD,R1 ;  
1465 015306 017101 006616 MOV @LOOPTB(R1),R1 ; GET THE CURRENT INDEX VALUE  
1466 015312 005301 DEC R1 ; ADJUST  
1467 015314 006301 ASL R1 ; PUT IN SPA FIELD POSITION  
1468 015316 006301 ASL R1 ; ---  
1469 015320 006301 ASL R1 ; ---  
1470 015322 042710 000170 BIC #170,(R0) ; CLEAR CURRENT SPA FIELD  
1471 015326 050110 BIS R1,(R0) ; INSERT NEW FIELD VALUE  
1472 015330 RETURN ; EXIT

1473  
1474  
1475  
1476

.SBTTL " SUBTEST SUBROUTINE  
;\*\*\*\*\*;  
; THIS ROUTINE INCREMENTS THE CURRENT VALUE OF THE SUBTEST COUNTER.  
;-  
;\*\*\*\*\*

1483 015332 005267 171042 \$SUBTEST:INC SUBTST ; INCREMENT THE COUNTER  
1484 015336 005067 171246 CLR LP1CNT ; CLEAR THE LOOP COUNTS FOR TYPEOUT  
1485 015342 005067 171244 CLR LP1CNT+2  
1486 015346 005067 171242 CLR LP1CNT+4  
1487 015352 005067 171030 CLR \$ERFLG ; AND THE ERROR FLAG  
1488 015356 012700 010324 MOV #CATCH,R0 ; GET ADDRESS OF TRAP CATCHER  
1489 015362 066700 171216 ADD LOADAD,R0 ; ADD RELOCATION FACTOR  
1490 015366 010037 000004 MOV R0,#4 ; SET THE TRAP TO 4 VECTOR  
1491 015372 012700 010466 MOV #CATCHI,R0 ; GET ADDRESS OF INTERRUPT CATCHER  
1492 015376 066700 171202 ADD LOADAD,R0 ; ADD RELOCATION FACTOR  
1493 015402 010037 000300 MOV R0,#300 ;  
1494 015406 010037 000304 MOV R0,#304  
1495 015412 RETURN

1496  
1497

.SBTTL " TEST V BUS SUBROUTINE  
;\*\*\*\*\*;  
; THIS ROUTINE TESTS THE SPECIFIED BIT OF THE V BUS TO BE  
; THE SAME AS THE SPECIFIED VALUE. IF THEY ARE DIFFERENT, THE  
; "ERFLG" AND THE "CHKFLG" ARE SET.  
;-

