

This image shows a microfiche card with a grid of frames. The frames contain data, likely from a database or a list, organized in columns and rows. The text is very small and difficult to read, but it appears to be a structured list of information. The card is oriented vertically, and the frames are arranged in a regular grid pattern. The data in the frames is organized into columns, with some frames containing multiple lines of text. The overall appearance is that of a standard microfiche card used for data storage and retrieval.

.NLIST SEQ,LOC,BIN,TOC  
.REM\_

IDENTIFICATION

PRODUCT CODE: AC-F172B-MC  
PRODUCT NAME: CR11ABO 11ST STND ALN SP  
DATE CREATED: 15 NOVEMBER 1978  
UPDATED : 15 JUNE 1979  
MAINTAINER: C.S.S. LOW VOLUME PRODUCTS GROUP  
AUTHOR: ROBERT J. COLLINS / W. WEISKE  
UPDATED BY : JAMES M. DUPRE / WILLIAM M. WEISKE

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

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

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

COPYRIGHT (c) 1978, 1979 BY DIGITAL EQUIPMENT CORPORATION

## 1.0 ABSTRACT

THE DIP11-A SINGLE-INTERFACE DIAGNOSTIC IS USED TO EXERCISE THE LOGIC OF A DIP11-A INTERFACE (IIST) IN STANDALONE MODE AND REPORT ANY MALFUNCTIONS ON THE CONSOLE TERMINAL. MOST OF THE TESTING IS PERFORMED IN WRAPAROUND MAINTENANCE MODE, BUT SOME TESTS CAN BE EXECUTED IN NORMAL MODE. THE INTERFACE UNDER TEST MAY BE CONNECTED TO THE IIST INTERCONNECTING BUS AS LONG AS OTHER INTERFACES ARE NOT TRANSMITTING INFORMATION ADDRESSED TO THE INTERFACE UNDER TEST.

## 2.0 MINIMUM EQUIPMENT

- A. PDP-11 WITH 12K AND A CONSOLE TERMINAL
- B. DIP11-A (IIST) INTERFACE (M8717 MODULE)

## 3.0 PRELIMINARY OPERATIONS

IF THE INTERFACE IS CONNECTED TO THE IIST BUS, THE OPERATOR MAY WISH TO PREPARE THE OTHER INTERFACES FOR THE RUNNING OF THE DIAGNOSTIC ON THIS INTERFACE. SUCH PREPARATION MIGHT INCLUDE INITIALIZATION OF THE OTHER INTERFACES OR ACTIVATION OF THE VARIOUS EXTERNAL INHIBIT CONTROLS (IF AVAILABLE) IN ORDER TO PREVENT INTERFERENCE WITH THE ONLINE INTERFACES.

## 4.0 LOADING PROCEDURE

WHEN A PAPER TAPE IS SUPPLIED, LOAD WITH AN ABS LOADER. WHEN THE PROGRAM IS AVAILABLE ON XXDP MEDIUM, FOLLOW THE LOADING PROCEDURES FOR THE MEDIUM.

## 5.0 CONSOLE SWITCH SETTINGS

- SR15 (1) HALT ON ERROR
- SR14 (1) LOOP ON TEST
- SR13 (1) INHIBIT TYPEOUTS
- SR10 (1) BELL ON ERROR
- SR09 (1) LOOP ON ERROR
- SROO (1) PRINT ALL IIST REGISTERS ON THE CONSOLE TERMINAL

COMPUTERS WITHOUT A HARDWARE SWITCH REGISTER HAVE A SOFTWARE SWITCH REGISTER LOCATION IN MEMORY CALLED "SWREG" (LOCATION 176). THIS LOCATION CAN BE CHANGED MANUALLY OR BY TYPING THE "CNTRL & G" KEYS AND RESPONDING TO THE RESULTING TERMINAL DIALOGUE.

6.0 STARTING PROCEDURE

LOADING ADDRESS 200(8) AND STARTING WILL IDENTIFY THE PROGRAM, INITIALIZE THE SYSTEM, AND BEGIN TESTING USING THE ADDRESS AND CONFIGURATION PARAMETERS ALREADY STORED WITHIN THE PROGRAM (SUCH AS THE DEFAULT PARAMETERS SUPPLIED UPON INITIAL LOAD, OR THE PARAMETERS SUPPLIED VIA THE CONSOLE ON A PREVIOUS START FROM LOCATION 204). ON INITIAL PROGRAM LOAD, THE OPERATING PARAMETERS ARE:

DEVICE ADDRESS = 777500  
INTERRUPT VECTOR = 260  
BR LEVEL = 6  
CONFIGURATION = 0 (MAINTENANCE MODE ONLY;  
NO COMMUNICATION WITH  
I1ST BUS)

LOADING ADDRESS 204(8) AND STARTING WILL INITIATE A CONSOLE DIALOG IN WHICH THE OPERATOR MAY SUPPLY NEW OPERATING PARAMETERS (DEVICE ADDRESS, INTERRUPT VECTOR AND LEVEL, AND TEST SELECTION).

SEVEN TEST SELECTION CODES ARE AVAILABLE FOR SELECTING PROGRAM OPERATION UNDER VARIOUS CONDITIONS. THE CODES ARE USED AS FOLLOWS:

CODE	USE
0	RUNS IN MAINTENANCE MODE ONLY; DOES NOT COMMUNICATE WITH THE I1ST BUS DRIVERS OR RECEIVERS. THIS MODE WOULD BE USED IF THERE IS NO TERMINATOR ON THE I1ST BUS LINES, IF THE TRANSMIT-INHIBIT INPUT IS ASSERTED, IF THE RECEIVER-INHIBIT INPUT FOR THE LINE CORRESPONDING TO THIS INTERFACE'S SELF-ID IS ASSERTED, OR IF CHECKING OF THE I1ST BUS DRIVERS AND RECEIVERS IS NOT DESIRED.
1	THIS CODE IS USED WHEN THERE IS AN I1ST BUS TERMINATOR INSTALLED ON THE MODULE BUT NO I1ST BUS CABLE IS CONNECTED, OR WHEN THE MODULE IS CONNECTED TO THE I1ST BUS AND ALL OTHER INTERFACES ON THE BUS ARE INHIBITED FROM TRANSMITTING OR THIS INTERFACE IS INHIBITED FROM RECEIVING FROM ALL INTERFACES OTHER THAN ITSELF. THIS CODE WOULD BE USED WHEN THE DIP11-C CONTROL PANEL HAS SELECTED THIS INTERFACE TO BE "ONLINE" AND "STANDALONE".
2	THIS CODE IS USED WHEN THE MODULE UNDER TEST IS CONNECTED TO THE I1ST BUS AND ALL OTHER INTERFACES ON THE BUS ARE EITHER IDLE (INITIALIZED) OR INHIBITED.
3	THIS CODE IS USED WHEN THE MODULE UNDER TEST IS CONNECTED TO THE I1ST BUS AND THERE ARE OTHER ACTIVE, NON-INHIBITED INTERFACES ON THE BUS.

CODE ----	USE ---
4	THIS CODE SELECTS THE MAINTENANCE-MODE SELF-BOOT TEST, IN WHICH THE OPERATOR CAN INITIATE A PROGRAMMED OR SANITY-TIMER BOOT OF THIS PROCESSOR VIA A PARTICULAR RECEIVING CHANNEL. THE OPERATOR MUST VERIFY THAT THE PROPER BOOT ACTION TOOK PLACE.
5	THIS CODE SELECTS THE MULTI-PROCESSOR DIALOG-INITIATED BOOT TEST, IN WHICH THE OPERATOR CAN SELECT ANOTHER PROCESSOR TO BE BOOTED. FOLLOWING THE SELECTED BOOT, AN INTERRUPT IS SENT TO THE SELECTED PROCESSOR, SO IF THE BOOT ROUTINE SETS THE INTERRUPT-ENABLE BIT OF THE DIP11-A THE PROGRAM IS REENTERED AND REPORTS THE ACTION.
6	THIS CODE SELECTS THE SANITY TIMEOUT HALT-FREEZE TEST, WHEREBY THE OPERATOR CAN INITIATE A SANITY-TIMER TIMEOUT AND THEN VERIFY THAT THE PROCESSOR WAS PROPERLY HALTED AND/OR THE UNIBUS HUNG. IF AN INDICATOR PANEL, SUCH AS THE DIP11-C, IS CONNECTED, OPERATION OF THE HALT INDICATOR CAN BE VERIFIED.
7	THIS CODE SELECTS THE MANUAL CONTROL PANEL TEST. THE PROGRAM CONTINUALLY CYCLES WHILE MONITORING CHANGES IN THE 'DCF' REGISTER. WHEN A CHANGE OCCURS (SUCH AS A 'BREAK' BIT CHANGING IN RESPONSE TO AN INHIBIT INPUT IN THE I1ST SYSTEM) THE PROGRAM PRINTS OUT THE NEW CONTENTS OF DCF.
10	THIS CODE SELECTS THE "SCOPE-LOOPS" SECTION OF THE PROGRAM. THIS SECTION IS A COLLECTION OF ROUTINES, SELECTABLE BY THE OPERATOR, WHICH REPEATEDLY CYCLE VARIOUS PARTS OF THE LOGIC FOR PURPOSES OF MONITORING WITH AN OSCILLOSCOPE. FOR EXAMPLE, ASSUMING THE 'BOOT' CABLE IS NOT CONNECTED TO THE PROCESSOR, ONE SCOPE LOOP CAN CAUSE A STRING OF BOOT PULSES TO BE GENERATED, WHICH CAN THEN BE MONITORED AT J2 PIN B OR J3 PIN 2. OTHER SCOPE LOOPS CAUSE VARIOUS TRANSMISSIONS ON THE I1ST BUS.

### 7.0 OPERATION

ONCE STARTED WITH CODES 0-3, THE PROGRAM RUNS CONTINUOUSLY AND PERIODICALLY PRINTS END OF PASS MESSAGES. THE HARDWARE SWITCH SELF ID CODE IS REPORTED ON PASS 1, AS IS THE SELECTED SANITY-TIMER COUNT RATE. THE OPERATOR SHOULD VERIFY THAT THESE ARE THE EXPECTED PARAMETERS. SELECTION OF CODES 4-10 CAUSES ENTRY TO OPERATOR-INTERVENTION OPERATIONS.

### 8.0 ERRORS

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE AND RELEVANT TEST DATA IN TABULAR FORM. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED, IF NEEDED, FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF.

### 9.0 LISTING

226

.LIST LOC,BIN,SEQ



(1) 000140 PR3= 140 ;:PRIORITY LEVEL 3  
(1) 000200 PR4= 200 ;:PRIORITY LEVEL 4  
(1) 000240 PR5= 240 ;:PRIORITY LEVEL 5  
(1) 000300 PR6= 300 ;:PRIORITY LEVEL 6  
(1) 000340 PR7= 340 ;:PRIORITY LEVEL 7

:\*'SWITCH REGISTER' SWITCH DEFINITIONS

(1) 100000 SW15= 100000  
(1) 040000 SW14= 40000  
(1) 020000 SW13= 20000  
(1) 010000 SW12= 10000  
(1) 004000 SW11= 4000  
(1) 002000 SW10= 2000  
(1) 001000 SW09= 1000  
(1) 000400 SW08= 400  
(1) 000200 SW07= 200  
(1) 000100 SW06= 100  
(1) 000040 SW05= 40  
(1) 000020 SW04= 20  
(1) 000010 SW03= 10  
(1) 000004 SW02= 4  
(1) 000002 SW01= 2  
(1) 000001 SW00= 1  
(1) .EQUIV SW09,SW9  
(1) .EQUIV SW08,SW8  
(1) .EQUIV SW07,SW7  
(1) .EQUIV SW06,SW6  
(1) .EQUIV SW05,SW5  
(1) .EQUIV SW04,SW4  
(1) .EQUIV SW03,SW3  
(1) .EQUIV SW02,SW2  
(1) .EQUIV SW01,SW1  
(1) .EQUIV SW00,SW0

:\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

(1) 100000 BIT15= 100000  
(1) 040000 BIT14= 40000  
(1) 020000 BIT13= 20000  
(1) 010000 BIT12= 10000  
(1) 004000 BIT11= 4000  
(1) 002000 BIT10= 2000  
(1) 001000 BIT09= 1000  
(1) 000400 BIT08= 400  
(1) 000200 BIT07= 200  
(1) 000100 BIT06= 100  
(1) 000040 BIT05= 40  
(1) 000020 BIT04= 20  
(1) 000010 BIT03= 10  
(1) 000004 BIT02= 4  
(1) 000002 BIT01= 2  
(1) 000001 BIT00= 1  
(1) .EQUIV BIT09,BIT9  
(1) .EQUIV BIT08,BIT8  
(1) .EQUIV BIT07,BIT7  
(1) .EQUIV BIT06,BIT6  
(1) .EQUIV BIT05,BIT5



```
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;:"T" BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;:"TRAP" TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
```

```

243
244      .SBTTL MEMORY MANAGEMENT DEFINITIONS
(1)
(1)      ;*KT11 VECTOR ADDRESS
(1)
(1)      MMVEC= 250
(1)
(1)      ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1)      SR0= 177572
(1)      SR1= 177574
(1)      SR2= 177576
(1)      SR3= 172516
(1)
(1)      ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
(1)
(1)      KIPDR0= 172300
(1)      KIPDR1= 172302
(1)      KIPDR2= 172304
(1)      KIPDR3= 172306
(1)      KIPDR4= 172310
(1)      KIPDR5= 172312
(1)      KIPDR6= 172314
(1)      KIPDR7= 172316
(1)
(1)      ;*KERNEL "I" PAGE ADDRESS REGISTERS
(1)
(1)      KIPAR0= 172340
(1)      KIPAR1= 172342
(1)      KIPAR2= 172344
(1)      KIPAR3= 172346
(1)      KIPAR4= 172350
(1)      KIPAR5= 172352
(1)      KIPAR6= 172354
(1)      KIPAR7= 172356
(1)
245      .SBTTL TRAP CATCHER
(1)
(1)      .=0
(1)      ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
(1)      ;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
(1)      ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
(1)      ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
(1)      ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
(1)      ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
(1)      ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
(1)      ;*
(1)      ;*   PC=YYYYYY UNEXPECTED TRAP TO XXX
(1)      ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
(1)      ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
(1)      ;*
(1)      ;*   YYYYYY=PC AT TIME OF TRAP
(1)      ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
(1)      ;*
(1)      ;*   CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.
(1)
(1)      $40CAT: HALT      ;;HALT
(1)      BR      .-100    ;;BRANCH TO 177700 & TIME OUT (NOT ON
(1)      ;;11/05)

```

```
(1) 000004 002644          .WORD  NOPAR          ;;VECTOR TO STARTING ADDRESS
(1) 000006 000340          .WORD  340           ;;WITH PRIORITY LEVEL 7
(1)          000174          .      =174
(1) 000174 000000  DISPREG: .WORD  0          ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000  SWREG:   .WORD  0          ;;SOFTWARE SWITCH REGISTER
(1)          .SBTTL  STARTING ADDRES(ES)
(1) 000200 000137 002644  JMP      @#NOPAR ;;GO TO START OF PROGRAM
246 000204 000137 002654  JMP      SELPAR    ;;GO TO SELECT PARAMETERS
247
```

```

249
250      .SBTTL  ACT11 HOOKS
(1)
(2)      ;*****
(1)      ;HOOKS REQUIRED BY ACT11
(1)      000210      $SVPC=.          ;SAVE PC
(1)      000046      .=46
(1) 000046 032140      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1)      000052      .=52
(1) 000052 000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)      000210      .=$$VPC          ;; RESTORE PC
251      001000      .=1000
252      .SBTTL  APT PARAMETER BLOCK
(1)
(2)      ;*****
(1)      ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)      ;*****
(1)      001000      .$X=.          ;;SAVE CURRENT LOCATION
(1)      000024      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200      200          ;;FOR APT START UP
(1)      000044      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 001000      $APTHDR        ;;POINT TO APT HEADER BLOCK
(1)      001000      .=.$X          ;;RESET LOCATION COUNTER
(2)      ;*****
(1)      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)      ;INTERFACE SPEC.
(1)
(1) 001000      $APTHD:
(1) 001000 000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 001230      $MBADR: .WORD $MAIL        ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 000017      $STMT: .WORD 15.          ;;RUN TIM OF LONGEST TEST
(1) 001006 000074      $PASTM: .WORD 60.          ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 000000      $UNITM: .WORD 0           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 000052      .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
    
```

254  
 (1)  
 (2)  
 (1)  
 (1)  
 (1)  
 (1)  
 (1) 001100  
 (1) 001100 000000  
 (1) 001102 000  
 (1) 001103 000  
 (1) 001104 000000  
 (1) 001106 000000  
 (1) 001110 000000  
 (1) 001112 000000  
 (1) 001114 000  
 (1) 001115 001  
 (1) 001116 000000  
 (1) 001120 000000  
 (1) 001122 000000  
 (1) 001124 000000  
 (1) 001126 000000  
 (1) 001130 000000  
 (1) 001132 000000  
 (1) 001134 000  
 (1) 001135 000  
 (1) 001136 000000  
 (1) 001140 177570  
 (1) 001142 177570  
 (1) 001144 177560  
 (1) 001146 177562  
 (1) 001150 177564  
 (1) 001152 177566  
 (1) 001154 000  
 (1) 001155 002  
 (1) 001156 012  
 (1) 001157 000  
 (1) 001160 000000  
 (1)  
 (3) 001162 000000  
 (3) 001164 000000  
 (3) 001166 000000  
 (3) 001170 000000  
 (3) 001172 000000  
 (5) 001174 000000  
 (3) 001176 000000  
 (3) 001200 000000  
 (3) 001202 000000  
 (3) 001204 000000  
 (3) 001206 000000  
 (3) 001210 000000  
 (3) 001212 000000  
 (1) 001214 000000  
 (1) 001216 000000  
 (1) 001220 177607  
 (1) 001224 077