1498  
1499  
1500  
1501  
1502  
1503  
1504

## " TEST V BUS SUBROUTINE

```

1505      ;:*****
1506
1507 015414 005067 171264 $TSTVB: CLR $CHKFLG
1508 015420 004767 172620 JSR PC,$READVB ; GET THE V BUS
1509 015424 016700 171224 MOV ARG1,R0 ; GET THE ADDRESS OF THE BIT TABLE
1510 015430 066700 171000 ADD RELGC,R0 ; ADD THE RELOCATION FACTOR
1511 015434 016701 171216 MOV ARG2,R1 ; IS IT INDEXED?
1512 015440 100414 BMI 2$ ; BRANCH IF NO
1513 015442 066701 171136 ADD LOADAD,R1 ; ADD RELOCATION CONSTANT
1514 015446 017101 006616 MOV @LOOPTB(R1),R1 ; GET THE CURRENT INDEX
1515 015452 005301 1$: DEC R1 ; DONE INDEXING?
1516 015454 001406 BEQ 2$ ; BRANCH IF YES
1517 015456 011002 MOV (R0),R2 ; GET THE SIZE OF THIS ENTRY
1518 015460 006302 ASL R2 ; CORRECT FOR WORD INDEXING
1519 015462 060200 ADD R2,R0 ; GENERATE ADDRESS OF NEXT TBL ENTRY
1520 015464 062700 000002 ADD #2,R0 ;
1521 015470 000770 BR 1$ ; CONTINUE
1522 015472 012001 2$: MOV (R0)+,R1 ; GET THE # OF ENTRYS IN THIS TABLE
1523 015474 012702 007164 3$: MOV #VBBUFF,R2 ; GET START ADDRESS OF VB BUFFER
1524 015500 066702 171100 ADD LOADAD,R2 ; ADD RELOCATION FACTOR
1525 015504 005067 170706 CLR GOODDAT ; INITIALIZE LOCATION FOR EXPECTED VALUE
1526 015510 116003 000001 MOVB 1(R0),R3 ; GET BIT ID ADM VALUE
1527 015514 042703 177400 BIC #177400,R3 ; CLEAR SIGN EXTEND
1528 015520 000241 CLC ;
1529 015522 106103 ROLB R3 ; PUT VALUE IN THE C BIT
1530 015524 005567 170666 ADC GOODDAT ; PUT VALUE IN BIT<0> OF GOODDAT
1531 015530 006203 ASR R3 ; PUT BIT NUMBER IN BITS<6:0>
1532 015532 042703 000200 BIC #200,R3 ; ...
1533 015536 060302 ADD R3,R2 ; SELECT THE BYTE IN THE BUFFER
1534 015540 006303 ASL R3 ; PUT BIT NUMBER IN BITS<10:4>
1535 015542 006303 ASL R3 ; ...
1536 015544 006303 ASL R3 ; ...
1537 015546 006303 ASL R3 ; ...
1538 015550 050367 170642 BIS R3,GOODDAT ; PUT IN BITS<9:3> OF GOODDAT
1539 015554 111004 MOVB (R0),R4 ; GET THE CHANNEL NUMBER
1540 015556 010405 MOV R4,R5 ; SAVE CHANNEL NUMBER
1541 015560 000305 SWAB R5 ; PUT CHANNEL NUMBER IN HIGH BYTE
1542 015562 006305 ASL R5 ; PUT CHANNEL NUMBER IN BITS<14:12>
1543 015564 006305 ASL R5 ; ...
1544 015566 006305 ASL R5 ; ...
1545 015570 006305 ASL R5 ; ...
1546 015572 050567 170620 BIS R5,GOODDAT ; INSERT INTO GOODDAT
1547 015576 016767 170614 170616 MOV GOODDAT,BADDAT ; COPY CHANNEL AND BIT NUMBER TO BAD DATA
1548 015604 042767 000001 170610 BIC #1,BADDAT ; GET READY TO INSERT RECEIVED VALUE OF BIT
1549 015612 111205 MOVB (R2),R5 ; GET THE BYTE FROM THE BUFFER
1550 015614 012702 000376 MOV #376,R2 ; INITIALIZE R2 TO MASK BIT 0
1551 015620 005304 7$: DEC R4 ; CONVERT CHANNEL NUMBER INTO A BIT POSITION
1552 015622 100403 BMI 8$ ; BRANCH IF DONE
1553 015624 000261 SEC ;
1554 015626 106102 ROLB R2 ; SHIFT THE MASK
1555 015630 000773 BR 7$ ; CONTINUE
1556 015632 000241 8$: CLC ;
1557 015634 140205 BICR R2,R5 ; CLEAR THE UNWANTED BITS FROM THE RECEIVED BYTE
1558 015636 001403 10$: BEQ 9$ ; NOW SHIFT RIGHT UNTIL THE WANTED BIT IS IN BIT POSITION 0
1559 015640 000241 CLC ;
1560 015642 106005 RORB R5 ;
1561 015644 000774 BR 10$ ; KEEP SHIFTING

```

## " TEST V BUS SUBROUTINE

```

1562 015646 006105    9$: ROL R5 ; BRING THE BIT BACK TO POSITION 0
1563 015650 150567 170546    BISB R5,BADDAT ; INSERT THE BIT INTO THE BAD DATA
1564 015654 026767 170536 170540    CMP GOODDAT,BADDAT ; SEE IF EXPECTED AND RECEIVED ARE EQUAL
1565 015662 001004    BNE 4$ ; BRANCH IF DIFFERENT
1566 015664 005720    TST (R0)+ ; BUMP R0 TO NEXT TABLE ENTRY
1567 015666 005301    DEC R1 ; DECREMENT THE LOOP COUNT
1568 015670 001301    BNE 3$ ; CONTINUE
1569 015672 000410    BR 6$ ; ALL BITS ARE OK
1570 015674 005067 171010    4$: CLR DATTYPE ; SET 16 BIT DATA TYPE
1571 015700 012767 000401 170500    MOV #401,$ERFLG ; SET THE ERROR FLAG
1572 015706 005267 170772    5$: INC $CHKFLG ;
1573 015712 001775    BEQ 5$ ;
1574 015714    6$: RETURN