000377

.SBTTL COMMON TAGS

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

.=1100  
 SCMTAG: ;:START OF COMMON TAGS  
 .WORD 0 ;:CONTAINS THE TEST NUMBER  
 \$STNM: .BYTE 0 ;:CONTAINS ERROR FLAG  
 \$ERFLG: .BYTE 0 ;:CONTAINS SUBTEST ITERATION COUNT  
 \$ICNT: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS  
 \$LPADR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS  
 \$LPERR: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED  
 \$ERTTL: .WORD 0 ;:CONTAINS ITEM CONTROL BYTE  
 \$ITEMB: .BYTE 0 ;:CONTAINS MAX. ERRORS PER TEST  
 \$ERMAX: .BYTE 1 ;:CONTAINS PC OF LAST ERROR INSTRUCTION  
 \$ERRPC: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA  
 \$GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA  
 \$BDADR: .WORD 0 ;:CONTAINS 'GOOD' DATA  
 \$GDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA  
 \$BDDAT: .WORD 0 ;:RESERVED--NOT TO BE USED  
 .WORD 0 ;:AUTOMATIC MODE INDICATOR  
 \$AUTOB: .BYTE 0 ;:INTERRUPT MODE INDICATOR  
 \$INTAG: .BYTE 0 ;:ADDRESS OF SWITCH REGISTER  
 \$SWR: .WORD DSWR ;:ADDRESS OF DISPLAY REGISTER  
 \$DISPLAY: .WORD DDISP ;:TTY KBD STATUS  
 \$TKS: 177560 ;:TTY KBD BUFFER  
 \$TKB: 177562 ;:TTY PRINTER STATUS REG. ADDRESS  
 \$TPS: 177564 ;:TTY PRINTER BUFFER REG. ADDRESS  
 \$TPB: 177566 ;:CONTAINS NULL CHARACTER FOR FILLS  
 \$NULL: .BYTE 0 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED  
 \$FILLS: .BYTE 2 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'  
 \$FILLC: .BYTE 12 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)  
 \$TPFLG: .BYTE 0 ;:CONTAINS THE ADDRESS FROM  
 \$REGAD: .WORD 0 ;:WHICH (\$REGO) WAS OBTAINED  
 \$REG0: .WORD 0 ;:CONTAINS ((\$REGAD)+0)  
 \$REG1: .WORD 0 ;:CONTAINS ((\$REGAD)+2)  
 \$REG2: .WORD 0 ;:CONTAINS ((\$REGAD)+4)  
 \$REG3: .WORD 0 ;:CONTAINS ((\$REGAD)+6)  
 \$REG4: .WORD 0 ;:CONTAINS ((\$REGAD)+10)  
 \$REG5: .WORD 0 ;:CONTAINS ((\$REGAD)+12)  
 \$TMP0: .WORD 0 ;:USER DEFINED  
 \$TMP1: .WORD 0 ;:USER DEFINED  
 \$TMP2: .WORD 0 ;:USER DEFINED  
 \$TMP3: .WORD 0 ;:USER DEFINED  
 \$TMP4: .WORD 0 ;:USER DEFINED  
 \$TMP5: .WORD 0 ;:USER DEFINED  
 \$TMP6: .WORD 0 ;:USER DEFINED  
 \$TIMES: 0 ;:MAX. NUMBER OF ITERATIONS  
 \$ESCAPE: 0 ;:ESCAPE ON ERROR ADDRESS  
 \$BELL: .ASC:Z <207><377><377> ;:CODE FOR BELL  
 \$QUES: .ASC:Z /? ;:QUESTION MARK

```

(1) 001225 015 SCRLF: .ASCII <15> ::CARRIAGE RETURN
(1) 001226 000012 SLF: .ASCIIZ <12> ::LINE FEED
(2) ::*****
(2) .SBTTL APT MAILBOX-ETABLE
(2)
(3) ::*****
(2) .EVEN
(2) 001230 $MAIL: ::APT MAILBOX
(2) 001230 000000 $MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
(2) 001232 000000 $FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
(2) 001234 000000 $TESTN: .WORD ATESTN ::TEST NUMBER
(2) 001236 000000 $PASS: .WORD APASS ::PASS COUNT
(2) 001240 000000 $DEVCT: .WORD ADEVCT ::DEVICE COUNT
(2) 001242 000000 $UNIT: .WORD AUNIT ::I/O UNIT NUMBER
(2) 001244 000000 $MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
(2) 001246 000000 $MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
(2) 001250 $ETABLE: ::APT ENVIRONMENT TABLE
(2) 001250 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
(2) 001251 000 $ENVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
(2) 001252 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
(2) 001254 000000 $USWR: .WORD AUSWR ::USER SWITCHES
(2) 001256 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
(2) * BITS 15-11=CPU TYPE
(2) * 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2) * 11/70=06,PDQ=07,Q=10
(2) * BIT 10=REAL TIME CLOCK
(2) * BIT 9=FLOATING POINT PROCESSOR
(2) * BIT 8=MEMORY MANAGEMENT
(2) 001260 000 $MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS,M.S. BYTE
(2) 001261 000 $MTYP1: .BYTE AMTYP1 ::MEM. TYPE,BLK#1
(2) * MEM.TYPE BYTE -- (HIGH BYTE)
(2) * 900 NSEC CORE=001
(2) * 300 NSEC BIPOLAR=002
(2) * 500 NSEC MOS=003
(2) 001262 000000 $MADR1: .WORD AMADR1 ::HIGH ADDRESS,BLK#1
(2) * MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
(2) 001264 000 $MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS,M.S. BYTE
(2) 001265 000 $MTYP2: .BYTE AMTYP2 ::MEM.TYPE,BLK#2
(2) 001266 000000 $MADR2: .WORD AMADR2 ::MEM.LAST ADDRESS,BLK#2
(2) 001270 000 $MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS,M.S.BYTE
(2) 001271 000 $MTYP3: .BYTE AMTYP3 ::MEM.TYPE,BLK#3
(2) 001272 000000 $MADR3: .WORD AMADR3 ::MEM.LAST ADDRESS,BLK#3
(2) 001274 000 $MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS,M.S.BYTE
(2) 001275 000 $MTYP4: .BYTE AMTYP4 ::MEM.TYPE,BLK#4
(2) 001276 000000 $MADR4: .WORD AMADR4 ::MEM.LAST ADDRESS,BLK#4
(2) 001300 000000 $VECT1: .WORD AVECT1 ::INTERRUPT VECTOR#1,BUS PRIORITY#1
(2) 001302 000000 $VECT2: .WORD AVECT2 ::INTERRUPT VECTOR#2BUS PRIORITY#2
(2) 001304 000000 $BASE: .WORD ABASE ::BASE ADDRESS OF EQUIPMENT UNDER TEST
(2) 001306 000000 $DEVN: .WORD ADEVN ::DEVICE MAP
(2) 001310 000000 $CDW1: .WORD ACDW1 ::CONTROLLER DESCRIPTION WORD#1
(2) 001312 000000 $CDW2: .WORD ACDW2 ::CONTROLLER DESCRIPTION WORD#2
(2) 001314 000000 $DDW0: .WORD ADDW0 ::DEVICE DESCRIPTOR WORD#0
(2) 001316 000000 $DDW1: .WORD ADDW1 ::DEVICE DESCRIPTOR WORD#1
(2) 001320 000000 $DDW2: .WORD ADDW2 ::DEVICE DESCRIPTOR WORD#2
(2) 001322 000000 $DDW3: .WORD ADDW3 ::DEVICE DESCRIPTOR WORD#3
(2) 001324 000000 $DDW4: .WORD ADDW4 ::DEVICE DESCRIPTOR WORD#4
    
```



```
(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
(1)
(1) 001354 $ERRTB:
255 ;ERROR PARAMETER TABLE
256
257 ;ERROR 1
258 001354 036162 EM1 ;ACR BIT3-0 LOAD-READER ERROR
259 001356 041056 DH1 ;PASS PC EXPCTD ACTUAL ACR
260 001360 042134 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
261 001362 042202 DAF1
262
263 ;ERROR 2
264 001364 036215 EM2 ;ACR BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
265 001366 041122 DH2 ;PASS PC ACR
266 001370 042150 DT2 ;$PASS,$ERRPC,DISPLY
267 001372 042202 DAF1
268
269 ;ERROR 3
270 001374 036655 EM3 ;PGTE BIT11-8 LOAD-READ ERROR
271 001376 041173 DH4 ;PASS PC EXPCTD ACUTAL PGTE
272 001400 042134 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
273 001402 042202 DAF1
274
275 ;ERROR 4
276 001404 036712 EM4 ;PGTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
277 001406 041146 DH3 ;PASS PC PGTE
278 001410 042150 DT2 ;$PASS,$ERRPC,DISPLY
279 001412 042202 DAF1
280
281 ;ERROR 5
282 001414 036773 EM5 ;PGTE BIT3-0 LOAD-READ ERROR
283 001416 041173 DH4 ;PASS PC EXPCTD ACUTAL PGTE
284 001420 042134 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
285 001422 042202 DAF1
286
287 ;ERROR 6
288 001424 037027 EM6 ;PGTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
289 001426 041146 DH3 ;PASS PC PGTE
290 001430 042150 DT2 ;$PASS,$ERRPC,DISPLY
291 001432 042202 DAF1
```



293			:ERROR 7	
294	001434	036274	EM7	:IE (PGTE BIT 6) CANNOT BE SET
295	001436	041146	DH3	:PASS PC PGTE
296	001440	042150	DT2	:\$PASS,\$ERRPC,DISPLY
297	001442	042202	DAF1	
298				
299			:ERROR 10	
300	001444	036331	EM10	:IE (PGTE BIT6) CANNOT BE WRITTEN TO 0
301	001446	041146	DH3	:PASS PC PGTE
302	001450	042134	DT1	:\$PASS,\$ERRPC,DISPLY
303	001452	042202	DAF1	
304				
305			:ERROR 11	
306	001454	036377	EM11	:IE (PGTE BIT6) CANNOT BE CLEARED BY RESET (ACR CLR)
307	001456	041146	DH3	:PASS PC PGTE
308	001460	042150	DT2	:\$PASS,\$ERRPC,DISPLY
309	001462	042202	DAF1	
310				
311			:ERROR 12	
312	001464	036463	EM12	:PTP (PGTE BIT1) CANNOT BE SET
313	001466	041146	DH3	:PASS PC PGTE
314	001470	042150	DT2	:\$PASS,\$ERRPC,DISPLY
315	001472	042202	DAF1	
316				
317			:ERROR 13	
318	001474	036521	EM13	:PIP (PGTE BIT1) CANNOT BE WRITTEN TO 0
319	001476	041146	DH3	:PASS PC PGTE
320	001500	042150	DT2	:\$PASS,\$ERRPC,DISPLY
321	001502	042202	DAF1	
322				
323			:ERROR 14	
324	001504	036570	EM14	:PTP (PGTE BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)
325	001506	041146	DH3	:PASS PC PGTE
326	001510	042150	DT2	:\$PASS,\$ERRPC,DISPLY
327	001512	042202	DAF1	
328				
329			:ERROR 15	
330	001514	037107	EM15	:SITE BIT11-8 LOAD-READ ERROR
331	001516	041240	DH5	:PASS PC EXPCTD ACUTAL SITE
332	001520	042134	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
333	001522	042202	DAF1	
334				
335			:ERROR 16	
336	001524	037144	EM16	:SITE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
337	001526	041305	DH6	:PASS PC SITE
338	001530	042150	DT2	:\$PASS,\$ERRPC,DISPLY
339	001532	042202	DAF1	
340				
341			:ERROR 17	
342	001534	037225	EM17	:SITE BIT3-0 LOAD-READ ERROR
343	001536	041240	DH5	:PASS PC EXPCTD ACUTAL SITE
344	001540	042134	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
345	001542	042202	DAF1	

347			:ERROR	20	
348	001544	037261		EM20	:STTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
349	001546	041305		DH6	:PASS PC STTE
350	001550	042150		DT2	:\$PASS,\$ERRPC,DISPLY
351	001552	042202		DAF1	
352					
353			:ERROR	21	
354	001554	037341		EM21	:LKE (STCS BIT2) CANNOT BE SET
355	001556	041332		DH7	:PASS PC STCS
356	001560	042150		DT2	:\$PASS,\$ERRPC,DISPLY
357	001562	042202		DAF1	
358					
359			:ERROR	22	
360	001564	037377		EM22	:LKE (STCS BIT2) CANNOT BE WRITTEN TO 0
361	001566	041332		DH7	:PASS PC STCS
362	001570	042150		DT2	:\$PASS,\$ERRPC,DISPLY
363	001572	042202		DAF1	
364					
365			:ERROR	23	
366	001574	037446		EM23	:LKE (STCS BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)
367	001576	041332		DH7	:PASS PC STCS
368	001600	042150		DT2	:\$PASS,\$ERRPC,DISPLY
369	001602	042202		DAF1	
370					
371			:ERROR	24	
372	001604	037533		EM24	:STP (STCS BIT1) CANNOT BE SET
373	001606	041332		DH7	:PASS PC STCS
374	001610	042150		DT2	:\$PASS,\$ERRPC,DISPLY
375	001612	042202		DAF1	
376					
377			:ERROR	25	
378	001614	037571		EM25	:STP (STCS BIT1) CANNOT BE WRITTEN TO 0
379	001616	041332		DH7	:PASS PC STCS
380	001620	042150		DT2	:\$PASS,\$ERRPC,DISPLY
381	001622	042202		DAF1	
382					
383			:ERROR	26	
384	001624	037640		EM26	:STP (STCS BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)
385	001626	041332		DH7	:PASS PC STCS
386	001630	042150		DT2	:\$PASS,\$ERRPC,DISPLY
387	001632	042202		DAF1	
388					
389			:ERROR	27	
390	001634	037725		EM27	:ENB (STCS BIT0) CANNOT BE SET
391	001636	041332		DH7	:PASS PC STCS
392	001640	042150		DT2	:\$PASS,\$ERRPC,DISPLY
393	001642	042202		DAF1	
394					
395			:ERROR	30	
396	001644	037763		EM30	:ENB (STCS BIT0) CANNOT BE WRITTEN TO 0
397	001646	041332		DH7	:PASS PC STCS
398	001650	042150		DT2	:\$PASS,\$ERRPC,DISPLY
399	001652	042202		DAF1	

401			:ERROR	31	
402	001654	040032		EM31	:ENB (STCS BIT0) CANNOT BE CLEARED BY RESET (ACR CLR)
403	001656	041332		DH7	:PASS PC STCS
404	001660	042150		DT2	:\$PASS,\$ERRPC,DISPLY
405	001662	042202		DAF1	
406					
407			:ERROR	32	
408	001664	040117		EM32	:STCS BIT15-8 LOAD-READ ERROR
409	001666	041357		DH8	:PASS PC EXPCTD ACUTAL STCS
410	001670	042134		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
411	001672	042202		DAF1	
412					
413			:ERROR	33	
414	001674	040154		EM33	:STCS BIT15-8 WERE NOT CLEARED BY RESET (ACR CLR)
415	001676	041332		DH7	:PASS PC STCS
416	001700	042150		DT2	:\$PASS,\$ERRPC,DISPLY
417	001702	042202		DAF1	
418					
419			:ERROR	34	
420	001704	040235		EM34	:IMSK BIT11-8 LOAD-READ ERROR
421	001706	041424		DH9	:PASS PC EXPCTD ACUTAL IMSK
422	001710	042134		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
423	001712	042202		DAF1	
424					
425			:ERROR	35	
426	001714	040272		EM35	:IMSK BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
427	001716	041471		DH10	:PASS PC IMSK
428	001720	042150		DT2	:\$PASS,\$ERRPC,DISPLY
429	001722	042202		DAF1	
430					
431			:ERROR	36	
432	001724	040353		EM36	:IMSK BIT3-0 LOAD-READ ERROR
433	001726	041424		DH9	:PASS PC EXPCTD ACUTAL IMSK
434	001730	042134		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
435	001732	042202		DAF1	
436					
437			:ERROR	37	
438	001734	040407		EM37	:IMSK BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
439	001736	041471		DH10	:PASS PC IMSK
440	001740	042150		DT2	:\$PASS,\$ERRPC,DISPLY
441	001742	042202		DAF1	
442					
443			:ERROR	40	
444	001744	040467		EM40	:MTCE BIT11-8 LOAD-READ ERROR
445	001746	041516		DH11	:PASS PC EXPCTD ACUTAL MTCE
446	001750	042134		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
447	001752	042202		DAF1	
448					
449			:ERROR	41	
450	001754	040523		EM41	:MTCE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
451	001756	041563		DH12	:PASS PC MTCE
452	001760	042150		DT2	:\$PASS,\$ERRPC,DISPLY
453	001762	042202		DAF1	

```

455          :ERROR 42
456 001764 040603          EM42          :MTCE BIT3-0 LOAD-READ ERROR
457 001766 041516          DH11          :PASS PC EXPCTD ACTUAL MTCE
458 001770 042134          DT1           :$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
459 001772 042202          DAF1
460
461          :ERROR 43
462 001774 040636          EM43          :MTCE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
463 001776 041563          DH12          :PASS PC MTCE
464 002000 042150          DT2           :$PASS,$ERRPC,DISPLY
465 002002 042202          DAF1
466          :ERROR 44
467 002004 040715          EM44          :LIST ERROR - REFER TO THE LISTING AT "PC"
468          :TO DETERMINE THE ERROR.
469 002006 041610          DH13          :PASS PC LINE # EXPCTD ACTUAL DISPLY $TMPO $TMP1
470 002010 042160          DT3           :$PASS,$ERRPC,LINMBR,$GDDAT,$BDDAT,DISPLY,$TMPO,$TMP1
471 002012 042202          DAF1
472
473          :ERROR 45
474 002014 041017          EM45          :ACR (3:0) AUTO-INCREMENT ERROR
475 002016 041056          DH1
476 002020 042134          DT1
477 002022 042202          DAF1
    
```

.SBTTL REGISTER DEFINITIONS

:ACR BIT DEFINITIONS

```

483          100000          CLR=BIT15          :MASTER CLEAR
484          000000          PGTEE=0
485          000001          PGCSE=1
486          000002          STTEE=2
487          000003          STCSE=3
488          000004          IMSKE=4
489          000005          PGFE=5
490          000006          STFE=6
491          000007          DCFE=7
492          000010          EXCE=10
493          000015          MTCE=15
    
```

:PGTE BIT DEFINITION

```

497          004000          PB3=BIT11          :PROGRAM BOOT TRANSMIT ENABLE 3
498          002000          PB2=BIT10          :PROGRAM BOOT TRANSMIT ENABLE 2
499          001000          PB1=BIT9           :PROGRAM BOOT TRANSMIT ENABLE 1
500          000400          PB0=BIT8           :PROGRAM BOOT TRANSMIT ENABLE 0
501          000010          PI3=BIT3          :PROGRAM INTERRUPT TRANSMIT ENABLE 3
502          000004          PI2=BIT2          :PROGRAM INTERRUPT TRANSMIT ENABLE 2
503          000002          PI1=BIT1          :PROGRAM INTERRUPT TRANSMIT ENABLE 1
504          000001          PI0=BIT0          :PROGRAM INTERRUPT TRANSMIT ENABLE 0
    
```

:PGCS BIT DEFINITIONS

```

508          100000          ERR=BIT15          :ERROR
509          040000          GRJ=BIT14          :GO REJECT
510          020000          PGMR=BIT13         :PGTE MODIFICATION REFUSED
    
```

```

511      010000      STMR=BIT12      ;STTE MODIFICATION REFUSED
512      004000      PRDY=BIT11      ;PG READY
513      000000      SID0=0      ;SELF ID 0
514      000400      SID1=BIT8      ;SELF ID 1
515      001000      SID2=BIT9      ;SELF ID 2
516      001400      SID3=BIT9+BIT8 ;SELF ID 3
517      000010      IP=BIT3      ;INTERRUPT PENDING IP=ERR+FLAGS
518      000004      IE=BIT2      ;IIST INTERRUPT ENABLE
519      000002      PTP=BIT1      ;PGTE PARITY BIT
520      000001      GO=BIT0      ;GENERATE BOOT/INTERRUPT
521
522      ;STTE BIT DEFINITIONS
523
524      004000      SB3=BIT11      ;SANITY TIMER BOOT TRANSMIT ENABLE 3
525      002000      SB2=BIT10      ;SANITY TIMER BOOT TRANSMIT ENABLE 2
526      001000      SB1=BIT9      ;SANITY TIMER BOOT TRANSMIT ENABLE 1
527      000400      SB0=BIT8      ;SANITY TIMER BOOT TRANSMIT ENABLE 0
528      000010      SI3=BIT3      ;SANITY TIMER INTERRUPT TRANSMIT ENABLE 3
529      000004      SI2=BIT2      ;SANITY TIMER INTERRUPT TRANSMIT ENABLE 2
530      000002      SI1=BIT1      ;SANITY TIMER INTERRUPT TRANSMIT ENABLE 1
531      000001      SI0=BIT0      ;SANITY TIMER INTERRUPT TRANSMIT ENABLE 0
532
533      ;STCS BIT DEFINITIONS
534
535      000010      TMO=BIT3      ;TIMER EXPIRED
536      000004      LKE=BIT2      ;LOCKUP ENABLE
537      000002      STP=BIT1      ;STTE PARITY BIT
538      000001      ENB=BIT0      ;SANITY TIMER ENABLE
539
540      ;IMSK BIT DEFINITIONS
541
542      004000      BM3=BIT11      ;BOOT INHIBIT 3
543      002000      BM2=BIT10      ;BOOT INHIBIT 2
544      001000      BM1=BIT9      ;BOOT INHIBIT 1
545      000400      BM0=BIT8      ;BOOT INHIBIT 0
546      000010      IM3=BIT3      ;INTERRUPT INHIBIT 3
547      000004      IM2=BIT2      ;INTERRUPT INHIBIT 2
548      000002      IM1=BIT1      ;INTERRUPT INHIBIT 1
549      000001      IM0=BIT0      ;INTERRUPT INHIBIT 0
550
551      ;PGF BIT DEFINITIONS
552
553      004000      PBF3=BIT11      ;PROGRAM GENERATED BOOT FLAG 3
554      002000      PBF2=BIT10      ;PROGRAM GENERATED BOOT FLAG 2
555      001000      PBF1=BIT9      ;PROGRAM GENERATED BOOT FLAG 1
556      000400      PBF0=BIT8      ;PROGRAM GENERATED BOOT FLAG 0
557      000010      PIF3=BIT3      ;PROGRAM GENERATED INTERRUPT FLAG 3
558      000004      PIF2=BIT2      ;PROGRAM GENERATED INTERRUPT FLAG 2
559      000002      PIF1=BIT1      ;PROGRAM GENERATED INTERRUPT FLAG 1
560      000001      PIF0=BIT0      ;PROGRAM GENERATED INTERRUPT FLAG 0
561
562      ;STF BIT DEFINITIONS
563
564      000010      SBF3=BIT3      ;SANITY TIMER GENERATED BOOT FLAG 3
565      000004      SBF2=BIT2      ;SANITY TIMER GENERATED BOOT FLAG 2
566      000002      SBF1=BIT1      ;SANITY TIMER GENERATED BOOT FLAG 1
    
```

567 000001 SBFO=BIT0 ;SANITY TIMER GENERATED BOOT FLAG 0  
568 000010 SIF3=BIT3 ;SANITY TIMER GENERATED INTERRUPT FLAG 3  
569 000004 SIF2=BIT2 ;SANITY TIMER GENERATED INTERRUPT FLAG 2  
570 000002 SIF1=BIT1 ;SANITY TIMER GENERATED INTERRUPT FLAG 1  
571 000001 SIFO=BIT0 ;SANITY TIMER GENERATED INTERRUPT FLAG 0

;DCF BIT DEFINITIONS

575 004000 BRK3=BIT11 ;DISCONNECT OR POWER LOSS STATE OF IIST 3  
576 002000 BRK2=BIT10 ;DISCONNECT OR POWER LOSS STATE OF IIST 2  
577 001000 BRK1=BIT9 ;DISCONNECT OR POWER LOSS STATE OF IIST 1  
578 000400 BRK0=BIT8 ;DISCONNECT OR POWER LOSS STATE OF IIST 0  
579 000010 DCF3=BIT3 ;DCLO/DISCONNECT INTERRUPT FLAG 3  
580 000004 DCF2=BIT2 ;DCLO/DISCONNECT INTERRUPT FLAG 2  
581 000002 DCF1=BIT1 ;DCLO/DISCONNECT INTERRUPT FLAG 1  
582 000001 DCF0=BIT0 ;DCLO/DISCONNECT INTERRUPT FLAG 0

;EXC BIT DEFINITIONS

586 004000 UI3=BIT11 ;UNEXPECTED VALID INPUT FLAG 3  
587 002000 UI2=BIT10 ;UNEXPECTED VALID INPUT FLAG 2  
588 001000 UI1=BIT9 ;UNEXPECTED VALID INPUT FLAG 1  
589 000400 UI0=BIT8 ;UNEXPECTED VALID INPUT FLAG 0  
590 000010 RTE3=BIT3 ;UNEXPECTED INVALID INPUT FLAG 3  
591 000004 RTE2=BIT2 ;UNEXPECTED INVALID INPUT FLAG 2  
592 000002 RTE1=BIT1 ;UNEXPECTED INVALID INPUT FLAG 1  
593 000001 RTE0=BIT0 ;UNEXPECTED INVALID INPUT FLAG 0

;MTC BIT DEFINITIONS

597 004000 MTYP=BIT11 ;MNT TYPE BIT.0=PG; 1=ST  
598 002000 MFRM=BIT10 ;MNT FRAME BIT.0=NORMAL; 1=FRAMING ERROR  
599 000000 MID0=0 ;MNT ID 0  
600 000400 MID1=BIT8 ;MNT ID 1  
601 001000 MID2=BIT9 ;MNT ID 2  
602 001400 MID3=BIT9+BIT8 ;MNT ID 3  
603 000010 DSBT=BIT3 ;DISABLE BOOT  
604 000004 MENB=BIT2 ;ENABLE B11-8 OF MNT  
605 000002 MLEN=BIT1 ;ENABLE MAINT. LOOP (IDLE XMIT DRIVERS)  
606 000001 MDSD=BIT0 ;DISABLE XMIT DRIVER (CAUSE A BREAK)  
607 000016 TMT=DSBT+MENB+MLEN

;OTHER DEFINITIONS

611 005726 POP=5726  
612 022626 POPPOP=22626  
613 000200 CRLF=200

615  
616  
662

.SBTTL TEST MACRO DEFINITIONS

```
703 002024 START:
(1) .SBTTL INITIALIZE THE COMMON TAGS
(1) ;;CLEAR THE COMMON TAGS (%CMTAG) AREA
(1) 002024 012706 001100 MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
(1) 002030 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
(1) 002032 022706 001140 CMP #SWR,R6 ;;DONE?
(1) 002036 001374 BNE -6 ;;LOOP BACK IF NO
(1) 002040 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
(1) ;;INITIALIZE A FEW VECTORS
(1) 002044 012737 032174 000020 MOV #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
(1) 002052 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7
(1) 002060 012737 032446 000030 MOV #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
(1) 002066 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
(1) 002074 012737 035454 000034 MOV #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
(1) 002102 012737 000340 000036 MOV #340,@#TRAPVEC+2;LEVEL 7
(1) 002110 012737 035536 000024 MOV #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
(1) 002116 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;LEVEL 7
(1) 002124 013737 032106 032100 MOV SENDCT,%EOPCT ;;SETUP END-OF-PROGRAM COUNTER
(1) 002132 005037 001214 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
(1) 002136 005037 001216 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
(1) 002142 112737 000001 001115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
(1) 002150 012737 002150 001106 MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
(1) 002156 012737 002156 001110 MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
(2) ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
(2) ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
(2) 002164 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
(2) 002170 012737 002224 000004 MOV #64,$@#ERRVEC ;;SET UP ERROR VECTOR
(2) 002176 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
(2) 002204 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
(2) 002212 022777 177777 176720 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
(2) 002220 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
(2) ;;AND THE HARDWARE SWR IS NOT -1
(2) 002222 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
(2) 002224 012716 002232 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
(2) 002230 000002 RTI
(2) 002232 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
(2) 002240 012737 000174 001142 MOV #DISPREG,DISPLAY
(2) 002246 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 002252 005037 001236 CLR $PASS ;;CLEAR PASS COUNT
(2) 002256 132737 000200 001251 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 002264 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
(2) 002266 012737 001252 001140 MOV #SSWREG,SWR ;;NG,USE APT SWITCH REGISTER
(2) 002274
704 .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 002274 005227 177777 INC #-1 ;;FIRST TIME?
(1) 002300 001070 BNE 68$ ;;BRANCH IF NO
(1) 002302 022737 032140 000042 CMP #SENDAD,@#42 ;;ACT-11?
(1) 002310 001464 BEQ 68$ ;;BRANCH IF YES
(1) 002312 104401 002360 TYPE ,69$ ;;TYPE ASCIZ STRING
(2) .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 002316 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 002322 001012 BNE 70$ ;;BRANCH IF YES
(2) 002324 123727 001250 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
(2) 002332 001406 BEQ 70$ ;;BRANCH IF YES
```



```

(2) 002334 023727 001140 000176      CMP      SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
(2) 002342 001005                      BNE      71$            ;;BRANCH IF NO
(2) 002344 104406                      GTSWR                      ;;GET SOFT-SWR SETTINGS
(2) 002346 000403                      BR       71$
(2) 002350 112737 000001 001134 70$:  MOVB     #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
(2) 002356 000441                      71$:
(1) 002356 000441                      BR       68$            ;;GET OVER THE ASCIZ
(1)                                     ;;69$: .ASCIZ <CRLF>#DIP11-A (IIST) SINGLE-INTERFACE DIAGNOSTIC MAINDEC-11-CR11A-B#<<
(1) 002462                                     68$:
705 002462 012737 000006 000004      MOV      #6,4          ;SET TO TRAP
706 002470 012737 000004 000006      MOV      #4,6          ;ON AN NXM
707                                     ; JSR     PC,$SIZE      ;SIZE MEMORY
708 002476 005737 025016                      TST     PARFLG ;SEE IF PARAMETERS TO BE INPUT
709 002502 001403                      BEQ     1$             ;SKIP IF NOT
710 002504 004737 023644                      JSR     PC,GETPAR      ;GET PARAMETERS
711 002510 000446                      BR      2$
712 002512 005227 177777 1$:      INC     #-1
713 002516 001043                      BNE     2$            ;FIRST TIME?
714 002520 022737 032140 000042      CMP     #SENDAD,@#42  ;ACT-11#
715 002526 001437                      BEQ     2$            ;DON'T PRINT IF YES.
716 002530 104401 002536                      TYPE    ,73$          ;;TYPE ASCIZ STRING
(1) 002534 000434                      BR      72$          ;;GET OVER THE ASCIZ
(1)                                     ;;73$: .ASCIZ <CRLF>/START AT 204 TO SELECT NEW PARAMETERS OR RUNNING MODE/<CRLF>
(1) 002626                                     72$:
717 002626 012737 000340 177776 2$:  MOV     #340,PSW      ;RISE PRIORITY
718 002634 005037 025022                      CLR     STCNT         ;INITIALIZE S.I. COUNT VALUE.
719 002640 000137 002666                      JMP     BEGIN         ;GO START TESTING
720
721
722                                     ;COME HERE ON START AT 200
723 002644 005037 025016      NOPAR: CLR     PARFLG      ;CLEAR PARAMETER FLAG
724 002650 000137 002024                      JMP     START         ;GO TO STARTUP
725
726                                     ;COME HERE ON START AT 204
727 002654 012737 000001 025016      SELPAR: MOV    #1,PARFLG  ;SET PARAMETER FLAG
728 002662 000137 002024                      JMP     START
729
730 002666 005737 000042      BEGIN: TST     @#42      ;ARE WE IN ACT?
731 002672 001026                      BNE     3$            ;IF YES, DON'T PRINT.
732 002674 104401 002702                      TYPE    ,65$          ;;TYPE ASCIZ STRING
(1) 002700 000411                      BR      64$          ;;GET OVER THE ASCIZ
(1)                                     ;;65$: .ASCIZ <CRLF>/RUNNING IN MODE /
(1) 002724                                     64$:
733 002724 013700 025020                      MOV     CONFIG,R0     ;GET SELECTED MODE CODE.
734 002730 006300                      ASL     R0             ;MAKE WORD INDEX.
735 002732 016037 002762 002742      MOV     MTAB(R0),2$   ;GET ADDRESS OF MESSAGE TO PRINT.
736 002740 104401                      TYPE
737 002742 000000 2$:      .WORD   0
738 002744 104401 001225                      TYPE    ,$CRLF        ;NEWLINE.
739 002750 013700 025020 3$:      MOV     CONFIG,R0
740 002754 006300                      ASL     R0
741 002756 000170 003540                      JMP     @DTAB(R0)     ;GO TO SELECTED TEST.
742 002762 1$:
743                                     ;TABLE OF POINTERS TO MESSAGES.
744 002762 003006 003040 003130      MTAB:  .WORD   TTMG0,TTMG1,TTMG2,TTMG3
002770 003216

```

745	002772	003303	003342	003414	.WORD	TTMG4,TTMG5,TTMG6,TTMG7
	003000	003460				
746	003002	003516			.WORD	TTMG10
747						
748	003004	000010			CNFMAX: .-MTAB-2/2	;MAXIMUM SELECTION CODE ALLOWED.
749						
750					;MESSAGES DESCRIBING THE CODES:	
751						
752	003006	020060	046450	044501	TTMG0: .ASCIZ	/0 (MAINTENANCE-MODE ONLY)/
	003014	052116	047105	047101		
	003022	042503	046455	042117		
	003030	020105	047117	054514		
	003036	000051				
753	003040	020061	051450	040524	TTMG1: .ASCIZ	/1 (STANDALONE-ONLINE TO BUS; ISOLATE FROM OTHER IIST'S)/
	003046	042116	046101	047117		
	003054	026505	047117	044514		
	003062	042516	052040	020117		
	003070	052502	035523	044440		
	003076	047523	040514	042524		
	003104	043040	047522	020115		
	003112	052117	042510	020122		
	003120	044511	052123	051447		
	003126	000051				
754	003130	020062	047450	046116	TTMG2: .ASCIZ	/2 (ONLINE TO IIST BUS; OTHER IIST'S ISOLATED OR IDLE)/
	003136	047111	020105	047524		
	003144	044440	051511	020124		
	003152	052502	035523	047440		
	003160	044124	051105	044440		
	003166	051511	023524	020123		
	003174	051511	046117	052101		
	003202	042105	047440	020122		
	003210	042111	042514	000051		
755	003216	020063	047450	046116	TTMG3: .ASCIZ	/3 (ONLINE TO IIST BUS; OTHER IIST'S MIGHT BE ACTIVE)/
	003224	047111	020105	047524		
	003232	044440	051511	020124		
	003240	052502	035523	047440		
	003246	044124	051105	044440		
	003254	051511	023524	020123		
	003262	044515	044107	020124		
	003270	042502	040440	052103		
	003276	053111	024505	000		
756	003303	064	024040	040515	TTMG4: .ASCIZ	/4 (MAINT.-MODE SELF-BOOT TEST)/
	003310	047111	027124	046455		
	003316	042117	020105	042523		
	003324	043114	041055	047517		
	003332	020124	042524	052123		
	003340	000051				
757	003342	020065	046450	046125	TTMG5: .ASCIZ	/5 (MULTI-PROC DIALOG-INITIATED BOOT TEST)/
	003350	044524	050055	047522		
	003356	020103	044504	046101		
	003364	043517	044455	044516		
	003372	044524	052101	042105		
	003400	041040	047517	020124		
	003406	042524	052123	000051		
758	003414	020066	051450	047101	TTMG6: .ASCIZ	/6 (SANITY-TIMEOUT HALT-FREEZE TEST)/
	003422	052111	026531	044524		

	003430	042515	052517	020124	
	003436	040510	052114	043055	
	003444	042522	055105	020105	
	003452	042524	052123	000051	
759	003460	020067	046450	047101	TTMG7: .ASCIZ /7 (MANUAL CONTROL PANEL TEST)/
	003466	040525	020114	047503	
	003474	052116	047522	020114	
	003502	040520	042516	020114	
	003510	042524	052123	000051	
760	003516	030061	024040	041523	TTMG10: .ASCIZ /10 (SCOPE LOOPS)/
	003524	050117	020105	047514	
	003532	050117	024523	000	
761		003540			.EVEN
762					
763					;DISPATCH TABLE:
764	003540	003562	003562	003562	DTAB: TST1, TST1, TST1, TST1
	003546	003562			
765	003550	025030	026160	027500	MMBOOT, MPDIBT, STHLT, MCP
	003556	027750			
766	003560	030160			SLOOPS
767					

```

769 .SBTTL
770 .SBTTL
771 .SBTTL BASIC REGISTER TESTS
772 .SBTTL
773
774
(3) :*****
(3) :*TEST 1 ACR (BIT3-0) WRITE-READ TEST USING A COUNT PATTERN
(3) :*****
(2) 003562 000004 TST1: SCOPE
775 003564 004737 023454 JSR PC,CLEAR ;CLEAR THE WORLD
776 003570 012737 003604 0C1110 MOV #1$,SLPERR ;SET SCOPE LOOP POINTER
777 003576 012737 000017 001124 MOV #17,$GDDAT ;INIT EXPECTED TC 17
778 003604 013777 001124 021120 1$: MOV $GDDAT,@ACR ;LOAD ACR WITH $GDDAT
779 003612 017737 021114 024750 MOV @ACR,DISPLY ;READ ACR
780 003620 013737 024750 001126 MOV DISPLY,$BDDAT ;COPY ACR INTO $BDDAT
781 003626 042737 177760 001126 BIC #-<17+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST
782 003634 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ
783 003642 001403 BEQ 2$ ;SKIP IF OK
784 003644 104001 ERROR+ 1 ;ACR BIT3-0 LOAD-READ ERROR
785 003646 004737 023526 JSR PC,DMPREG
786 003652 005237 001124 2$: INC $GDDAT ;BUMP EXPECTED
787 003656 042737 177760 001124 BIC #-<17+1>,$GDDAT ;MASK TO BITS UNDER TEST
788 003664 023727 001124 000017 CMP $GDDAT,#17 ;DONE ALL?
789 003672 001344 BNE 1$ ;TEST AGAIN IF NO
790 003674 012737 003702 001110 MOV #3$,SLPERR ;CHANGE LOOP CONTROL
791 003702 012777 000017 021022 3$: MOV #17,@ACR ;SET ALL BITS
792 003710 012777 100000 021014 MOV #CLR,@ACR ;CLEAR THE WORLD
793 003716 017737 021010 024750 MOV @ACR,DISPLY ;READ ACR
794 003724 032737 0C1017 024750 BIT #17,DISPLY ;CHECK BIT3-0 IN ACR FOR 0
795 003732 001403 BEQ TST2 ;SKIP IF RESET CLEARED BIT3-0 IN ACR
796 003734 104002 ERROR+ 2 ;BIT3-0 IN ACR WERE NOT CLEARED BY RESET
797 003736 004737 023526 JSR PC,DMPREG
    
```

```

802          :*****
(3)          :*TEST 2      CHECK ACR AUTO-INCREMENT LOGIC
(3)          :*****
(2) 003742 000004          TST2: SCOPE
803 003744 005037 024750  CLR      DISPLY      ;START INITIAL ACR LOAD VALUE AT 0
804 003750 012737 000001 001124  MOV      #1,$GDDAT  ;INIT EXPECTED VALUE TO 1.
805 003756 012737 003764 001110  MOV      #1,$SLPERR ;SET SCOPE LOOP POINTER
806 003764 013777 024750 020740 1$: MOV      @ACR      ;LOAD ACR WITH INITIAL VALUE.
807 003772 005777 020740  TST      @ADR      ;DO A DATI TO ADR
808 003776 012700 000005  MOV      #5,R0     ;SET LOOP COUNTER FOR REPEAT READS.
809 004002 017737 020724 001126 2$: MOV      @ACR,$BDDAT ;GET ACR CONTENTS
810 004010 123737 001126 001124  CMPB    $BDDAT,$GDDAT ;CHECK LOWER BYTE (ACCESS CODE)
811 004016 001403  BEQ      3$      ;FOR INCREMENT OR NOT.
812 004020 104045  ERROR+  45      ;ACR AUTO-INCR ERROR
813 004022 004737 023526  JSR      PC,DMPREG
814 004026 005300 3$: DEC      R0      ;LOOP TO READ ACR SEVERAL TIMES
815 004030 001364  BNE      2$      ; TO MAKE SURE IT DOESN'T CHANGE
816 004032 013777 024750 020672  MOV      DISPLY,@ACR ;LOAD ACR WITH INITIAL VALUE.
817 004040 012777 000000 020670  MOV      #0,@ADR   ;DO DATO TO ADR
818 004046 017737 020660 001126  MOV      @ACR,$BDDAT ;GET ACR CONTENTS
819 004054 123737 001126 001124  CMPB    $BDDAT,$GDDAT ;CHECK LOWER BYTE (ACCESS CODE)
820 004062 001403  BEQ      4$      ;FOR INCREMENT OR NOT.
821 004064 104045  ERROR+  45      ;ACR AUTO-INCR ERROR
822 004066 004737 023526  JSR      PC,DMPREG
823 004072 013777 024750 020632 4$: MOV      DISPLY,@ACR ;LOAD ACR WITH INITIAL VALUE.
824 004100 042777 177777 020630  BIC      #-1,@ADR  ;DO DATIP-DATO INTO ADR
825 004106 017737 020620 001126  MOV      @ACR,$BDDAT ;GET ACR CONTENTS
826 004114 123737 001126 001124  CMPB    $BDDAT,$GDDAT ;CHECK LOWER BYTE (ACCESS CODE)
827 004122 001403  BEQ      5$      ;FOR INCREMENT OR NOT.
828 004124 104045  ERROR+  45      ;ACR AUTO-INCR ERROR
829 004126 004737 023526  JSR      PC,DMPREG
830 004132 013777 024750 020572 5$: MOV      DISPLY,@ACR ;LOAD ACR WITH INITIAL VALUE.
831 004140 112777 000000 020570  MOVB    #0,@ADR   ;DO DATOB TO LOW BYTE OF ADR
832 004146 017737 020560 001126  MOV      @ACR,$BDDAT ;GET ACR CONTENTS
833 004154 123737 001126 001124  CMPB    $BDDAT,$GDDAT ;CHECK LOWER BYTE (ACCESS CODE)
834 004162 001403  BEQ      6$      ;FOR INCREMENT OR NOT.
835 004164 104045  ERROR+  45      ;ACR AUTO-INCR ERROR
836 004166 004737 023526  JSR      PC,DMPREG
837 004172 013777 024750 020532 6$: MOV      DISPLY,@ACR ;LOAD ACR WITH INITIAL VALUE.
838 004200 112777 000000 020532  MOVB    #0,@ADRH  ;DO DATOB TO HIGH BYTE OF ADR
839 004206 017737 020520 001126  MOV      @ACR,$BDDAT ;GET ACR CONTENTS
840 004214 123737 001126 001124  CMPB    $BDDAT,$GDDAT ;CHECK LOWER BYTE (ACCESS CODE)
841 004222 001403  BEQ      7$      ;FOR INCREMENT OR NOT.
842 004224 104045  ERROR+  45      ;ACR AUTO-INCR ERROR
843 004226 004737 023526  JSR      PC,DMPREG
844 004232 005237 024750 7$: INC      DISPLY   ;BUMP INITIAL VALUE
845 004236 005237 001124  INC      $GDDAT   ;BUMP EXPECTED VALUE
846 004242 023727 024750 000011  CMP     DISPLY,#11 ;LOOP THRU CODE 0-10 FIRST (AUTO-INCREMENT)
847 004250 002645  BLT     1$      ;
848 004252 003005  BGT     8$      ;CODES 11-17 DO NOT INCREMENT.
849 004254 012737 000011 001124  MOV     #11,$GDDAT ;SO SET INITIAL & EXPECTED EQUAL.
850 004262 000137 003764  JMP     1$      ;
851 004266 023727 024750 000020 8$: CMP     DISPLY,#20 ;ARE WE DONE?
852 004274 002633  BLT     1$      ;LOOP IF NOT.