```

```

1575
1576
1577 .SBTTL " TYPE WCS SIZE SUBROUTINE
1578 ;*****+
1579 ; THIS ROUTINE TYPES THE NUMBER OF WCS MODULES AS A FUNCTION
1580 ; OF THE CONTENTS OF "BADDAT" WHICH IS ASSUMED TO BE THE CONTENTS
1581 ; OF THE WCS DATA REGISTER.
1582 ;-
1583 ;*****
1584

```

```

1585 015716    $TYPESIZE:
1586 015716 032767 000010 170502    BIT #NER,SWR ; INHIBIT ERROR TYPEOUT?
1587 015724 001143    BNE 5$ ; BRANCH IF YES
1588 015726 032767 000300 170472    BIT #LOSS+LOST,SWR ; LOOPING ON THIS TEST?
1589 015734 001403    BEQ 7$ ; BRANCH IF NO
1590 015736 005267 170754    INC SIZEFLG ; TYPED IT YET?
1591 015742 001134    BNE 5$ ; BRANCH IF YES
1592 015744    7$: TYPE #CRLF,ASCII
1593 015764 032767 000017 170430    BIT #17,BADDAT ; IS THERE 4 PCS MODULES?
1594 015772 001431    BEQ 20$ ; BRANCH IF YES
1595 015774    TYPE #MSG3 ; TYPE ILLEGAL CONFIG. MESSAGE
1596 016012    TYPES #BADDAT,HEX ; TYPE CONTENTS OF DATA REG
1597 016032    6: TYPE #CRLF,ASCII
1598 016052    CALLMICMON ; GO TO THE MONITOR
1599 016054 000467    BR 5$ ; EXIT
1600 016056 005000    20$: CLR R0
1601 016060    TYPE #MSG1 ; TYPE THE SIZE MESSAGE
1602 016076 032767 000020 170316    BIT #20,BADDAT ; IS THE 5TH K THERE?
1603 016104 001011    BNE 1$ ; BRANCH IF YES
1604 016106    TYPES #ZERO,HEX ; TYPE A ZERO
1605 016126 000741    BR 6$ ; GO TO THE MONITOR
1606 016130 005200    1$: INC R0
1607 016132 032767 000040 170262    BIT #40,BADDAT ; IS THE 6TH K THERE?
1608 016140 001401    BEQ 3$ ; BRANCH IF NO
1609 016142 005200    INC R0
1610 016144 032767 000100 170250    3$: BIT #100,BADDAT ; IS THERE A 7TH K THERE?
1611 016152 001401    BEQ 4$ ; BRANCH IF NO
1612 016154 005200    INC R0 ;
1613 016156 032767 000200 170236    4$: BIT #200,BADDAT ; 8TH K?
1614 016164 001401    BEQ 10$
1615 016166 005200    INC R0
1616 016170 010067 170516    10$: MOV R0,WCSIZE ; SAVE THE NUMBER OF MODULES
1617 016174    TYPES #WCSIZE,HEX ; TYPE THEM
1618 016214    TYPE #CRLF,ASCII