```

```
854
(4)
(4)
(3) 004276 000004
(1) 004300 004737 023454
(1) 004304 012737 004320 001110
(1) 004312 012737 007400 001124
(1) 004320 012777 000000 020404
(1) 004326 013777 001124 020402
(1) 004334 012777 000000 020370
(1) 004342 017737 020370 024750
(1) 004350 013737 024750 001126
(1) 004356 042737 170377 001126
(1) 004364 023737 001124 001126
(1) 004372 001403
(1) 004374 104003
(1) 004376 004737 023526
(1) 004402 062737 000400 001124
(1) 004410 042737 170377 001124
(1) 004416 023727 001124 007400
(1) 004424 001335
(1) 004426 012737 004434 001110
(1) 004434 012777 000000 020270
(1) 004442 012777 007400 020266
(1) 004450 004737 023454
(1) 004454 012777 000000 020250
(1) 004462 017737 020250 024750
(1) 004470 032737 007400 024750
(1) 004476 001403
(1) 004500 104004
(1) 004502 004737 023526
(
004506
```

```
*****
: *TEST 3 PGTE (BIT11-8) WRITE-READ TEST USING PGTE COUNT PATTERN
: *****
TST3: SCOPE
      JSR PC,CLEAR ;CLEAR THE WORLD
      MOV #1$, $LPERR ;SET SCOPE LOOP POINTER
      MOV #7400, $GDDAT ;INIT EXPECTED TO 7400
1$:   MOV #PGTEE, @ACR ;SET ADR TO PGTE
      MOV $GDDAT, @PGTE ;LOAD PGTE WITH $GDDAT
      MOV #PGTEE, @ACR ;RESET ADR TO PGTE
      MOV @PGTE, DISPLY ;READ PGTE
      MOV DISPLY, $BDDAT ;COPY PGTE INTO $BDDAT
      BIC #-<7400+1>, $BDDAT ;CLEAR ALL BUT BITS UNDER TEST
      CMP $GDDAT, $BDDAT ;CHECK BITS LOADED AGAINST BITS READ
      BEQ 2$ ;SKIP IF OK
      ERROR+ N ;PGTE BIT11-8 LOAD-READ ERROR
      JSR PC, DMPREG
2$:   ADD #BIT8, $GDDAT ;BUMP EXPECTED
      BIC #-<7400+1>, $GDDAT ;MASK TO BITS UNDER TEST
      CMP $GDDAT, #7400 ;DONE ALL?
      BNE 1$ ;TEST AGAIN IF NO
      MOV #3$, $LPERR ;CHANGE LOGP CONTROL
3$:   MOV #PGTEE, @ACR ;RESET ADR TO PGTE
      MOV #7400, @PGTE ;SET ALL BITS
      JSR PC, CLEAR ;CLEAR THE WORLD
      MOV #PGTEE, @ACR ;SET ADDR TO PGTE
      MOV @PGTE, DISPLY ;READ PGTE
      BIT #7400, DISPLY ;CHECK BIT11-8 IN PGTE FOR 0
      BEQ 4$ ;SKIP IF RESET CLEARED BIT11-8 IN PGTE
      ERROR+ N ;BIT11-8 IN PGTE WERE NOT CLEARED BY RESET
      JSR PC, DMPREG
4$:
```

```

856
(4)
(4)
(3) 004506 000004
(1) 004510 004737 023454
(1) 004514 012737 004530 001110
(1) 004522 012737 000017 001124
(1) 004530 012777 000000 020174
(1) 004536 013777 001124 020172
(1) 004544 012777 000000 020160
(1) 004552 017737 020160 024750
(1) 004560 013737 024750 001126
(1) 004566 042737 177760 001126
(1) 004574 023737 001124 001126
(1) 004602 001403
(1) 004604 104005
(1) 004606 004737 023526
(1) 004612 062737 000001 001124
(1) 004620 042737 177760 001124
(1) 004626 023727 001124 000017
(1) 004634 001335
(1) 004636 012737 004644 001110
(1) 004644 012777 000000 020060
(1) 004652 012777 000017 020056
(1) 004660 004737 023454
(1) 004664 012777 000000 020040
(1) 004672 017737 020040 024750
(1) 004700 032737 000017 024750
(1) 004706 001403
(1) 004710 104006
(1) 004712 004737 023526
(1) 004716

```

```

*****
*TEST 4 PGTE (BIT3-0) WRITE-READ TEST USING PGTE COUNT PATTERN
*****
TST4: SCOPE
      JSR PC,CLEAR ;CLEAR THE WORLD
      MOV #1$,SLPERR ;SET SCOPE LOOP POINTER
      MOV #17,$GDDAT ;INIT EXPECTED TO 17
1$:   MOV #PGTEE,@ACR ;SET ADR TO PGTE
      MOV $GDDAT,@PGTE ;LOAD PGTE WITH $GDDAT
      MOV #PGTEE,@ACR ;RESET ADR TO PGTE
      MOV @PGTE,DISPLY ;READ PGTE
      MOV DISPLY,$BDDAT ;COPY PGTE INTO $BDDAT
      BIC #-<17+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST
      CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ
      BEQ 2$ ;SKIP IF OK
      ERROR+ N ;PGTE BIT3-0 LOAD-READ ERROR
      JSR PC,DMPREG
2$:   ADD #BIT0,$GDDAT ;BUMP EXPECTED
      BIC #-<17+1>,$GDDAT ;MASK TO BITS UNDER TEST
      CMP $GDDAT,#17 ;DONE ALL?
      BNE 1$ ;TEST AGAIN IF NO
3$:   MOV #3$,SLPERR ;CHANGE LOOP CONTROL
      MOV #PGTEE,@ACR ;RESET ADR TO PGTE
      MOV #17,@PGTE ;SET ALL BITS
      JSR PC,CLEAR ;CLEAR THE WORLD
      MOV #PGTEE,@ACR ;SET ADDR TO PGTE
      MOV @PGTE,DISPLY ;READ PGTE
      BIT #17,DISPLY ;CHECK BIT3-0 IN PGTE FOR 0
      BEQ 4$ ;SKIP IF RESET CLEARED BIT3-0 IN PGTE
      ERROR+ N ;BIT3-0 IN PGTE WERE NOT CLEARED BY RESET
      JSR PC,DMPREG
4$:

```

858

(4)  
(4)  
(3) 004716 000004  
(1) 004720 004737 023454  
(1) 004724 012777 000001 020000  
(1) 004732 012777 000004 017776  
(1) 004740 012777 000001 017764  
(1) 004746 017737 017764 024750  
(1) 004754 032737 000004 024750  
(3) 004762 001003  
(1) 004764 104007  
(1) 004766 004737 -023526  
(1)  
(5)  
(4)  
(4)

\*\*\*\*\*  
: \*TEST 5 TEST THAT IE (PGCS BIT2) CAN BE SET  
: \*\*\*\*\*

TST5: SCOPE  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #PGCSE,@ACR ;SET ADR TO PGCS  
MOV #IE,@PGCS ;SET IE IN PGCS  
MOV #PGCSE,@ACR ;RESET ADR TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #IE,DISPLY ;CHECK FOR IE=1  
BNE TST6 ;SKIP IF IE IS SET IN PGCS  
ERROR+ N ;IE (PGCS BIT2) CANNOT BE SET  
JSR PC,DMPREG

(1)  
(5)  
(4)  
(4)  
(3) 004772 000004  
(1) 004774 012777 000001 017730  
(1) 005002 012777 000004 017726  
(1) 005010 012777 000001 017714  
(1) 005016 042777 000004 017712  
(1) 005024 012777 000001 017700  
(1) 005032 017737 017700 024750  
(1) 005040 032737 000004 024750  
(3) 005046 001403  
(1) 005050 104010  
(1) 005052 004737 023526  
(1)  
(1)

\*\*\*\*\*  
: \*TEST 6 TEST THAT IE (PGCS BIT2) CAN BE CLEARED  
: \*\*\*\*\*

TST6: SCOPE  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV #IE,@PGCS ;SET IE IN PGCS  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
BIC #IE,@PGCS ;CLEAR IE IN PGCS  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #IE,DISPLY ;CHECK FOR IE=0  
BEQ TST7 ;SKIP IF IE IS CLEAR IN PGCS  
ERROR+ N ;IE (PGCS BIT2) CANNOT BE CLEARED  
JSR PC,DMPREG  
;BY WRITING IT TO 0

(1)  
(5)  
(4)  
(4)  
(3) 005056 000004  
(1) 005060 012777 000001 017644  
(1) 005066 012777 000004 017642  
(1) 005074 004737 023454  
(1) 005100 012777 000001 017624  
(1) 005106 017737 017624 024750  
(1) 005114 032737 000004 024750  
(3) 005122 001403  
(1) 005124 104011  
(1) 005126 004737 023526

\*\*\*\*\*  
: \*TEST 7 TEST THAT IE (PGCS BIT2) CAN BE CLEARED BY RESET  
: \*\*\*\*\*

TST7: SCOPE  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV #IE,@PGCS ;SET IE IN PGCS  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #PGCSE,@ACR ;SET ADR TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #IE,DISPLY ;CHECK FOR IE=0  
BEQ TST10 ;SKIP IS IE IS CLEARED BY RESET IN PGCS  
ERROR+ N ;IE (PGCS BIT2) CANNOT BE CLEARED BY RESET  
JSR PC,DMPREG



860  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(1)  
(1)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(1)  
(1)  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(1)  
(1)  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(1)  
(1)  
(5)  
(4)  
(4)  
(3)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(3)  
(1)  
(1)

005132 000004  
005134 004737 023454  
005140 012777 000001 017564  
005146 012777 000002 017562  
005154 012777 000001 017550  
005162 017737 017550 024750  
005170 032737 000002 024750  
005176 001003  
005200 104012  
005202 004737 023526  
  
005206 000004  
005210 012777 000001 017514  
005216 012777 000002 017512  
005224 012777 000001 017500  
005232 042777 000002 017476  
005240 012777 000001 017464  
005246 017737 017464 024750  
005254 032737 000002 024750  
005262 001403  
005264 104013  
005266 004737 023526  
  
005272 000004  
005274 012777 000001 017430  
005302 012777 000002 017426  
005310 004737 023454  
005314 012777 000001 017410  
005322 017737 017410 024750  
005330 032737 000002 024750  
005336 001403  
005340 104014  
005342 004737 023526

```
*****  
*TEST 10 TEST THAT PTP (PGCS BIT1) CAN BE SET  
*****  
TST10: SCOPE  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #PGCSE,@ACR ;SET ADR TO PGCS  
MOV #PTP,@PGCS ;SET PTP IN PGCS  
MOV #PGCSE,@ACR ;RESET ADR TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #PTP,DISPLY ;CHECK FOR PTP=1  
BNE TST11 ;SKIP IF PTP IS SET IN PGCS  
ERROR+ N ;PTP (PGCS BIT1) CANNOT BE SET  
JSR PC,DMPREG  
  
*****  
*TEST 11 TEST THAT PTP (PGCS BIT1) CAN BE CLEARED  
*****  
TST11: SCOPE  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV #PTP,@PGCS ;SET PTP IN PGCS  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
BIC #PTP,@PGCS ;CLEAR PTP IN PGCS  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #PTP,DISPLY ;CHECK FOR PTP=0  
BEQ TST12 ;SKIP IF PTP IS CLEAR IN PGCS  
ERROR+ N ;PTP (PGCS BIT1) CANNOT BE CLEARED  
JSR PC,DMPREG ;BY WRITING IT TO 0  
  
*****  
*TEST 12 TEST THAT PTP (PGCS BIT1) CAN BE CLEARED BY RESET  
*****  
TST12: SCOPE  
MOV #PGCSE,@ACR ;SET ADDRESS TO PGCS  
MOV #PTP,@PGCS ;SET PTP IN PGCS  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #PGCSE,@ACR ;SET ADR TO PGCS  
MOV @PGCS,DISPLY ;READ PGCS  
BIT #PTP,DISPLY ;CHECK FOR PTP=0  
BEQ TST13 ;SKIP IS PTP IS CLEARED BY RESET IN PGCS  
ERROR+ N ;PTP (PGCS BIT1) CANNOT BE CLEARED BY RESET  
JSR PC,DMPREG
```

```
862  
(4) :*****  
(4) :*TEST 13 SITE (BIT11-8) WRITE-READ TEST USING SITE COUNT PATTERN  
(4) :*****  
(3) 005346 000004 TST13: SCOPE  
(1) 005350 004737 023454 JSR PC,CLEAR ;CLEAR THE WORLD  
(1) 005354 012737 005370 001110 MOV #1$,SLPERR ;SET SCOPE LOOP POINTER  
(1) 005362 012737 007400 001124 MOV #7400,$GDDAT ;INIT EXPECTED TO 7400  
(1) 005370 012777 000002 017334 1$: MOV #STIE,@ACR ;SET ADR TO SITE  
(1) 005376 013777 001124 017332 MOV $GDDAT,@STIE ;LOAD SITE WITH $GDDAT  
(1) 005404 012777 000002 017320 MOV #STIE,@ACR ;RESET ADR TO SITE  
(1) 005412 017737 017320 024750 MOV @STIE,DISPLY ;READ SITE  
(1) 005420 013737 024750 001126 MOV DISPLY,$BDDAT ;COPY SITE INTO $BDDAT  
(1) 005426 042737 170377 001126 BIC #-<7400+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST  
(1) 005434 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ  
(1) 005442 001403 BEQ 2$ ;SKIP IF OK  
(1) 005444 104015 ERROR+ N ;SITE BIT11-8 LOAD-READ ERROR  
(1) 005446 004737 023526 JSR PC,DMPREG  
(1) 005452 062737 000400 001124 2$: ADD #BIT8,$GDDAT ;BUMP EXPECTED  
(1) 005460 042737 170377 001124 BIC #-<7400+1>,$GDDAT ;MASK TO BITS UNDER TEST  
(1) 005466 023727 001124 007400 CMP $GDDAT,#7400 ;DONE ALL?  
(1) 005474 001335 BNE 1$ ;TEST AGAIN IF NO  
(1) 005476 012737 005504 001110 MOV #3$,SLPERR ;CHANGE LOOP CONTROL  
(1) 005504 012777 000002 017220 3$: MOV #STIE,@ACR ;RESET ADR TO SITE  
(1) 005512 012777 007400 017216 MOV #7400,@STIE ;SET ALL BITS  
(1) 005520 004737 023454 JSR PC,CLEAR ;CLEAR THE WORLD  
(1) 005524 012777 000002 017200 MOV #STIE,@ACR ;SET ADDR TO SITE  
(1) 005532 017737 017200 024750 MOV @STIE,DISPLY ;READ SITE  
(1) 005540 032737 007400 024750 BIT #7400,DISPLY ;CHECK BIT11-8 IN SITE FOR 0  
(1) 005546 001403 BEQ 4$ ;SKIP IF RESET CLEARED BIT11-8 IN SITE  
(1) 005550 104016 ERROR+ N ;BIT11-8 IN SITE WERE NOT CLEARED BY RESET  
(1) 005552 004737 023526 JSR PC,DMPREG  
(1) 005556 4$:
```



866

(4) :  
(4) :  
(3) 005766 000004  
(1) 005770 004737 023454  
(1) 005774 012777 000003 016730  
(1) 006002 012777 000004 016726  
(1) 006010 012777 000003 016714  
(1) 006016 017737 016714 024750  
(1) 006024 032737 000004 024750  
(3) 006032 001003  
(1) 006034 104021  
(1) 006036 004737 023526  
(1)  
(5)  
(4)  
(4)

\*\*\*\*\*  
: \*TEST 15 TEST THAT LKE (STCS BIT2) CAN BE SET  
\*\*\*\*\*  
TST15: SCOPE  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV #LKE,@STCS ;SET LKE IN STCS  
MOV #STCSE,@ACR ;RESET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #LKE,DISPLY ;CHECK FOR LKE=1  
BNE TST16 ;SKIP IF LKE IS SET IN STCS  
ERROR+ N ;LKE (STCS BIT2) CANNOT BE SET  
JSR PC,DMPREG

(1)  
(5)  
(4)  
(4)  
(3) 006042 000004  
(1) 006044 012777 000003 016660  
(1) 006052 012777 000004 016656  
(1) 006060 012777 000003 016644  
(1) 006066 042777 000004 016642  
(1) 006074 012777 000003 016630  
(1) 006102 017737 016630 024750  
(1) 006110 032737 000004 024750  
(3) 006116 001403  
(1) 006120 104022  
(1) 006122 004737 023526  
(1)  
(1)  
(5)  
(4)  
(4)

\*\*\*\*\*  
: \*TEST 16 TEST THAT LKE (STCS BIT2) CAN BE CLEARED  
\*\*\*\*\*  
TST16: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #LKE,@STCS ;SET LKE IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
BIC #LKE,@STCS ;CLEAR LKE IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #LKE,DISPLY ;CHECK FOR LKE=0  
BEQ TST17 ;SKIP IF LKE IS CLEAR IN STCS  
ERROR+ N ;LKE (STCS BIT2) CANNOT BE CLEARED  
JSR PC,DMPREG  
;BY WRITING IT TO 0

(1)  
(1)  
(5)  
(4)  
(4)  
(3) 006126 000004  
(1) 006130 012777 000003 016574  
(1) 006136 012777 000004 016572  
(1) 006144 004737 023454  
(1) 006150 012777 000003 016554  
(1) 006156 017737 016554 024750  
(1) 006164 032737 000004 024750  
(3) 006172 001403  
(1) 006174 104023  
(1) 006176 004737 023526

\*\*\*\*\*  
: \*TEST 17 TEST THAT LKE (STCS BIT2) CAN BE CLEARED BY RESET  
\*\*\*\*\*  
TST17: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #LKE,@STCS ;SET LKE IN STCS  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #LKE,DISPLY ;CHECK FOR LKE=0  
BEQ TST20 ;SKIP IS LKE IS CLEARED BY RESET IN STCS  
ERROR+ N ;LKE (STCS BIT2) CANNOT BE CLEARED BY RESET  
JSR PC,DMPREG

868

(4)  
(4)  
(3) 006202 000004  
(1) 006204 004737 023454  
(1) 006210 012777 000003 016514  
(1) 006216 012777 000002 016512  
(1) 006224 012777 000003 016500  
(1) 006232 017737 016500 024750  
(1) 006240 032737 000002 024750  
(3) 006246 001003  
(1) 006250 104024  
(1) 006252 004737 023526  
(1)

```
*****  
: *TEST 20 TEST THAT STP (STCS BIT1) CAN BE SET  
: *****  
TST20: SCOPE  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV #STP,@STCS ;SET STP IN STCS  
MOV #STCSE,@ACR ;RESET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #STP,DISPLY ;CHECK FOR STP=1  
BNE TST21 ;;SKIP IF STP IS SET IN STCS  
ERROR+ N ;STP (STCS BIT1) CANNOT BE SET  
JSR PC,DMPREG
```

(5)  
(4)  
(4)  
(3) 006256 000004  
(1) 006260 012777 000003 016444  
(1) 006266 012777 000002 016442  
(1) 006274 012777 000003 016430  
(1) 006302 042777 000002 016426  
(1) 006310 012777 000003 016414  
(1) 006316 017737 016414 024750  
(1) 006324 032737 000002 024750  
(3) 006332 001403  
(1) 006334 104025  
(1) 006336 004737 023526  
(1)

```
*****  
: *TEST 21 TEST THAT STP (STCS BIT1) CAN BE CLEARED  
: *****  
TST21: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #STP,@STCS ;SET STP IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
BIC #STP,@STCS ;CLEAR STP IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #STP,DISPLY ;CHECK FOR STP=0  
BEQ TST22 ;;SKIP IF STP IS CLEAR IN STCS  
ERROR+ N ;STP (STCS BIT1) CANNOT BE CLEARED  
JSR PC,DMPREG
```

;BY WRITING IT TO 0

(5)  
(4)  
(4)  
(3) 006342 000004  
(1) 006344 012777 000003 016360  
(1) 006352 012777 000002 016356  
(1) 006360 004737 023454  
(1) 006364 012777 000003 016340  
(1) 006372 017737 016340 024750  
(1) 006400 032737 000002 024750  
(3) 006406 001403  
(1) 006410 104026  
(1) 006412 004737 023526

```
*****  
: *TEST 22 TEST THAT STP (STCS BIT1) CAN BE CLEARED BY RESET  
: *****  
TST22: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #STP,@STCS ;SET STP IN STCS  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #STP,DISPLY ;CHECK FOR STP=0  
BEQ TST23 ;;SKIP IS STP IS CLEARED BY RESET IN STCS  
ERROR+ N ;STP (STCS BIT1) CANNOT BE CLEARED BY RESET  
JSR PC,DMPREG
```

870

(4)  
(4)  
(3) 006416 000004  
(1) 006420 004737 023454  
(1) 006424 012777 000003 016300  
(1) 006432 012777 000001 016276  
(1) 006440 012777 000003 016264  
(1) 006446 017737 016264 024750  
(1) 006454 032737 000001 024750  
(3) 006462 001003  
(1) 006464 104027  
(1) 006466 004737 023526  
(1)

```
*****  
;*TEST 23 TEST THAT ENB (STCS BIT0) CAN BE SET  
*****  
TST23: SCOPE  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV #ENB,@STCS ;SET ENB IN STCS  
MOV #STCSE,@ACR ;RESET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #ENB,DISPLY ;CHECK FOR ENB=1  
BNE TST24 ;SKIP IF ENB IS SET IN STCS  
ERROR+ N ;ENB (STCS BIT0) CANNOT BE SET  
JSR PC,DMPREG
```

(5)  
(4)  
(4)  
(3) 006472 000004  
(1) 006474 012777 000003 016230  
(1) 006502 012777 000001 016226  
(1) 006510 012777 000003 016214  
(1) 006516 042777 000001 016212  
(1) 006524 012777 000003 016200  
(1) 006532 017737 016200 024750  
(1) 006540 032737 000001 024750  
(3) 006546 001403  
(1) 006550 104030  
(1) 006552 004737 023526  
(1)

```
*****  
;*TEST 24 TEST THAT ENB (STCS BIT0) CAN BE CLEARED  
*****  
TST24: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #ENB,@STCS ;SET ENB IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
BIC #ENB,@STCS ;CLEAR ENB IN STCS  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV @STCS,DISPLY ;PEAD STCS  
BIT #ENB,DISPLY ;CHECK FOR ENB=0  
BEQ TST25 ;SKIP IF ENB IS CLEAR IN STCS  
ERROR+ N ;ENB (STCS BIT0) CANNOT BE CLEARED  
JSR PC,DMPREG
```

;BY WRITING IT TO 0

(5)  
(4)  
(4)  
(3) 006556 000004  
(1) 006560 012777 000003 016144  
(1) 006566 012777 000001 016142  
(1) 006574 004737 023454  
(1) 006600 012777 000003 016124  
(1) 006606 017737 016124 024750  
(1) 006614 032737 000001 024750  
(3) 006622 001403  
(1) 006624 104031  
(1) 006626 004737 023526

```
*****  
;*TEST 25 TEST THAT ENB (STCS BIT0) CAN BE CLEARED BY RESET  
*****  
TST25: SCOPE  
MOV #STCSE,@ACR ;SET ADDRESS TO STCS  
MOV #ENB,@STCS ;SET ENB IN STCS  
JSR PC,CLEAR ;CLEAR THE INTERFACE  
MOV #STCSE,@ACR ;SET ADR TO STCS  
MOV @STCS,DISPLY ;READ STCS  
BIT #ENB,DISPLY ;CHECK FOR ENB=0  
BEQ TST26 ;SKIP IS ENB IS CLEARED BY RESET IN STCS  
ERROR+ N ;ENB (STCS BIT0) CANNOT BE CLEARED BY RESET  
JSR PC,DMPREG
```

```

872
(4)
(4)
(3) 006632 000004
(1) 006634 004737 023454
(1) 006640 012737 006654 001110
(1) 006646 012737 177400 001124
(1) 006654 012777 000003 016050 1$:
(1) 006662 013777 001124 016046
(1) 006670 012777 000003 016034
(1) 006676 017737 016034 024750
(1) 006704 013737 024750 001126
(1) 006712 042737 000377 001126
(1) 006720 023737 001124 001126
(1) 006726 001403
(1) 006730 104032
(1) 006732 004737 023526
(1) 006736 062737 000400 001124 2$:
(1) 006744 042737 000377 001124
(1) 006752 023727 001124 177400
(1) 006760 001335
(1) 006762 012737 006770 001110
(1) 006770 012777 000003 015734 3$:
(1) 006776 012777 177400 015732
(1) 007004 004737 023454
(1) 007010 012777 000003 015714
(1) 007016 017737 015714 024750
(1) 007024 032737 177400 024750
(1) 007032 001403
(1) 007034 104033
(1) 007036 004737 023526
(1) 007042

```

```

*****
*TEST 26 STCS (BIT15-8) WRITE-READ TEST USING STCS COUNT PATTERN
*****
TST26: SCOPE
JSR PC,CLEAR ;CLEAR THE WORLD
MOV #1$,SLPERR ;SET SCOPE LOOP POINTER
MOV #177400,$GDDAT ;INIT EXPECTED TO 177400
MOV #STCSE,@ACR ;SET ADR TO STCS
1$: MOV $GDDAT,@STCS ;LOAD STCS WITH $GDDAT
MOV #STCSE,@ACR ;RESET ADR TO STCS
MOV @STCS,DISPLY ;READ STCS
MOV DISPLY,$BDDAT ;COPY STCS INTO $BDDAT
BIC #-<177400+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST
CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ
BEQ 2$ ;SKIP IF OK
ERROR+ N ;STCS BIT15-8 LOAD-READ ERROR
JSR PC,DMPREG
2$: ADD #BIT8,$GDDAT ;BUMP EXPECTED
BIC #-<177400+1>,$GDDAT ;MASK TO BITS UNDER TEST
CMP $GDDAT,#177400 ;DONE ALL?
BNE 1$ ;TEST AGAIN IF NO
MOV #3$,SLPERR ;CHANGE LOOP CONTROL
3$: MOV #STCSE,@ACR ;RESET ADR TO STCS
MOV #177400,@STCS ;SET ALL BITS
JSR PC,CLEAR ;CLEAR THE WORLD
MOV #STCSE,@ACR ;SET ADDR TO STCS
MOV @STCS,DISPLY ;READ STCS
BIT #177400,DISPLY ;CHECK BIT15-8 IN STCS FOR 0
BEQ 4$ ;SKIP IF RESET CLEARED BIT15-8 IN STCS
ERROR+ N ;BIT15-8 IN STCS WERE NOT CLEARED BY RESET
JSR PC,DMPREG
4$:

```

```
874
(4)
(4)
(3) 007042 000004
(1) 007044 004737 023454
(1) 007050 012737 007064 001110
(1) 007056 012737 007400 001124
(1) 007064 012777 000004 015640
(1) 007072 013777 001124 015636
(1) 007100 012777 000004 015624
(1) 007106 017737 015624 024750
(1) 007114 013737 024750 001126
(1) 007122 042737 170377 001126
(1) 007130 023737 001124 001126
(1) 007136 001403
(1) 007140 104034
(1) 007142 004737 023526
(1) 007146 062737 000400 001124
(1) 007154 042737 170377 001124
(1) 007162 023727 001124 007400
(1) 007170 001335
(1) 007172 012737 007200 001110
(1) 007200 012777 000004 015524
(1) 007206 012777 007400 015522
(1) 007214 004737 023454
(1) 007220 012777 000004 015504
(1) 007226 017737 015504 024750
(1) 007234 032737 007400 024750
(1) 007242 001403
(1) 007244 104035
(1) 007246 004737 023526
(1) 007252
```

```
*****
:TEST 27      IMSK (BIT11-8) WRITE-READ TEST USING IMSK COUNT PATTERN
*****
TST27: SCOPE
      JSR      PC,CLEAR      ;CLEAR THE WORLD
      MOV      #1$,SLPERR    ;SET SCOPE LOOP POINTER
      MOV      #7400,$GDDAT  ;INIT EXPECTED TO 7400
1$:   MOV      #IMSK,@ACR    ;SET ADR TO IMSK
      MOV      $GDDAT,@IMSK  ;LOAD IMSK WITH $GDDAT
      MOV      #IMSK,@ACR    ;RESET ADR TO IMSK
      MOV      @IMSK,DISPLY  ;READ IMSK
      MOV      DISPLY,$BDDAT ;COPY IMSK INTO $BDDAT
      BIC      #-<7400+1>,$BDDAT;CLEAR ALL BUT BITS UNDER TEST
      CMP      $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ
      BEQ      2$           ;SKIP IF OK
      ERROR+  N            ;IMSK BIT11-8 LOAD-READ ERROR
      JSR      PC,DMPREG
2$:   ADD      #BIT8,$GDDAT  ;BUMP EXPECTED
      BIC      #-<7400+1>,$GDDAT ;MASK TO BITS UNDER TEST
      CMP      $GDDAT,#7400  ;DONE ALL?
      BNE      1$           ;TEST AGAIN IF NO
      MOV      #3$,SLPERR    ;CHANGE LOOP CONTROL
3$:   MOV      #IMSK,@ACR    ;RESET ADR TO IMSK
      MOV      #7400,@IMSK   ;SET ALL BITS
      JSR      PC,CLEAR      ;CLEAR THE WORLD
      MOV      #IMSK,@ACR    ;SET ADDR TO IMSK
      MOV      @IMSK,DISPLY  ;READ IMSK
      BIT      #7400,DISPLY  ;CHECK BIT11-8 IN IMSK FOR 0
      BEQ      4$           ;SKIP IF RESET CLEARED BIT11-8 IN IMSK
      ERROR+  N            ;BIT11-8 IN IMSK WERE NOT CLEARED BY RESET
      JSR      PC,DMPREG
4$:
```



876

(4)  
(4)  
(3) 007252 000004  
(1) 007254 004737 023454  
(1) 007260 012737 007274 001110  
(1) 007266 012737 000017 001124  
(1) 007274 012777 000004 015430  
(1) 007302 013777 001124 015426  
(1) 007310 012777 000004 015414  
(1) 007316 017737 015414 024750  
(1) 007324 013737 024750 001126  
(1) 007332 042737 177760 001126  
(1) 007340 023737 001124 001126  
(1) 007346 001403  
(1) 007350 104036  
(1) 007352 004737 023526  
(1) 007356 062737 000001 001124  
(1) 007364 042737 177760 001124  
(1) 007372 023727 001124 000017  
(1) 007400 001335  
(1) 007402 012737 007410 001110  
(1) 007410 012777 000004 015314  
(1) 007416 012777 000017 015312  
(1) 007424 004737 023454  
(1) 007430 012777 000004 015274  
(1) 007436 017737 015274 024750  
(1) 007444 032737 000017 024750  
(1) 007452 001403  
(1) 007454 104037  
(1) 007456 004737 023526  
(1) 007462

```
*****
: *TEST 30      IMSK (BIT3-0) WRITE-READ TEST USING IMSK COUNT PATTERN
: *****
TST30:  SCOPE
        JSR      PC,CLEAR      :CLEAR THE WORLD
        MOV      #1$, $LPERR   :SET SCOPE LOOP POINTER
        MOV      #17,$GDDAT    :INIT EXPECTED TO 17
1$:     MOV      #IMSKE,@ACR    :SET ADR TO IMSK
        MOV      $GDDAT,@IMSK  :LOAD IMSK WITH $GDDAT
        MOV      #IMSKE,@ACR   :RESET ADR TO IMSK
        MOV      @IMSK,DISPLY  :READ IMSK
        MOV      DISPLY,$BDDAT :COPY IMSK INTO $BDDAT
        BIC      #-<17+1>,$BDDAT: CLEAR ALL BUT BITS UNDER TEST
        CMP      $GDDAT,$BDDAT :CHECK BITS LOADED AGAINST BITS READ
        BEQ      2$           :SKIP IF OK
        ERROR+  N             :IMSK BIT3-0 LOAD-READ ERROR
        JSR      PC,DMPREG
        ADD      #BIT0,$GDDAT  :BUMP EXPECTED
        BIC      #-<17+1>,$GDDAT: MASK TO BITS UNDER TEST
        CMP      $GDDAT,#17    :DONE ALL?
        BNE      1$           :TEST AGAIN IF NO
        MOV      #3$, $LPERR   :CHANGE LOOP CONTROL
        MOV      #IMSKE,@ACR   :RESET ADR TO IMSK
        MOV      #17,@IMSK     :SET ALL BITS
3$:     JSR      PC,CLEAR      :CLEAR THE WORLD
        MOV      #IMSKE,@ACR   :SET ADDR TO IMSK
        MOV      @IMSK,DISPLY  :READ IMSK
        BIT      #17,DISPLY    :CHECK BIT3-0 IN IMSK FOR 0
        BEQ      4$           :SKIP IF RESET CLEARED BIT3-0 IN IMSK
        ERROR+  N             :BIT3-0 IN IMSK WERE NOT CLEARED BY RESET
        JSR      PC,DMPREG
4$:
```

```
878  
(4) .....  
(4) *TEST 31 MTC (BIT11-8) WRITE-READ TEST USING MTC COUNT PATTERN  
(3) .....  
TST31: SCOPE  
(1) 007462 000004 JSR PC,CLEAR ;CLEAR THE WORLD  
(1) 007464 004737 023454 MOV #1$,SLPERR ;SET SCOPE LOOP POINTER  
(1) 007470 012737 007504 001110 MOV #7400,$GDDAT ;INIT EXPECTED TO 7400  
(1) 007476 012737 007400 001124 MOV #MTC,@ACR ;SET ADR TO MTC  
(1) 007504 012777 000015 015220 1$: MOV $GDDAT,@MTC ;LOAD MTC WITH $GDDAT  
(1) 007512 013777 001124 015216 MOV #MTC,@ACR ;RESET ADR TO MTC  
(1) 007520 012777 000015 015204 MOV @MTC,DISPLY ;READ MTC  
(1) 007526 017737 015204 024750 MOV DISPLY,$BDDAT ;COPY MTC INTO $BDDAT  
(1) 007534 013737 024750 001126 BIC #-<7400+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST  
(1) 007542 042737 170377 001126 CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ  
(1) 007550 023737 001124 001126 BEQ 2$ ;SKIP IF OK  
(1) 007556 001403 ERROR+ N ;MTC BIT11-8 LOAD-READ ERROR  
(1) 007560 104040 JSR PC,DMPREG  
(1) 007562 004737 023526 JSR PC,DMPREG  
(1) 007566 062737 000400 001124 2$: ADD #BIT8,$GDDAT ;BUMP EXPECTED  
(1) 007574 042737 170377 001124 BIC #-<7400+1>,$GDDAT ;MASK TO BITS UNDER TEST  
(1) 007602 023727 001124 007400 CMP $GDDAT,#7400 ;DONE ALL?  
(1) 007610 001335 BNE 1$ ;TEST AGAIN IF NO  
(1) 007612 012737 007620 001110 MOV #3$,SLPERR ;CHANGE LOOP CONTROL  
(1) 007620 012777 000015 015104 3$: MOV #MTC,@ACR ;RESET ADR TO MTC  
(1) 007626 012777 007400 015102 MOV #7400,@MTC ;SET ALL BITS  
(1) 007634 004737 023454 JSR PC,CLEAR ;CLEAR THE WORLD  
(1) 007640 012777 000015 015064 MOV #MTC,@ACR ;SET ADDR TO MTC  
(1) 007646 017737 015064 024750 MOV @MTC,DISPLY ;READ MTC  
(1) 007654 032737 007400 024750 BIT #7400,DISPLY ;CHECK BIT11-8 IN MTC FOR 0  
(1) 007662 001403 BEQ 4$ ;SKIP IF RESET CLEARED BIT11-8 IN MTC  
(1) 007664 104041 ERROR+ N ;BIT11-8 IN MTC WERE NOT CLEARED BY RESET  
(1) 007666 004737 023526 JSR PC,DMPREG  
(1) 007672 4$: JSR PC,DMPREG
```

```
880
(4)
(4)
(3) 007672 000004
(1) 007674 004737 023454
(1) 007700 012737 007714 001110
(1) 007706 012737 000017 001124
(1) 007714 012777 000015 015010
(1) 007722 013777 001124 015006
(1) 007730 012777 000015 014774
(1) 007736 017737 014774 024750
(1) 007744 013737 024750 001126
(1) 007752 042737 177760 001126
(1) 007760 023737 001124 001126
(1) 007766 001403
(1) 007770 104042
(1) 007772 004737 023526
(1) 007776 062737 000001 001124
(1) 010004 042737 177760 001124
(1) 010012 023727 001124 000017
(1) 010020 001335
(1) 010022 012737 010030 001110
(1) 010030 012777 000015 014674
(1) 010036 012777 000017 014672
(1) 010044 004737 023454
(1) 010050 012777 000015 014654
(1) 010056 017737 014654 024750
(1) 010064 032737 000017 024750
(1) 010072 001403
(1) 010074 104043
(1) 010076 004737 023526
(1) 010102
881
882 010102 004737 024634
```

```
*****
*TEST 32 MTC (BIT3-0) WRITE-READ TEST USING MTC COUNT PATTERN
*****
TST32: SCOPE
JSR PC,CLEAR ;CLEAR THE WORLD
MOV #1$,SLPERR ;SET SCOPE LOOP POINTER
MOV #17,$GDDAT ;INIT EXPECTED TO 17
1$: MOV #MTC,@ACR ;SET ADR TO MTC
MOV $GDDAT,@MTC ;LOAD MTC WITH $GDDAT
MOV #MTC,@ACR ;RESET ADR TO MTC
MOV @MTC,DISPLY ;READ MTC
MOV DISPLY,$BDDAT ;COPY MTC INTO $BDDAT
BIC #-<17+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST
CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ
BEQ 2$ ;SKIP IF OK
ERROR+ N ;MTC BIT3-0 LOAD-READ ERROR
JSR PC,DMPREG
2$: ADD #BIT0,$GDDAT ;BUMP EXPECTED
BIC #-<17+1>,$GDDAT ;MASK TO BITS UNDER TEST
CMP $GDDAT,#17 ;DONE ALL?
BNE 1$ ;TEST AGAIN IF NO
MOV #3$,SLPERR ;CHANGE LOOP CONTROL
3$: MOV #MTC,@ACR ;RESET ADR TO MTC
MOV #17,@MTC ;SET ALL BITS
JSR PC,CLEAR ;CLEAR THE WORLD
MOV #MTC,@ACR ;SET ADDR TO MTC
MOV @MTC,DISPLY ;READ MTC
BIT #17,DISPLY ;CHECK BIT3-0 IN MTC FOR 0
BEQ 4$ ;SKIP IF RESET CLEARED BIT3-0 IN MTC
ERROR+ N ;BIT3-0 IN MTC WERE NOT CLEARED BY RESET
JSR PC,DMPREG
4$:
JSR PC,GETSID ;DUMP SID ON PASS 1
```

```
884 :DUAL ADDRESSING TEST
885 :A COUNT PATTERN IS DRIVEN THROUGH THE REGISTER UNDER TEST
886 :AND ALL OTHER REGISTERS ARE CHECKED FOR ANY CHANGE
887 :ALL THE IIST REGISTERS ARE TESTED SEQUENTIALLY
888 :THIS TEST IS NOT PERFORMED IF THERE ARE OTHER TRANSMITTING
889 :NON-INHIBITED UNITS ON THE IIST BUS (CONFIG=3), SINCE THE
890 :REGISTER SNAPSHOT COULD BE INVALID.
891
892 :*****
(3) :*TEST 33 DUAL ADDRESSING TEST OF ALL REGISTERS
(3) :*****
(2) 010106 000004 IIST33: SCOPE
893 010110 023727 025020 000003 CMP CONFIG,#3 ;SEE IF CONFIG #3 (OTHER ACTIVE UNITS)
894 010116 001002 BNE 11$ ;SKIP IF NO
895 010120 000137 010536 JMP 99$ ;DON'T DO TEST IF YES
896 010124 012705 000020 11$: MOV #16.,R5 ;SET COUNTER FOR 16 REGISTERS
897 010130 012704 000000 MOV #0,R4 ;SET POINTER FOR REGISTER ADDRESS
898 010134 012703 024756 MOV #SNAPO,R3 ;SET POINTER FOR INITIAL REGISTER DATA
899 010140 004737 023450 JSR PC,CLEARX ;CLEAR THE IIST
900 010144 005037 001176 CLR $TMP0 ;CLEAR THE REGISTER NUMBER
901 010150 110477 014556 1$: MOVB R4,@ACR ;SELECT REGISTER FROM TABLE
902 010154 017723 014556 MOV @ADR,(3)+ ;PUT REGISTER CONTENTS INTO SNAP
903 010160 005204 INC R4 ;BUMP ADR CODE
904 010162 005305 DEC R5 ;DONE 10?
905 010164 001371 BNE 1$ ;DO 10
906 010166 005000 CLR R0 ;CLEAR TEST DATA WORD
907 010170 010077 014536 2$: MOV R0,@ACR ;LOAD DATA INTO REGISTER
908 010174 005200 INC R0 ;BUMP DATA
909 010176 100374 BPL 2$ ;DO 0-7777. (DO NOT SET "CLR")
910 010200 012705 000020 MOV #16.,R5 ;SET COUNTER FOR 16 REGISTERS
911 010204 012704 000000 MOV #0,R4 ;SET POINTER FOR REGISTER ADDRESS TABLE
912 010210 012703 024756 MOV #SNAPO,R3 ;SET POINTER FOR INITIAL REGISTER DATA
913 010214 110437 001176 3$: MOVB R4,$TMP0 ;SELECT REGISTER FROM TABLE
914 010220 013777 001176 014504 MOV $TMP0,@ACR ;LOAD ACR WITH ADDRESS
915 010226 017737 014504 001126 MOV @ADR,$BDDAT ;READ REGISTER INTO $BDDAT
916 010234 012337 001124 MOV (3)+,$GDDAT ;FETCH ORIGINAL REGISTER DATA
917 010240 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR ANY CHANGE IN REGISTER
918 010246 001403 BEQ 4$ ;SKIP IF NO CHANGE
919 010250 104044 ERROR+ 44 ;ADDRESSING ACR ALSO ADDRESSED REGISTER IN $TMP0
920 010252 004737 023526 JSR PC,DMPREG
921 010256 005204 4$: INC R4 ;BUMP ADR CODE
922 010260 005305 DEC R5 ;CHECKED ALL REGISTERS?
923 010262 001354 BNE 3$ ;CHECK ALL REGISTERS
924 010264 012705 000020 MOV #16.,R5 ;SET COUNTER FOR 16 REGISTERS
925 010270 012704 000000 MOV #0,R4 ;SET POINTER FOR REGISTER ADDRESS
926 010274 012737 010306 001110 MOV #5$,$LPERR ;SET SCOPE LOOP POINT
927 010302 005037 001204 CLR $TMP3 ;INITIALIZE $TMP3
928 010306 012777 100000 014416 5$: MOV #CLR,@ACR ;CLEAR THE IIST
929 010314 004737 023450 JSR PC,CLEARX
930 010320 012703 024756 MOV #SNAPO,R3 ;SET POINTER FOR INITIAL REGISTER DATA
931 010324 012701 000020 MOV #16.,R1 ;SET COUNTER TO READ 16 REGISTERS
932 010330 012702 000000 MOV #0,R2 ;SET POINTER FOR REGISTER ADDRESS
933 010334 110277 014372 6$: MOVB R2,@ACR ;SELECT REGISTER
934 010340 017723 014372 MOV @ADR,(3)+ ;READ INITIAL REGISTER DATA
935 010344 005202 INC R2 ;BUMP ADR CODE
936 010346 005301 DEC R1 ;DONE 16?
```