```

```

1619 016234    S$: RETURN    ; EXIT
1620
1621
1622
1623
1624
1625
1626          ;+
1627          ; THE FOLLOWING LOCATION CONTROL ALGORITHM IS NECESSARY TO MAKE THE
1628          ; LENGTH OF THE HARDCORE MONITGR AN INTEGER NUMBER OF SECTORS LONG.
1629          ; THIS IS REQUIRED SO THAT WHEN THE MONITOR IS SWAPPED OUT AND EVENTUALLY
1630          ; READ BACK IN, THE TEST STREAM OVERLAY DOES NOT GET OVER WRITTEN.
1631          ;
1632          ; THE ALGORITHM ALSO DISPLACES THE BUFFER AREA FOR THE TEST STREAM
1633          ; OVERLAYS TO START AT THE LOCATION SPECIFIED BY "$ENDADR".
1634          ; SO THAT THE PARSER FILE CAN BE READ IN WITHOUT DESTROYING THE
1635          ; TEST STREAM OVERLAY.
1636          ;-
1637
1638 016236      TEMP=.
1639 007446      X=<TEMP-HEAD> ; LENGTH OF THIS FILE
1640 000044      X=X&177    ; IN SECTORS
1642 000132      X=200-X    ; NUMBER OF BYTES TO MAKE EVEN SECTOR
1643 016236      FILL X
1645 016370      TEMP=.
1646 007600      X=<TEMP-HEAD>
1647 006570      .=HEAD
1648 006570 007600 .WORD <TEMP-HEAD>
1649 016370      .=TEMP
1650 002010      X=$ENDADR-<X-OFFSET>-200 ; NUMBER OF BYTES TO ADDRESS $ENDADR
1652 016370      FILL X
1654
1655          ;+
1656          ; THE TEST STREAM OVERLAYS START HERE. THEY ARE A MAXIMUM OF 1536.
1657          ; BYTES LONG AND A MINIMUM OF 128 BYTES LONG.
1658          ;-
1659
1660
1661 020400 000000 END: .WORD
1662 000001      .END