937 01350 001371  
938 010352 110437 001176  
939 010356 005000

BNE 68  
MOVB R4,\$TMPO  
CLR R0

;DO 16  
;\$TMPO=REGISTER BEING WRITTEN  
;START AT 0

941	010360	113777	001176	014344	7\$:	MOVB	\$TMP0,@ACR	:SELECT REGISTER
942	010366	01000i				MOV	R0,R1	:COPY R0
943	010370	120427	000001			CMPB	R4,#PGCSE	:PGCS?
944	010374	001411				BEQ	10\$	:SKIP IF YES
945	010376	120427	000003			CMPB	R4,#STCSE	:STCS?
946	010402	001406				BEQ	10\$	:SKIP IF YES
947	010404	120427	000015			CMPB	R4,#MTCSE	:MTC?
948	010410	001005				BNE	20\$	:SKIP IF NO
949	010412	042701	000007			BIC	#7,R1	:CLEAR MAINT BITS
950	010416	000402				BR	20\$	:SKIP
951	010420	042701	000001		10\$:	BIC	#1,R1	:DO NOT SET GO
952	010424	010177	014306		20\$:	MOV	R1,@ADR	:LOAD DATA INTO REGISTER
953	010430	005200				INC	R0	:BUMP DATA
954	010432	001352				BNE	7\$	:LOAD 0-77777 INTO REGISTER
955	010434	012702	000000			MOV	#0,R2	:SET POINTER FOR REGISTER ADDRESS TABLE
956	010440	012701	000020			MOV	#16.,R1	:SET COUNTER FOR 16 REGISTERS
957	010444	012703	024756			MOV	#SNAP0,R3	:SET POINTER FOR INITIAL CONTENTS TABLE
958	010450	110277	014256		8\$:	MOVB	R2,@ACR	:SELECT A REGISTER
959	010454	012337	001124			MOV	(3)+,\$GDDAT	:FETCH INITIAL CONTENTS OF REGISTER
960	010460	127737	014246	001176		CMPB	@ACR,\$TMP0	:IS SELECTED REGISTER THE ONE WRITTEN?
961	010466	001415				BEQ	9\$	:SKIP TESTING IF YES
962	010470	117737	014236	001200		MOVB	@ACR,\$TMP1	:SAVE ADDRESS OF REGISTER CHECKED IN \$TMP1
963	010476	017737	014234	001126		MOV	@ADR,\$BDDAT	:FETCH REGISTER DATA
964	010504	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	:CHECK REGISTER FOR ANY CHANGES
965	010512	001403				BEQ	9\$	:SKIP IF NO CHANGE
966	010514	104044				ERROR+	44	:ADDRESSING REGISTER IN \$TMP0 ALSO
967	010516	004737	023526			JSR	PC,DMPREG	
968								:ADDRESSED REGISTER IN \$TMP1
969	010522	005202			9\$:	INC	R2	:BUMP ADR CODE
970	010524	005301				DEC	R1	:CHECKED ALL REGISTERS?
971	010526	001350				BNE	8\$	:CHECK ALL REGISTERS
972	010530	005204				INC	R4	:BUMP POINTER FOR REGISTER ADDRESS
973	010532	005305				DEC	R5	:DONE ALL?
974	010534	001264				BNE	5\$	:DO ALL
975	010536				99\$:			

```
977 .SBTTL
978 .SBTTL
979 .SBTTL INDIVIDUAL LINE TESTS
980 .SBTTL
981
982 010536 012704 000000 LINE0: MOV #0,R4 ;SET INDEX TO 0
983 010542 004737 010626 JSR PC,LINTST ;TEST LINE
984 010546 012704 000001 MOV #1,R4 ;SET INDEX TO 1
985 010552 004737 010626 JSR PC,LINTST ;TEST LINE
986 010556 012704 000002 MOV #2,R4 ;SET INDEX TO 2
987 010562 004737 010626 JSR PC,LINTST ;TEST LINE
988 010566 012704 000003 MOV #3,R4 ;SET INDEX TO 3
989 010572 004737 010626 JSR PC,LINTST ;TEST LINE
990 010576 013704 024730 LINE1: MOV SIDNUM,R4 ;SET INDEX TO ID #
991 010602 006304 ASL R4 ;SHIFT
992 010604 004737 017400 JSR PC,PGONL ;TEST PGF ONLINE
993 010610 013704 024730 LINE2: MOV SIDNUM,R4 ;SET INDEX TO ID #
994 010614 006304 ASL R4 ;SHIFT
995 010616 004737 021134 JSR PC,STONL ;TEST STF ONLINE
996 010622 000137 022012 JMP ENDTST ;SKIP OVER SUBROUTINE
```

```

998 010626 010437 024754 LINTST: MOV R4,LINMBR ;SAVE LINE UNDER TEST
999 010632 006304 ASL R4 ;MAKE INDEX INTO A WORD POINTER
1000
1001 ;ISSUE A PROGRAM INTERRUPT REQUEST AND CHECK FOR:
1002 ;RDY TO RESET
1003 ;RDY TO SET
1004 ;PIF BIT TO SET
1005 ;DCF REGISTER TO STAY RESET
1006 ;EXC REGISTER TO STAY RESET
1007
1008
(3)
(3)
(2) 010634 000004
1009 010636 005037 177776 CLR PSW ;LOWER PRIORITY TO ALLOW INTERUPTS
1010 010642 004737 023306 JSR PC,SETMNT ;ENABLE MAINT LOOPBACK
1011 010646 012777 000000 014056 MOV #PGTEE,@ACR ;SELECT PGTE
1012 010654 016477 021772 014054 MOV PGITAB(4),@PGTE ;LOAD PIE
1013 010662 012777 000001 014046 MOV #GO,@PGCS ;TRANSMIT
1014 010670 012777 000001 014034 MOV #PGCSE,@ACR ;SELECT PGCS
1015 010676 017737 014034 001126 MOV @PGCS,$BDDAT ;READ PGCS
1016 010704 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1017 010712 001406 BEQ 1$ ;=0
1018 010714 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED PGCS
1019 010722 104044 ERROR+ 44 ;SETTING PGCS GO DID NOT CLEAR RDY
1020 010724 004737 023526 JSR PC,DMPREG
1021 010730 005000 1$: CLR R0 ;INITIALIZE TIMER
1022 010732 012777 000001 013772 2$: MOV #PGCSE,@ACR ;SELECT PGCS
1023 010740 017737 013772 001126 MOV @PGCS,$BDDAT ;READ PGCS
1024 010746 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1025 010754 001013 BNE 3$ ;SKIP IF SET
1026 010756 005300 DEC R0 ;TIMER =0?
1027 010760 001364 BNE 2$ ;TRY AGAIN IF NO
1028 010762 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
1029 010770 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY
1030 010776 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM PI
1031 011000 004737 023526 JSR PC,DMPREG
1032 011004 012777 000005 013720 3$: MOV #PGFE,@ACR ;SELECT PGF
1033 011012 017737 013720 001126 MOV @PGF,$BDDAT ;READ PGF
1034 011020 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAG BITS
1035 011026 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
1036 011034 001403 BEQ 4$ ;SKIP IF OK
1037 011036 104044 ERROR+ 44 ;PGM PI DID NOT SET CORRECT FLAG
1038 011040 004737 023526 JSR PC,DMPREG
1039 011044 012777 000007 013660 4$: MOV #DCFE,@ACR ;SELECT DCF
1040 011052 017737 013660 024750 MOV @DCF,DISPLY ;READ DCF
1041 011060 001403 BEQ 5$ ;SKIP IF 0
1042 011062 104044 ERROR+ 44 ;PGM PI SET A DCF CONDITION
1043 011064 004737 023526 JSR PC,DMPREG
1044 011070 012777 000010 013634 5$: MOV #EXCE,@ACR ;SELECT EXC
1045 011076 017737 013634 024750 MOV @EXC,DISPLY ;READ EXC
1046 011104 001403 BEQ 6$ ;SKIP IF 0
1047 011106 104044 ERROR+ 44 ;PGM PI SET AN EXC CONDITION
1048 011110 004737 023526 JSR PC,DMPREG
1049 011114 012777 000006 013610 6$: MOV #STFE,@ACR ;SELECT STF
1050 011122 017737 013610 024750 MOV @STF,DISPLY ;READ STF
    
```



1051 011130 001403  
1052 011132 104044  
1053 011134 004737 023526

BEQ 7\$ :SKIP IF CLEAR  
ERROR+ 44 :PGM PI SET AN STF CONDITION  
JSR PC,DMPREG

K 4

MAINDEC-11-CR11A-B      MACY11 30A(1052) 18-JUN-79 14:32 PAGE 36  
 CR11AB.P11      18-JUN-79 14:24      T34      CHECK A LEGAL PGM PI REQUEST FOR PROPER OPERATION      SEQ 0049

1055	011140	012700	000017		7\$:	MOV	#17,RO	:SET ALL 4 LINES
1056	011144	046400	021772			BIC	PGITAB(4),RO	:CLEAR LINE UNDER TEST
1057	011150	012777	000005	013554		MOV	#PGFE,@ACR	:SELECT PGF
1058	011156	010077	013554			MOV	RO,@PGF	:CLEAR ALL PIFS EXCEPT LINE UNDER TEST
1059	011162	012777	000005	013542		MOV	#PGFE,@ACR	:SELECT PGF
1060	011170	017737	013542	001202		MOV	@PGF,\$TMP2	:READ PGF
1061	011176	012777	000001	013526		MOV	#PGCSE,@ACR	:SELECT PGCS
1062	011204	017737	013526	024750		MOV	@PGCS,DISPLY	:READ PGCS
1063	011212	032737	000010	024750		BIT	#IP,DISPLY	:CHECK FOR IP
1064	011220	001003				BNE	8\$	:SKIP IF SET
1065	011222	104044				ERROR+	44	:IP DID NOT SET ON PGF
1066	011224	004737	023526			JSR	PC,DMPREG	
1067	011230	017746	013510		8\$:	MOV	@111P,-(SP)	:SAVE OLD
1068	011234	017746	013502			MOV	@111V,-(SP)	:PI INFORMATION
1069	011240	012777	011272	013474		MOV	#9\$,@111V	:LOAD NEW PI VECTOR
1070	011246	012777	000340	013470		MOV	#340,@111P	:LOAD NEW INT. LEVEL
1071	011254	005037	177776			CLR	PS	:ALLOW PI
1072	011260	000240				NOP		:STALL
1073	011262	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
1074	011270	000404				BR	10\$	:SKIP OVER ERROR
1075	011272	022626			9\$:	POPPOP		:FIX STACK
1076	011274	104044				ERROR+	44	:PGF PI WITH IE=0
1077	011276	004737	023526			JSR	PC,DMPREG	
1078	011302	012777	011376	013432	10\$:	MOV	#12\$,@111V	:LOAD NEW PI VECTOR
1079	011310	012777	000001	013414		MOV	#PGCSE,@ACR	:SELECT PGCS
1080	011316	012777	000004	013412		MOV	#IE,@PGCS	:SET IE
1081	011324	012737	000340	001212		MOV	#340,\$TMP6	:INIT PI LEVEL
1082	011332	162737	000040	001212	11\$:	SUB	#40,\$TMP6	:DROP PI LEVEL BY 1
1083	011340	013737	001212	177776		MOV	\$TMP6,PS	:LOWER PSW
1084	011346	000240				NOP		:STALL
1085	011350	023727	001212	000140		CMP	\$TMP6,#140	:TRIED ALL LEVELS?
1086	011356	001365				BNE	11\$	:BR IF NO
1087	011360	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
1088	011366	104044				ERROR+	44	:NO PGF PI WITH IE-1
1089	011370	004737	023526			JSR	PC,DMPREG	
1090	011374	000422				BR	13\$	:SKIP NEXT TEST
1091	011376	022626			12\$:	POPPOP		:FIX STACK
1092	011400	006337	001212			ASL	\$TMP5	:RIGHT JUSTIFY
1093	011404	006337	001212			ASL	\$TMP6	
1094	011410	006337	001212			ASL	\$TMP6	
1095	011414	000337	001212			SWAB	\$TMP6	:THE PI LEVEL
1096	011420	005237	001212			INC	\$TMP6	:ADD 1 TO MAKE CORECT LEVEL
1097	011424	023737	001212	024746		CMP	\$TMP6,111L	:CHECK AGAINST KNOWN LEVEL
1098	011432	001403				BEQ	13\$	:SKIP IF OK
1099	011434	104044				ERROR+	44	:PGF PI TO THE WRONG LEVEL
1100	011436	004737	023526			JSR	PC,DMPREG	
1101	011442	012677	013274		13\$:	MOV	(SP)+,@111V	:RESTORE
1102	011446	012677	013272			MOV	(SP)+,@111P	:OLD PI INFO
1103	011452	012777	000005	013252	14\$:	MOV	#PGFE,@ACR	:SELECT PGF
1104	011460	016437	021772	001124		MOV	PIFTAB(4),\$GDDAT	:FETCH FLAG BIT
1105	011466	013777	001124	013242		MOV	\$GDDAT,@PGF	:CLEAR FLAG
1106	011474	005037	001124			CLR	\$GDDAT	:CLEAR EXPECTED
1107	011500	012777	000005	013224		MOV	#PGFE,@ACR	:SELECT PGF
1108	011506	017737	013224	001126		MOV	@PGF,\$BDDAT	:READ PGF
1109	011514	001403				BEQ	15\$	:SKIP IF OK
1110	011516	104044				ERROR+	44	:WRITING A 1 TO PGF FLAG DID NOT

MAINDEC-11-CR11A-B MACY11 30A(1052) 18-JUN-79 14:32 PAGE 36-1  
CR11AB.P11 18-JUN-79 14:24 T34 CHECK A LEGAL PGM PI REQUEST FOR PROPER OPERATION

SEQ 0050

1111 011520 004737 023526  
1112  
1113 011524

JSR PC,DMPREG

;CLEAR IT

158:

```
1115 ;ISSUE A PROGRAM BOOT REQUEST AND CHECK FOR:  
1116 ;RDY TO SET  
1117 ;PBF BIT TO SET  
1118 ;DCF REGISTER TO STAY RESET  
1119 ;EXC REGISTER TO STAY RESET  
1120  
1121 ;*****  
  (3) ;*TEST 35 CHECK A LEGAL PGM BOOT REQUEST FOR PROPER OPERATION  
  (3) ;*****  
  (2) 011524 000004 TST35: SCOPE  
1122 011526 004737 023306 JSR PC,SETMNT ;ENABLE MAINT LOOPBACK  
1123 011532 012777 000000 013172 MOV #PGTEE,@ACR ;SELECT PGTE  
1124 011540 016477 022002 013170 MOV PGBTAB(4),@PGTE ;LOAD PIB  
1125 011546 012777 000001 013162 MOV #GO,@PGCS ;TRANSMIT  
1126 011554 005000 1$: CLR RO ;INITIALIZE TIMER  
1127 011556 012777 000001 013146 2$: MOV #PGCSE,@ACR ;SELECT PGCS  
1128 011564 017737 013146 001126 MOV @PGCS,$BDDAT ;READ PGCS  
1129 011572 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY  
1130 011600 001013 BNE 3$ ;SKIP IF SET  
1131 011602 005300 DEC RO ;TIMER =0?  
1132 011604 001364 BNE 2$ ;TRY AGAIN IF NO  
1133 011606 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH  
1134 011614 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY  
1135 011622 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM BOOT  
1136 011624 004737 023526 JSR PC,DMPREG  
1137 011630 012777 000005 013074 3$: MOV #PGFE,@ACR ;SELECT PGF  
1138 011636 017737 013074 001126 MOV @PGF,$BDDAT ;READ PGF  
1139 011644 012737 007400 001124 MOV #7400,$GDDAT ;FETCH EXPECTED FLAG BITS  
1140 011652 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG  
1141 011660 001403 BEQ 4$ ;SKIP IF OK  
1142 011662 104044 ERROR+ 44 ;PGM BOOT DID NOT SET CORRECT FLAG  
1143 011664 004737 023526 JSR PC,DMPREG  
1144 011670 012777 000007 013034 4$: MOV #DCFE,@ACR ;SELECT DCF  
1145 011676 017737 013034 024750 MOV @DCF,DISPLY ;READ DCF  
1146 011704 001403 BEQ 5$ ;SKIP IF 0  
1147 011706 104044 ERROR+ 44 ;PGM BOOT SET A DCF CONDITION  
1148 011710 004737 023526 JSR PC,DMPREG  
1149 011714 012777 000010 013010 5$: MOV #EXCE,@ACR ;SELECT EXC  
1150 011722 017737 013010 024750 MOV @EXC,DISPLY ;READ EXC  
1151 011730 001403 BEQ 6$ ;SKIP IF 0  
1152 011732 104044 ERROR+ 44 ;PGM BOOT SET AN EXC CONDITION  
1153 011734 004737 023526 JSR PC,DMPREG  
1154 011740 012777 000006 012764 6$: MOV #STFE,@ACR ;SELECT STF  
1155 011746 017737 012764 024750 MOV @STF,DISPLY ;READ STF  
1156 011754 001403 BEQ 7$ ;SKIP IF 0  
1157 011756 104044 ERROR+ 44 ;PGM BOOT SET AN STF CONDITION  
1158 011760 004737 023526 JSR PC,DMPREG  
1159 011764 012777 000005 012740 7$: MOV #PGFE,@ACR ;SELECT PGF  
1160 011772 016437 022002 001124 MOV PBFTAB(4),$GDDAT ;FETCH FLAG BIT  
1161 012000 013777 001124 012730 MOV $GDDAT,@PGF ;CLEAR FLAG  
1162 012006 005137 001124 COM $GDDAT ;FLIP GDDAT  
1163 012012 042737 170377 001124 BIC #170377,$GDDAT ;MASK OUT JUNK  
1164 012020 012777 000005 012704 MOV #PGFE,@ACR ;SELECT PGF  
1165 012026 017737 012704 001126 MOV @PGF,$BDDAT ;READ PGF  
1166 012034 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0  
1167 012042 001403 BEQ 8$ ;SKIP IF OK
```