```

## Symbol table

ACCMNT=	000026	CONT =	004000	GOCHA2=	000012	MAXCNT	006702R	P1LR =	000075
ACCST =	006027	CONTXD=	000007	GOODDA	006416R	MAY =	000025	Q.SV =	000057
ACCO =	000024	CONTXS=	000006	GOTUPC	006470R	MAY16K	007462R	RADGET=	000010
ACC1 =	000025	CPURUN=	003400	HALTD =	000001	MAY4 =	000046	RADHEX=	000020
ADAOFF=	000126	CSBUS =	000050	HALTI =	000002	MAY4K	007442R	RADOCT=	000010
ADAPT	007516R	CTRLC =	040000	HARDC =	000001	MAY6 =	000035	RDIDHI	006466R
ARG1	006654R	DAP =	000004	HARDCO	011000R	MAY8 =	000047	RDIDLO	006464R
ARG2	006656R	DATTYP	006710R	HCMONI=	000000	MBA =	000054	RDYIE =	000100
ARG3	006660R	DBLFLG	006676R	HEAD =	006570R	MCN =	000023	READSC=	000004
ARG4	006662R	DBP =	000010	IBCLKS=	000012	MDMTYP=	000022	RELOC	006434R
ARG5	006664R	DCP =	000005	IBDAT =	000000	MDT =	000024	RMWRON=	000015
ARG6	006666R	DDP =	000006	IBICT =	000013	MIC1FL=	002000	ROMO =	173000
BADDAT	006422R	DEP =	000007	IBNIN =	000011	MIC2FL=	004000	ROM1 =	173002
BELL =	000020	DICMD =	001000	IBTOD =	000001	MNTRTN=	002000	RUNFLG=	000002
BRTBL	012424R	DIRECT=	000014	ICL =	000012	MODADR	006446R	RWDQ =	000010
BUSES	007510R	DIRERR=	000100	IDADR	006462R	MODLNK	006564R	RXDNE =	173014
BUSOFF=	000120	DISPAT	007060R	IDBUS =	000051	MODULE	007370R	RXVEC =	000304
BYL =	000044	DNEIE =	000040	IDCS =	173030	MPC =	000037	R\$SET =	000020
BYU =	000045	DRA =	000055	IDCYCL=	100000	MPFC =	000026	R6 =	000006
B1FULL=	000004	D.SV =	000056	IDDAT	006460R	MPGOCH=	000024	R7 =	000007
B1INUS=	000400	END	020400R	IDDATA=	173010	MPI =	000036	SBC =	000002
B2FULL=	000200	ENDADR	006672R	IDDATA=	173006	MPS =	000040	SBH =	000017
B2INUS=	000040	ENDERR	013544R	IDMAIN=	000200	MSB =	000022	SBICP =	000036
CAM =	000013	ENDSPA	006374R	IDP =	000021	MSBE =	000043	SBIERR=	000031
CATCH	010324R	EQ. =	000000	IDREGH=	000001	MSFC =	000032	SBIFLT=	000033
CATCHI	010466R	ERABT =	000040	IDREGL=	177777	MSGA	006502R	SBIMAT=	000035
CATEX	010362R	ERRCON	006706R	IDWRIT=	000100	MSGB	006510R	SBISCM=	000034
CCPT0 =	000200	ESP =	000051	IINDX	006624R	MSGC	006516R	SBISIL=	000030
CCPT1 =	000100	E2ERR	013452R	INIT =	010000	MSGOCH=	000030	SBITO =	000032
CCPT2 =	000040	FAD =	000033	IRC =	000020	MSG1	006726R	SBL =	000016
CCPT3 =	000020	FAIL	012332R	ISP =	000054	MSG2	006750R	SBR =	000046
CDM =	000014	FAILCH=	000016	ITSTPT=	000004	MSG24	007046R	SCBB =	000073
CEH =	000011	FCHAI1=	000020	JINDX	006634R	MSG3	006776R	SCTSPA=	040000
CES =	000014	FCHAI2=	000022	KEYBUF	006722R	MSG4	007022R	SECTNO	006404R
CHAR =	000002	FCHR1 =	000015	KEYCOD	006542R	MSKFLG	006674R	SECTOR	006556R
CHKLOP	013470R	FCHR2 =	000002	KEYERR=	020000	M256KO=	000116	SGLINS	010602R
CHKSWI=	000023	FCT =	000034	KEYQUE=	010000	M4KOFF=	000052	SINST =	000400
CH1 =	000001	FILPTR	006436R	KINDX	006644R	M6KOFF=	000072	SIR =	000016
CH2 =	000000	FIRSTC=	000000	KSP =	000050	M64KOF=	000114	SIXSPC	006534R
CH3 =	000000	FLPYMS=	003000	LCANW =	000014	NER =	000010	SIZEFL	006716R
CIA =	000056	FLPYON=	010000	LCWRON=	000016	NE. =	000004	SLFTST=	000004
CIB =	000000	FLPY2 =	001000	LDCONS=	000021	NOCHAR=	000003	SLR =	000076
CLK =	000026	FLPY3 =	002000	LOADAD	006604R	OFFSET=	006370	SOMM =	000100
CLKFST=	000010	FLPY4 =	003000	LOADCN=	000006	OPENFL=	000003	SPANFL	006720R
CLKSLO=	000020	FMH =	000031	LOOP =	000004	OPNFL1=	000011	SPARE1=	173004
CLKSTP=	000040	FMIDHI=	173026	LOOPTB	006616R	OVRADR	006440R	SPARE2=	173012
CLRURR=	000200	FMIDLO=	173024	LOPF! =	000000	OVRBYT	006442R	SRCADR	006450R
CNVEPT=	000007	FML =	000032	LOPFJ =	000000	PARSER=	000010	SSP =	000052
COM =	100000	FNM =	000030	LOPFK =	000000	PASCNT	006546R	STADR	006670R
COMSPC	006476R	FPA =	010000	LOSLNK	006552R	PASS	012346R	STS =	000004
CONACK=	000200	FPDA =	000055	LOSS =	000100	PCBB =	000072	STSNO	006456R
CONCM =	001000	FPSYNC	006560R	LOSSEC	006554R	PCS =	000003	SUBTST	006400R
CONID =	000003	FPYVEC	006550R	LOST =	000200	PROCEE=	000001	SWR	006426R
CONMCR=	173032	FPG =	000010	LOSTAD	006700R	PSL =	000017	SWR1	006430R
CONMCS=	173034	FR1 =	000020	LP1CNT	006610R	POBR =	000044	TBDAT =	000020
CONRXD=	000005	GOCHAI=	000004	LSTFIL=	000001	POLR =	000074	TBERO =	000022
CONRXS=	000004	GOCHA1=	000006	MAT =	000041	P1BR =	000045	TBER1 =	000023

## Symbol table

TBLEND= 007164R	TREAD = 000002	VBLOAD= 000002	\$ENDLO 012604R	\$NOOP 015016R
TBLHEA= 007060R	TRS = 000027	VBUS = 000052	\$ENDOV 012702R	\$PASS 006370R
TBLSIZ= 000042	TSTMFG= 000024	VECT = 000015	\$ERFLG 006406R	\$PSW 006544R
TBM = 000015	TSTPTR 006576R	W = 006370R	\$ERRLO 013030R	\$READI 015020R
TEMP = 016370R	TSTSPA= 020000	WCS = 000002	\$ERRPC 006414R	\$READV 010244R
TEMP0 = 000060	TWOSPC 006724R	WCSADR 006452R	\$FER = 000001	\$REPOR 015032R
TEMP1 = 000061	TWRITE= 000001	WCSCNT 006454R	\$FETCH 013040R	\$RESET 015160R
TEMP2 = 000062	TXRDY = 173016	WCSSIZ 006712R	\$FLAG 006606R	\$SCTNO 006402R
TEMP3 = 000063	TXVEC = 000300	WCS2K = 000042	\$FLTON 013174R	\$SECTO= 000000
TEMP4 = 000064	TYPADR 006444R	WRITSC= 000005	\$FLTZR 013246R	\$SETPS 015164R
TEMP5 = 000065	TYPDAT 007526R	X = 002010	\$FNF = 000002	\$SETVE 015174R
TEMP6 = 000066	TYPVER 010526R	XX = 000176	\$FNR = 000003	\$SKIP 015220R
TEMP7 = 000067	TYP1 = 000012	Y = 007344R	\$FOR = 000004	\$SKIPE 015236R
TEMP8 = 000070	TYP2 = 000013	Z = 000001	\$IFERR 013324R	\$SN = 000001
TEMP9 = 000071	UBA = 000053	ZERO 006714R	\$INIT 013552R	\$SPAGE 015262R
TERMIN 006562R	UPC12 = 001000	\$BLKMI 011464R	\$ITEMB 006602R	\$SUBTE 015332R
TESTNO 006376R	USC = 000001	\$CHKFL 006704R	\$KMXGE 013604R	\$TBSY = 000005
TESTST= 000002	USCADR= 000042	\$CHKPN 011656R	\$LDCA 013654R	\$TCTC = 000006
TINIT = 000000	USCBRK= 000041	\$CLOCK 012010R	\$LDIDR 013730R	\$TEMP1= 007526R
TMERTR= 000017	USCDAT= 000043	\$CMPCA 012042R	\$LOOP 014046R	\$TER = 000007
TOIDHI= 173022	USCSTK= 000040	\$CMPPC 012434R	\$LPADR 006410R	\$TMP0 006600R
TOIDLO= 173020	USP = 000053	\$CPCAM 012036R	\$LPERR 006412R	\$TN = 000001
TPC 006432R	VBBUFF 007164R	\$CRLF 006472R	\$MASK 014370R	\$TSTNM 006372R
TPCINI= 000034	VBCLK = 000001	\$ENDAD= 020400	\$MOVE 014420R	\$TSTVB 015414R
TRAP 010314R	VBCTRL= 173036	\$ENDHC 012534R	\$NEWTs 014506R	\$TYPsI 015716R
TRAPVE= 000034				

. ABS. 000000 000 (RW,I,GBL,ABS,OVR)  
 020402 001 (RW,I,LCL,REL,CON)

Errors detected: 0

## \*\*\* Assembler statistics

Work file reads: 146  
 Work file writes: 142  
 Size of work file: 30477 Words ( 120 Pages)  
 Size of core pool: 19978 Words ( 76 Pages)  
 Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:02:37.23  
 ,ESKAC/-SP=ESKABMAC.MLB/ML,ENABLELST.MAC,ESKAC.MAC