MAINDEC-11-CRIIA-B MACY11 30A(1052) 18-JUN-79 14:32 PAGE 37-1 N 4  
CRIIAB.P11 18-JUN-79 14:24 T35 CHECK A LEGAL PGM BOOT REJUEST FOR PROPER OPERATION SEQ 0052  
1168 012044 104044 ERROR+ 44 ;WRITING A 1 TO PGF FLAG DID NOT  
1169 012046 C04737 023526 JSR PC,DMPREG ;CLEAR IT  
1170  
1171 012052 8\$:

```
1173 ;ISSUE A SANITY TIMER INTERRUPT REQUEST AND CHECK FOR:
1174 ;RDY TO RESET
1175 ;RDY TO SET
1176 ;SIF BIT TO SET
1177 ;DCF REGISTER TO STAY RESET
1178 ;EXC REGISTER TO STAY RESET
1179 ;PGF REGISTER TO STAY RESET
1180 ;SIF BIT TO BE CLEARED BY WRITING A 1 TO IT
1181
1182 ;*****
(3) ;+TEST 36 CHECK A LEGAL ST INTR REQUEST FOR PROPER OPERATION
(3) ;*****
(2) 012052 000004
1183 012054 004737 023476 JSR PC,SSTMNT ;ENABLE MAINT LOOPBACK
1184 012060 012777 000000 012644 MOV #PGTEE,@ACR ;SELECT PGTE
1185 012066 016477 021772 012642 MOV PGITAB(4),@PGTE ;LOAD PIE
1186 012074 012777 000003 012634 MOV #PTP+GO,@PGCS ;TRANSMIT
1187 012102 012777 000001 012622 MOV #PGCSE,@ACR ;SELECT PGCS
1188 012110 017737 012622 001126 MOV @PGCS,$BDDAT ;READ PGCS
1189 012116 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1190 012124 001406 BEQ 1$ ;=0
1191 012126 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED PGCS
1192 012134 104044 ERROR+ 44 ;SETTING PGCS GO DID NOT CLEAR RDY
1193 012136 004737 023526 JSR PC,DMPREG
1194 012142 005000 1$: CLR RO ;INITIALIZE TIMER
1195 012144 012777 000001 012560 2$: MOV #PGCSE,@ACR ;SELECT PGCS
1196 012152 017737 012560 001126 MOV @PGCS,$BDDAT ;READ PGCS
1197 012160 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1198 012166 001013 BNE 3$ ;SKIP IF SET
1199 012170 005300 DEC RO ;TIMER =0?
1200 012172 001364 BNE 2$ ;TRY AGAIN IF NO
1201 012174 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
1202 012202 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY
1203 012210 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM PI
1204 012212 004737 023526 JSR PC,DMPREG
1205 012216 012777 000006 012506 3$: MOV #STFE,@ACR ;SELECT STF
1206 012224 017737 012506 001126 MOV @STF,$BDDAT ;READ STF
1207 012232 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAG BITS
1208 012240 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
1209 012246 001403 BEQ 4$ ;SKIP IF OK
1210 012250 104044 ERROR+ 44 ;ST INTR DID NOT SET CORRECT FLAG
1211 012252 004737 023526 JSR PC,DMPREG
1212 012256 012777 000007 012446 4$: MOV #DCFE,@ACR ;SELECT DCF
1213 012264 017737 012446 024750 MOV @DCF,DISPLY ;READ DCF
1214 012272 001403 BEQ 5$ ;SKIP IF 0
1215 012274 104044 ERROR+ 44 ;ST INTR SET A DCF CONDITION
1216 012276 004737 023526 JSR PC,DMPREG
1217 012302 012777 000010 012422 5$: MOV #EXCE,@ACR ;SELECT EXC
1218 012310 017737 012422 024750 MOV @EXC,DISPLY ;READ EXC
1219 012316 001403 BEQ 6$ ;SKIP IF 0
1220 012320 104044 ERROR+ 44 ;ST INTR SET AN EXC CONDITION
1221 012322 004737 023526 JSR PC,DMPREG
1222 012326 012777 000005 012376 6$: MOV #PGFE,@ACR ;SELECT PGF
1223 012334 017737 012376 024750 MOV @PGF,DISPLY ;READ PGF
1224 012342 001403 BEQ 7$ ;SKIP IF CLEAR
1225 012344 104044 ERROR+ 44 ;ST INTR SET A PGF CONDITION
```

MAINDEC-11-CR11A-B MACY11 30A(1052) 18-JUN-79 14:32 PAGE 38-1  
CR11AB.P11 18-JUN-79 14:24 T36 CHECK A LEGAL ST INTR REQJEST FOR PROPER OPERATION

SEQ 0054

1226 012346 004737 023526

JSR PC,DMPREG

```

1228 012352 012700 000017      7$:  MOV      #17,RO      ;SET ALL 4 LINES
1229 012356 046400 021772      BIC      S1TAB(4),RO  ;CLEAR LINE UNDER TEST
1230 012362 012777 000006 012342  MOV      #STFE,@ACR   ;SELECT STF
1231 012370 010077 012342      MOV      RO,@STF     ;CLEAR ALL SIFS EXCEPT LINE UNDER TEST
1232 012374 012777 000006 012330  MOV      #STFE,@ACR   ;SELECT STF
1233 012402 017737 012330 001202  MOV      @STF,$TMP2   ;READ STF
1234 012410 012777 000001 012314  MOV      #PGCSE,@ACR  ;SELECT PGCS
1235 012416 017737 012314 024750  MOV      @PGCS,DISPLY ;READ PGCS
1236 012424 032737 000010 024750  BIT      #IP,DISPLY   ;CHECK FOR IP
1237 012432 001003      BNE      8$          ;SKIP IF SET
1238 012434 104044      ERROR+  44           ;IP DID NOT SET ON STF
1239 012436 004737 023526      JSR      PC,DMPREG
1240 012442 017746 012276      8$:  MOV      @11IP,-(SP)  ;SAVE OLD
1241 012446 017746 012270      MOV      @11IV,-(SP)  ;PI INFORMATION
1242 012452 012777 012504 012262  MOV      #9$,@11IV    ;LOAD NEW PI VECTOR
1243 012460 012777 000340 012256  MOV      #340,@11IP   ;LOAD NEW INT. LEVEL
1244 012466 005037 177776      CLR      PS          ;ALLOW PI
1245 012472 000240      NOP
1246 012474 012737 000340 177776  MOV      #340,PS      ;LOCKOUT PI
1247 012502 000404      BR       10$        ;SKIP OVER ERROR
1248 012504 022626      9$:  POPPOP
1249 012506 104044      ERROR+  44           ;FIX STACK
1250 012510 004737 023526      JSR      PC,DMPREG   ;STF PI WITH IE=0
1251 012514 012777 012562 012220 10$:  MOV      #11$,@11IV   ;LOAD NEW PI VECTOR
1252 012522 012777 000001 012202  MOV      #PGCSE,@ACR  ;SELECT PGCS
1253 012530 012777 000004 012200  MOV      #IE,@PGCS    ;SET IE
1254 012536 005037 177776      CLR      PS          ;ALLOW PI
1255 012542 000240      NOP
1256 012544 012737 000340 177776  MOV      #340,PS      ;LOCKOUT PI
1257 012552 104044      ERROR+  44           ;NO STF PI WITH IE=1
1258 012554 004737 023526      JSR      PC,DMPREG
1259 012560 000401      BR       12$        ;SKIP NEXT TEST
1260 012562 022626      11$: POPPOP
1261 012564 012677 012152 12$:  MOV      (SP)+,@11IV   ;FIX STACK
1262 012570 012677 012150      MOV      (SP)+,@11IP  ;RESTORE
1263 012574 012777 000006 012130 13$:  MOV      #STFE,@ACR   ;OLD P' INFO
1264 012602 016437 021772 001124  MOV      S1TAB(4),$GDDAT ;SELECT STF
1265 012610 013777 001124 012120  MOV      $GDDAT,@STF  ;FETCH FLAG BIT
1266 012616 005037 001124      CLR      $GDDAT     ;CLEAR FLAG
1267 012622 012777 000006 012102  MOV      #STFE,@ACR   ;CLEAR EXPECTED
1268 012630 017737 012102 001126  MOV      @STF,$BDDAT  ;SELECT STF
1269 012636 001403      BEQ     14$        ;READ STF
1270 012640 104044      ERROR+  44           ;SKIP IF OK
1271 012642 004737 023526      JSR      PC,DMPREG   ;WRITING A 1 TO STF FLAG DID NOT
1272      ;CLEAR IT
1273 012646      14$:
  
```



```

1275 ;ISSUE A SANITY TIMER BOOT REQUEST AND CHECK FOR:
1276 ;RDY TO SET
1277 ;SBF BIT TO SET
1278 ;DCF REGISTER TO STAY RESET
1279 ;EXC REGISTER TO STAY RESET
1280
1281 ;*****
(3) ;+TEST 37 CHECK A LEGAL ST BOOT REQUEST FOR PROPER OPERATION
(3) ;*****
(2) 012646 000004 TST37: SCOPE
1282 012650 004737 023476 JSR PC,SSTMNT ;ENABLE MAINT LOOPBACK
1283 012654 012777 000000 012050 MOV #PGTEE,@ACR ;SELECT PGTE
1284 012662 016477 022002 012046 MOV PGBTAB(4),@PGTE ;LOAD PIB
1285 012670 012777 000003 012040 MOV #PTP+GO,@PGCS ;TRANSMIT
1286 012676 005000 1$: CLR RO ;INITIALIZE TIMER
1287 012700 012777 000001 012024 2$: MOV #PGCSE,@ACR ;SELECT PGCS
1288 012706 017737 012024 001126 MOV @PGCS,$BDDAT ;READ PGCS
1289 012714 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1290 012722 001013 BNE 3$ ;SKIP IF SET
1291 012724 005300 DEC RO ;TIMER =0?
1292 012726 001364 BNE 2$ ;TRY AGAIN IF NO
1293 012730 016437 021762 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
1294 012736 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY
1295 012744 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A ST BOOT
1296 012746 004737 023526 JSR PC,DMPREG
1297 012752 012777 000006 011752 3$: MOV #STFE,@ACR ;SELECT STF
1298 012760 017737 011752 001126 MOV @STF,$BDDAT ;READ STF
1299 012766 012737 007400 001124 MOV #7400,$GDDAT ;FETCH EXPECTED FLAG BITS
1300 012774 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
1301 013002 001403 BEQ 4$ ;SKIP IF OK
1302 013004 104044 ERROR+ 44 ;ST BOOT DID NOT SET CORRECT FLAG
1303 013006 004737 023526 JSR PC,DMPREG
1304 013012 012777 000007 011712 4$: MOV #DCFE,@ACR ;SELECT DCF
1305 013020 017737 011712 024750 MOV @DCF,DISPLY ;READ DCF
1306 013026 001403 BEQ 5$ ;SKIP IF 0
1307 013030 104044 ERROR+ 44 ;ST BOOT SET A DCF CONDITION
1308 013032 004737 023526 JSR PC,DMPREG
1309 013036 012777 000010 011666 5$: MOV #EXCE,@ACR ;SELECT EXC
1310 013044 017737 011666 024750 MOV @EXC,DISPLY ;READ EXC
1311 013052 001403 BEQ 6$ ;SKIP IF 0
1312 013054 104044 ERROR+ 44 ;ST BOOT SET AN EXC CONDITION
1313 013056 004737 023526 JSR PC,DMPREG
1314 013062 012777 000005 011642 6$: MOV #PGFE,@ACR ;SELECT PGF
1315 013070 017737 011642 024750 MOV @PGF,DISPLY ;READ PGF
1316 013076 001403 BEQ 7$ ;SKIP IF 0
1317 013100 104044 ERROR+ 44 ;ST BOOT SET A PGF CONDITION
1318 013102 004737 023526 JSR PC,DMPREG
1319 013106 012777 000006 011616 7$: MOV #STFE,@ACR ;SELECT STF
1320 013114 016437 022002 001124 MOV SBFTAB(4),$GDDAT ;FETCH FLAG BIT
1321 013122 013777 001124 011606 MOV $GDDAT,@STF ;CLEAR FLAG
1322 013130 005137 001124 COM $GDDAT ;FLIP GDDAT
1323 013134 042737 170377 001124 BIC #170377,$GDDAT ;MASK OUT JUNK
1324 013142 012777 000006 011562 MOV #STFE,@ACR ;SELECT STF
1325 013150 017737 011562 001126 MOV @STF,$BDDAT ;READ STF
1326 013156 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1327 013164 001403 BEQ 8$ ;SKIP IF OK
    
```

1328 013166 104044 ERROR+ 44 :WRITING A 1 TO STF FLAG DID NOT  
1329 013170 004737 023526 JSR PC,DMPREG :CLEAR IT  
1330  
1331 013174 8\$:

```

1333 ;ISSUE A PROGRAM INTERRUPT REQUEST WITH IMSK BIT =1 AND CHECK:
1334 ;PIF BIT TO STAY 0
1335 ;UI BIT TO SET
1336 ;UI BIT TO CLEAR WHEN WRITTEN TO A 1
1337

```

```

1338 ;*****
(3) ;*TEST 40 CHECK THE IMSK BIT FOR PROPER OPERATION
(3) ;*****

```

```

(2) 013174 000004
1339 013176 004737 023306 JSR PC,SETMNT ;ENABLE MAINT. LOOP
1340 013202 012777 000000 011522 MOV #PGTEE,@ACR ;SELECT PGTE
1341 013210 016477 021772 011520 MOV PGITAB(4),@PGTE ;LOAD PIB
1342 013216 012777 000004 011506 MOV #IMSKE,@ACR ;SELECT IMSK
1343 013224 016477 021772 011504 MOV IMITAB(4),@IMSK ;INHIBIT LINE
1344 013232 012777 000001 011472 MOV #PGCSE,@ACR ;SELECT PGCS
1345 013240 012777 000001 011470 MOV #GO,@PGCS ;TRANSMIT
1346 013246 012777 000001 011456 1$: MOV #PGCSE,@ACR ;SELECT PGCS
1347 013254 032777 004000 011454 BIT #PRDY,@PGCS ;CHECK FOR
1348 013262 001771 BEQ 1$ ;READY
1349 013264 012777 000005 011440 MOV #PGFE,@ACR ;SELECT PGF
1350 013272 017737 011440 001126 MOV @PGF,$BDDAT ;READ PGF
1351 013300 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAGS
1352 013306 046437 021772 001124 BIC PGITAB(4),$GDDAT ;MINUS EXPECTED FLAG
1353 013314 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1354 013322 001403 BEQ 2$ ;SKIP IF OK
1355 013324 104044 ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS PGF BIT
1356 013326 004737 023526 JSR PC,DMPREG
1357 013332 016437 022002 001124 2$: MOV UVITAB(4),$GDDAT ;LOAD EXPECTED UI BIT
1358 013340 012777 000010 011364 MOV #EXCE,@ACR ;SELECT EXC
1359 013346 017737 011364 001126 MOV @EXC,$BDDAT ;READ EXC
1360 013354 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK THE UI BIT
1361 013362 001403 BEQ 3$ ;SKIP IF SET
1362 013364 104044 ERROR+ 44 ;UI DID NOT SET ON A PIF WITH IMSK=1
1363 013366 004737 023526 JSR PC,DMPREG
1364 013372 005037 001124 3$: CLR $GDDAT ;EXPECT 0
1365 013376 012777 000010 011326 MOV #EXCE,@ACR ;SELECT EXC
1366 013404 016477 022002 011324 MOV UVITAB(4),@EXC ;CLEAR UI BIT
1367 013412 012777 000010 011312 MOV #EXCE,@ACR ;SELECT EXC
1368 013420 017737 011312 024750 MOV @EXC,DISPLY ;READ EXC
1369 013426 013737 024750 001126 MOV DISPLY,$BDDAT ;TEST EXC
1370 013434 001403 BEQ 4$ ;SKIP IF 0
1371 013436 104044 ERROR+ 44 ;WRITING A 1 TO UI BIT DID NOT CLEAR IT
1372 013440 004737 023526 JSR PC,DMPREG
1373 013444 012777 000006 011260 4$: MOV #STFE,@ACR ;SELECT STF REGISTER
1374 013452 017737 011260 001126 MOV @STF,$BDDAT ;GET AND TEST STF REGISTER
1375 013460 001403 BEQ 5$ ;STF SHOULD BE 0
1376 013462 104044 ERROR+ 44 ;STF NOT ZERO ON A PGI WITH IMSK=1
1377 013464 004737 023526 JSR PC,DMPREG
1378 013470 5$:

```

1380 :ISSUE A PROGRAM BOOT REQUEST WITH BIMSK BIT =1 AND CHECK:  
 1381 :PBF BIT TO STAY 0  
 1382 :UI BIT TO SET  
 1383

1384 :\*\*\*\*\*  
 (3) :\*TEST 41 CHECK THE IMSK BIT FOR PROPER OPERATION ON PGM BOOT  
 (3) :\*\*\*\*\*

```

(2) 013470 000004
1385 013472 004737 023306      JSR    PC,SETMNT      ;ENABLE MAINT. LOOP
1386 013476 012777 000000 011226  MOV    #PGTEE,@ACR    ;SELECT PGTE
1387 013504 016477 022002 011224  MOV    PGBTAB(4),@PGTE ;LOAD PBB
1388 013512 012777 000004 011212  MOV    #IMSKE,@ACR    ;SELECT IMSK
1389 013520 016477 022002 011210  MOV    BMITAB(4),@IMSK ;INHIBIT LINE
1390 013526 012777 000001 011176  MOV    #PGCSE,@ACR    ;SELECT PGCS
1391 013534 012777 000001 011174  MOV    #GO,@PGCS      ;TRANSMIT
1392 013542 012777 000001 011162 1$:  MOV    #PGCSE,@ACR    ;SELECT PGCS
1393 013550 032777 004000 011160  BIT    #PRDY,@PGCS    ;CHECK FOR
1394 013556 001771          BEQ    1$             ;READY
1395 013560 012777 000005 011144  MOV    #PGFE,@ACR     ;SELECT PGF
1396 013566 017737 011144 001126  MOV    @PGF,$BDDAT    ;READ PGF
1397 013574 012737 007400 001124  MOV    #7400,$GDDAT   ;FETCH EXPECTED FLAGS
1398 013602 046437 022002 001124  BIC    PGBTAB(4),$GDDAT ;MINUS EXPECTED FLAG
1399 013610 023737 001124 001126  CMP    $GDDAT,$BDDAT  ;CHECK FLAG FOR 0
1400 013616 001403          BEQ    2$             ;SKIP IF OK
1401 013620 104044          ERROR+ 44           ;IMSK BIT DID NOT INHIBIT ITS PGF BIT
1402 013622 004737 023526      JSR    PC,DMPREG
1403 013626 016437 022002 001124 2$:  MOV    UVITAB(4),$GDDAT ;LOAD EXPECTED UI BIT
1404 013634 012777 000010 011070  MOV    #EXCE,@ACR     ;SELECT EXC
1405 013642 017737 011070 001126  MOV    @EXC,$BDDAT    ;READ EXC
1406 013650 023737 001124 001126  CMP    $GDDAT,$BDDAT  ;CHECK THE UI BIT
1407 013656 001403          BEQ    3$             ;SKIP IF SET
1408 013660 104044          ERROR+ 44           ;UI DID NOT SET ON A PIF WITH IMSK=1
1409 013662 004737 023526      JSR    PC,DMPREG
1410 013666 005037 001124          CLR    $GDDAT         ;EXPECT STF TO BE 0
1411 013672 012777 000006 011032 3$:  MOV    #STFE,@ACR     ;SELECT STF
1412 013700 017737 011032 001126  MOV    @STF,$BDDAT    ;GET AND TEST STF
1413 013706 001403          BEQ    4$             ;IT SHOULD BE 0
1414 013710 104044          ERROR+ 44
1415 013712 004737 023526      JSR    PC,DMPREG
1416 013716          4$:
  
```

```

1418 :ISSUE A SANITY TIMER INTERRUPT REQUEST WITH IMSK BIT =1 AND CHECK:
1419 :SIF BIT TO STAY 0
1420 :UI BIT TO SET
1421 :UI BIT TO CLEAR WHEN WRITTEN TO A 1
1422
1423

```

```

(3) :*****
(3) :*TEST 42 CHECK THE IMSK BIT FOR PROPER OPERATION ON ST INTR
(2) :*****

```

```

(2) 013716 000004
1424 013720 004737 023476 JSR PC,SSTMNT ;ENABLE MAINT. LOOP
1425 013724 012777 000000 011000 MOV #PGTEE,@ACR ;SELECT PGTE
1426 013732 016477 021772 010776 MOV PGITAB(4),@PGTE ;LOAD PIB
1427 013740 012777 000004 010764 MOV #IMSKE,@ACR ;SELECT IMSK
1428 013746 016477 021772 010762 MOV IMITAB(4),@IMSK;INHIBIT LINE
1429 013754 012777 000001 010750 MOV #PGCSE,@ACR ;SELECT PGCS
1430 013762 012777 000003 010746 MOV #PTP+GO,@PGCS ;TRANSMIT
1431 013770 012777 000001 010734 1$: MOV #PGCSE,@ACR ;SELECT PGCS
1432 013776 032777 004000 010732 BIT #PRDY,@PGCS ;CHECK FOR
1433 014004 001771 BEQ 1$ ;READY
1434 014006 012777 000006 010716 MOV #STFE,@ACR ;SELECT STF
1435 014014 017737 010716 001126 MOV @STF,$BDDAT ;READ STF
1436 014022 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAGS
1437 014030 046437 021772 001124 BIC PGITAB(4),$GDDAT;MINUS EXPECTED FLAG
1438 014036 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1439 014044 001403 BEQ 2$ ;SKIP IF OK
1440 014046 104044 ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS SIF BIT
1441 014050 004737 023526 JSR PC,DMPREG
1442 014054 016437 022002 001124 2$: MOV UVITAB(4),$GDDAT;LOAD EXPECTED UI BIT
1443 014062 012777 000010 010642 MOV #EXCE,@ACR ;SELECT EXC
1444 014070 017737 010642 001126 MOV @EXC,$BDDAT ;READ EXC
1445 014076 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK THE UI BIT
1446 014104 001403 BEQ 3$ ;SKIP IF SET
1447 014106 104044 ERROR+ 44 ;UI DID NOT SET ON A SIF WITH IMSK=1
1448 014110 004737 023526 JSR PC,DMPREG
1449 014114 005037 001124 3$: CLR $GDDAT ;EXPECT 0
1450 014120 012777 000010 010604 MOV #EXCE,@ACR ;SELECT EXC
1451 014126 016477 022002 010602 MOV UVITAB(4),@EXC ;CLEAR UI BIT
1452 014134 012777 000010 010570 MOV #EXCE,@ACR ;SELECT EXC
1453 014142 017737 010570 024750 MOV @EXC,DISPLY ;READ EXC
1454 014150 013737 024750 001126 MOV DISPLY,$BDDAT ;TEST EXC
1455 014156 001403 BEQ 4$ ;SKIP IF 0
1456 014160 104044 ERROR+ 44 ;WRITING A 1 TO UI BIT DID NOT CLEAR IT
1457 014162 004737 023526 JSR PC,DMPREG
1458 014166 012777 000005 010536 4$: MOV #PGFE,@ACR ;SELECT PGF REGISTER
1459 014174 017737 010536 001126 MOV @PGF,$BDDAT ;GET AND TEST STF REGISTER
1460 014202 001403 BEQ 5$ ;PGF SHOULD BE 0
1461 014204 104044 ERROR+ 44 ;PGF NOT ZERO ON A STI WITH IMSK=1
1462 014206 004737 023526 JSR PC,DMPREG
1463 014212 5$:

```

1465 ;ISSUE A SANITY TIMER BOOT REQUEST WITH BIMSK BIT =1 AND CHECK:  
1466 ;SBF BIT TO STAY 0  
1467 ;UI BIT TO SET  
1468  
1469

(3) :\*\*\*\*\*  
(3) :\*TEST 43 CHECK THE IMSK BIT FOR PROPER OPERATION ON ST BOOT  
(2) :\*\*\*\*\*

```

TST43: SCOPE
1470 JSR PC,SSTMNT ;ENABLE MAINT. LOOP
1471 MOV #PGTEE,@ACR ;SELECT PGTE
1472 MOV PGBTAB(4),@PGTE ;LOAD PBB
1473 MOV #IMSKE,@ACR ;SELECT IMSK
1474 MOV BMITAB(4),@IMSK ;INHIBIT LINE
1475 MOV #PGCSE,@ACR ;SELECT PGCS
1476 MOV #PTP+GO,@PGCS ;TRANSMIT
1477 1$: MOV #PGCSE,@ACR ;SELECT PGCS
1478 BIT #PRDY,@PGCS ;CHECK FOR
1479 BEQ 1$ ;READY
1480 MOV #STFE,@ACR ;SELECT STF
1481 MOV @STF,$BDDAT ;READ STF
1482 MOV #7400,$GDDAT ;FETCH EXPECTED FLAGS
1483 BIC PGBTAB(4),$GDDAT ;MINUS EXPECTED FLAG
1484 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1485 BEQ 2$ ;SKIP IF OK
1486 ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS SBF BIT
1487 JSR PC,DMPREG
1488 2$: MOV UVITAB(4),$GDDAT ;LOAD EXPECTED UI BIT
1489 MOV #EXCE,@ACR ;SELECT EXC
1490 MOV @EXC,$BDDAT ;READ EXC
1491 CMP $GDDAT,$BDDAT ;CHECK THE UI BIT
1492 BEQ 3$ ;SKIP IF SET
1493 ERROR+ 44 ;UI DID NOT SET ON A SBF WITH IMSK-1
1494 JSR PC,DMPREG
1495 3$: CLR $GDDAT ;EXPECT PGF TO BE 0
1496 MOV #PGFE,@ACR ;SELECT PGF
1497 MOV @PGF,$BDDAT ;GET AND TEST PGF
1498 BEQ 4$ ;IT SHOULD BE 0
1499 ERROR+ 44
1500 JSR PC,DMPREG
1501 4$:

```

```

1503 ;TEST OPERATION OF THE TIMER LOGIC
1504
1505 *****
(3) ;*TEST 44 CHECK THE TIMER INTERRUPT LOGIC AND COUNT REGISTER
(3) *****
(2) 014440 000004 15T44: SCOPE
1506 014442 004737 023476 JSR PC,SSTMNT ;SET MAINT. LOOPBACK
1507 014446 005000 CLR RO ;INIT. TIMER COUNTER
1508 014450 012777 000002 010254 MOV #STEE,@ACR ;SELECT SITE
1509 014456 016477 021772 010252 MOV SIFTAB(4),@STTE ;ENABLE SI LINE
1510 014464 012777 000001 010244 MOV #ENB,@STCS ;ENABLE SANITY TIMER
1511 014472 012777 000003 010232 1$: MOV #STCSE,@ACR ;SELECT STCS
1512 014500 017737 010232 024750 MOV @STCS,DISPLY ;READ STCS
1513 014506 032737 177410 024750 BIT #177410,DISPLY ;CHECK FOR TMO OR COUNT=-1
1514 014514 001007 BNE 2$ ;SKIP IF ANY BITS SET
1515 014516 005200 INC RO ;COUNT BACKGROUND TIMER
1516 014520 001364 BNE 1$ ;KEEP COUNTING UNTIL OVERFLOW
1517 014522 104044 ERROR+ 44 ;TIMER DID NOT COUNTDOWN WHEN ENABLED
1518 014524 004737 023526 JSR PC,DMPREG
1519 014530 012700 177777 MOV #-1,RO ;SET TIME TO MAX
1520 014534 010037 024752 2$: MOV RO,TIMPER ;SAVE TIME PERIOD CONSTANT
1521 014540 006200 ASR RO ;DIVIDE BY
1522 014542 006200 ASR RO ;8
1523 014544 006200 ASR RO
1524 014546 060037 024752 ADD RO,TIMPER ;ADD 12%
1525 014552 012777 000003 010152 MOV #STCSE,@ACR ;**
1526 014560 017737 010152 024750 MOV @STCS,DISPLY ;**
1527 014566 005037 001126 CLR $BDDAT ;INIT. DISPLAY
1528 014572 113737 024751 001126 MOVB DISPLY+1,$BDDAT ;FETCH COUNT
1529 014600 012737 000377 001124 MOV #377,$GDDAT ;LOAD EXPECTED
1530 014606 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR TIMER = -1
1531 014614 001403 BEQ 3$ ;SKIP IF OK
1532 014616 104044 ERROR+ 44 ;TIMER DID NOT COUNT FROM 0 TO -1
1533 014620 004737 023526 JSR PC,DMPREG
1534 014624 012737 000011 001124 3$: MOV #TMO+ENB,$GDDAT ;LOAD EXPECTED
1535 014632 113737 024750 001126 MOVB DISPLY,$BDDAT ;LOAD ACTUAL
1536 014640 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR TMO
1537 014646 001403 BEQ 4$ ;SKIP IF SET
1538 014650 104044 ERROR+ 44 ;TMO DID NOT SET ON TIMER UNDERFLOW
1539 014652 004737 023526 JSR PC,DMPREG
1540 014656 012777 000001 010046 4$: MOV #PGCSE,@ACR ;SELECT PGCS
1541 014664 017737 010046 024750 MOV @PGCS,DISPLY ;READ PGCS
1542 014672 032737 000010 024750 BIT #IP,DISPLY ;TEST FOR IP
1543 014700 001003 BNE 5$ ;SKIP IF SET
1544 014702 104044 ERROR+ 44 ;SIF DID NOT SET IP
1545 014704 004737 023526 JSR PC,DMPREG
1546 014710 5$:

```

```

1548 (3)
1549 (3)
1550 (2) 014710 000004
1549 014712 004737 023476
1550 014716 012777 000002 010006
1551 014724 016477 022002 010004
1552 014732 012777 000001 007776
1553 014740 005000
1554 014742 012777 000003 007762 1$:
1555 014750 017737 007762 024750
1556 014756 032737 177410 024750
1557 014764 001005
1558 014766 005200
1559 014770 001364
1560 014772 104044
1561 014774 004737 023526
1562 015000 012777 000003 007724 2$:
1563 015006 017737 007724 024750
1564 015014 005037 001126
1565 015020 113737 024751 001126
1566 015026 012737 000377 001124
1567 015034 023737 001124 001126
1568 015042 001403
1569 015044 104044
1570 015046 004737 023526
1571 015052 012737 000011 001124 3$:
1572 015060 113737 024750 001126
1573 015066 023737 001124 001126
1574 015074 001403
1575 015076 104044
1576 015100 004737 023526
1577 015104 012777 000001 007620 4$:
1578 015112 017737 007620 024750
1579 015120 032737 000010 024750
1580 015126 001403
1581 015130 104044
1582 015132 004737 023526
1583 015136 5$:

```

```

*****
*TEST 45 CHECK THE TIMER BOOT LOGIC AND COUNT REGISTER
*****
TST45: SCOPE
JSR PC,SSTMNT ;SET MAINT. LOOPBACK
MOV #SITEE,@ACR ;SELECT SITE
MOV SBFTAB(4),@STTE ;ENABLE SB LINE
MOV #ENB,@STCS ;ENABLE SANITY TIMER
CLR RO ;CLR COUNTER
MOV #STCSE,@ACR ;SELECT STCS
MOV @STCS,DISPLY ;READ STCS
BIT #177410,DISPLY ;CHECK FOR TMO OR COUNT=-1
BNE 2$ ;SKIP IF ANY BITS SET
INC RO ;COUNT BACKGROUND TIMER
BNE 1$ ;KEEP COUNTING UNTIL OVERFLOW
ERROR+ 44 ;TIMER DID NOT COUNTDOWN WHEN ENABLED
JSR PC,DMPREG
MOV #STCSE,@ACR ;**
MOV @STCS,DISPLY ;**
CLR $BDDAT ;INIT. DISPLAY
MOVB DISPLY+1,$BDDAT ;FETCH COUNT
MOV #377,$GDDAT ;LOAD EXPECTED
CMP $GDDAT,$BDDAT ;CHECK FOR TIMER = -1
BEQ 3$ ;SKIP IF OK
ERROR+ 44 ;TIMER DID NOT COUNT FROM 0 TO -1
JSR PC,DMPREG
MOV #TMO+ENB,$GDDAT ;LOAD EXPECTED
MOVB DISPLY,$BDDAT ;LOAD ACTUAL
CMP $GDDAT,$BDDAT ;CHECK FOR TMO
BEQ 4$ ;SKIP IF SET
ERROR+ 44 ;TMO DID NOT SET ON TIMER UNDERFLOW
JSR PC,DMPREG
MOV #PGCSE,@ACR ;SELECT PGCS
MOV @PGCS,DISPLY ;READ PGCS
BIT #IP,DISPLY ;TEST FOR IP
BEQ 5$ ;SKIP IF CLR
ERROR+ 44 ;SBF SET IP
JSR PC,DMPREG

```



```
1585      ;:*****  
(3)      ;*TEST 46      CHECK THE COUNTER FOR PROPER DECREMENTATION  
(3)      ;:*****  
(2) 015136 000004 TST46: SCOPE  
1586 015140 012737 000001 001204      MOV      #1,$TMP3      ;INIT THE COUNT VALUE  
1587 015146 005037 001126      CLR      $BDDAT      ;ZERO THE HIGH BYTE OF $BDDAT  
1588 015152 012737 015160 001110      MOV      #1$, $LPERR      ;SET SCOPE POINT  
1589 015160 004737 023476      JSR      PC,$STMNT      ;ENTER MAINT. LOOP MODE  
1590 015164 012777 000002 007540      MOV      #STTEE,@ACR      ;SELECT STTE  
1591 015172 016477 021772 007536      MOV      SIFTAB(4),@STTE      ;SET LINE ENABLE  
1592 015200 112777 000001 007530      MOV      #ENB,@STCS      ;ENABLE COUNTING  
1593 015206 005377 007520      DEC      @ACR      ;SELECT STCS  
1594 015212 113777 001204 007520      MOV      $TMP3,@STCSH      ;LOAD COUNT AND START  
1595 015220 013737 001204 001124      MOV      $TMP3,$GDDAT      ;GET STARTING COUNT VALUE  
1596 015226 005337 001124      DEC      $GDDAT      ;MINUS 1  
1597 015232 013700 024752      MOV      TIMPER,RO      ;GET BASIC TIME CONSTANT  
1598 015236 006300      ASL      RO      ;*2  
1599 015240 005377 007466      DEC      @ACR      ;SELECT STCS  
1600 015244 117737 007470 001126      MOV      @STCSH,$BDDAT      ;READ COUNT VALUE  
1601 015252 123737 001204 001126      CMP      $TMP3,$BDDAT      ;CHECK FOR CHANGE  
1602 015260 001006      BNE      3$      ;SKIP IF CHANGED  
1603 015262 005300      DEC      RO      ;DEFLATE COUNTER  
1604 015264 001365      BNE      2$      ;CHECK AGAIN  
1605 015266 104044      ERROR+ 44      ;TIMER DID NOT DECREMENT  
1606 015270 004737 023526      JSR      PC,DMPREG  
1607 015274 000414      BR      4$      ;SKIP NEXT CHECK  
1608 015276 005377 007430      DEC      @ACR      ;SELECT STCS  
1609 015302 117737 007432 001126      MOV      @STCSH,$BDDAT      ;REREAD TIMFR  
1610 015310 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;CHECK FOR CORRECT DECREMENTATION  
1611 015316 001403      BEQ      4$      ;SKIP IF OK  
1612 015320 104044      ERROR+ 44      ;COUNT REGISTER DID NOT DECREMENT CORRECTLY  
1613 015322 004737 023526      JSR      PC,DMPREG  
1614 015326 105237 001204      INC      $TMP3      ;BUMP TEST VALUE  
1615 015332 001312      BNE      1$      ;DO 1 - 377
```

```

1617 ;ISSUE A PROGRAM-GENERATED REQUEST WITH INTERRUPT MASKS CLEAR WITH ALL COMBINATIONS OF
1618 ;TRANSMIT ENABLES, MAINTENANCE TYPE, AND MAINTENANCE FRAMING
1619 ;AND CHECK FOR:
1620 ;   CORRECT CONTENTS OF PGCS (RDY, IP, SELF-ID)
1621 ;   RTE BITS IN THE EXC REGISTER TO BE SET
1622 ;   ALL OTHER FLAGS TO BE CLEAR (PGF, STF, DCF, UIF)
1623 ;   SELECTED RTE FLAG CAN BE CLEARED

```

```

1624
1625     000005           W=5
1626
1627

```

```

:*****
:*TEST 47     CHECK OPERATION OF PARITY & FRAMING ERRORS, IM(3:0)=0
:*****

```

```

(2) 015334 000004
1628 015336 004737 023306      JSR     PC,SETMNT      ;ENABLE MAINTENANCE LOOPBACK
1629 015342 012737 015366 001110  MOV     #1,$LPERR     ;FIX START OF LOOP
1630 015350 012777 000007 007354  MOV     #DCF,@ACR     ;SAVE IMAGE OF DCF REGISTER
1631 015356 017737 007354 024756  MOV     @DCF,$NAPO    ;INIT FRAME, TYPE, PGTE DATA SETUP
1632 015364 005000      CLR     RO
1633
1634 015366 010037 001200      1$:   MOV     RO,$TMP1
1635 015372 004737 023506      JSR     PC,PARCAL     ;CALCULATE PARITY OF DATA
1636 015376 113737 001200 001201  MOVB   $TMP1,$TMP1+1
1637 015404 106037 001201      RORB   $TMP1+1       ;POSITION DATA
1638 015410 106037 001201      RORB   $TMP1+1
1639 015414 106037 001201      RORB   $TMP1+1
1640 015420 106037 001201      RORB   $TMP1+1
1641 015424 012777 000000 007300  MOV     #PGTEE,@ACR
1642 015432 013777 001200 007276  MOV     $TMP1,@PGTE  ;LOAD TRANSMIT ENABLES
1643 015440 006301      ASL     R1           ;GET PARITY
1644 015442 042701 177775      2$:   BIC     #177775,R1
1645 015446 016437 021762 001124  MOV     $IDTAB(4),$GDDAT ;GET SELF-ID
1646 015454 050137 001124      BIS    R1,$GDDAT
1647 015460 052701 000001      BIS    #1,R1
1648 015464 012777 000001 007240  MOV     #PGCSE,@ACR
1649 015472 010177 007240      MOV     R1,@PGCS    ;LOAD PARITY, SET GO
1650 015476 005001      CLR     R1
1651 015500 052737 004010 001124  BIS    #PRDY+IP,$GDDAT
1652 015506 105377 007220      3$:   DECB   @ACR
1653 015512 017737 007220 001126  MOV     @PGCS,$BDDAT
1654 015520 032737 004000 001126  BIT    #PRDY,$BDDAT ;WAIT FOR READY
1655 015526 001005      BNE    4$
1656 015530 005301      DEC    R1           ;BUT DON'T WAIT TOO LONG
1657 015532 001365      BNE    3$
1658 015534 104044      ERROR+ 44
1659 015536 004737 023526      JSR     PC,DMPREG
1660 015542 023737 001126 001124  4$:   CMP     $BDDAT,$GDDAT ;IS PGCS OK?
1661 015550 001403      BEQ    5$
1662 015552 104044      ERROR+ 44
1663 015554 004737 023526      JSR     PC,DMPREG

```

```

1665 015560 012737 000017 001124 5$:  MOV    #17,$GDDAT      ;GET EXPECTED RTE FLAGS
1666 015566 012777 000G10 007136      MOV    #EXCE,@ACR
1667 015574 017737 007136 001126      MOV    @EXC,$BDDAT    ;GET EXC REGISTER
1668 015602 023737 001124 001126      CMP    $GDDAT,$BDDAT
1669 015610 001403                BEQ    6$
1670 015612 104044                ERROR+ 44
1671 015614 004737 023526                JSR    PC,DMPREG
1672 015620 105377 007106      6$:  DECB   @ACR
1673 015624 016437 021772 001124      MOV    PIFTAB(4),$GDDAT
1674 015632 013777 001124 007076      MOV    $GDDAT,@EXC    ;CLEAR FLAG
1675 015640 005137 001124                COM    $GDDAT        ;MAKE FLAG DATA
1676 015644 042737 177760 001124      BIC    #177760,$GDDAT
1677 015652 105377 007054                DECB   @ACR
1678 015656 017737 007054 001126      MOV    @EXC,$BDDAT
1679 015664 023737 001126 001124      CMP    $BDDAT,$GDDAT
1680 015672 001403                BEQ    7$
1681 015674 104044                ERROR+ 44
1682 015676 004737 023526                JSR    PC,DMPREG
1683 015702 005037 001124      7$:  CLR    $GDDAT
1684 015706 012777 000005 007016      MOV    #PGFE,@ACR
1685 015714 017737 007016 001126      MOV    @PGF,$BDDAT    ;PGF SHOULD BE 0
1686 015722 001403                BEQ    8$
1687 015724 104044                ERROR+ 44
1688 015726 004737 023526                JSR    PC,DMPREG
1689 015732 017737 007000 001126 8$:  MOV    @STF,$BDDAT    ;GET STF
1690 015740 001403                BEQ    9$            ;STF SHOULD BE 0
1691 015742 104044                ERROR+ 44
1692 015744 004737 023526                JSR    PC,DMPREG
1693 015750 013737 024756 001124 9$:  MOV    SNAPO,$GDDAT    ;GET EXPECTED DATA FOR DCF
1694 015756 017737 006754 001126      MOV    @DCF,$BDDAT    ;GET DCF REGISTER
1695 015764 023737 001126 001124      CMP    $BDDAT,$GDDAT ;CHECK THAT THERE IS NO CHANGE
1696 015772 001403                BEQ    10$
1697 015774 104044                ERROR+ 44
1698 015776 004737 023526                JSR    PC,DMPREG
1699 016002 005200      10$: INC    R0            ;INCREMENT TEST DATA
1700 016004 012777 000015 006720      MOV    #MTCE,@ACR    ;POINT TO MTC
1701 016012 042777 006000 006716      BIC    #MTYP+MFRM,@MTC ;CLEAR TYPE AND FRAME ERROR
1702 016020 032700 000400                BIT    #400,R0
1703 016024 001403                BEQ    11$          ;IF R0(8)=1, SET MTYPE
1704 016026 052777 004000 006702      BIS    #MTYP,@MTC
1705 016034 032700 001000      11$: BIT    #1000,R0
1706 016040 001403                BEQ    12$
1707 016042 052777 002000 006666      BIS    #MFRM,@MTC    ;IF R0(9)=1, SET FRAMING ERROR
1708 016050 020027 002000      12$: CMP    R0,#2000
1709 016054 103002                BHS    13$          ;LOOP UNTIL ALL COMBOS OF DATA, TYPE &
1710                                ;FRAMING HAVE BEEN TESTED.
1711 016056 000137 015366                JMP    1$
1712 016062      13$:

```

1714 ;ISSUE A PROGRAM-GENERATED REQUEST WITH INTERRUPT MASKS SET WITH ALL COMBINATIONS OF  
1715 ;TRANSMIT ENABLES, MAINTENANCE TYPE, AND MAINTENANCE FRAMING  
1716 ;AND CHECK FOR:  
1717 : CORRECT CONTENTS OF PGCS (RDY, SELF-ID)  
1718 : RTE BITS IN THE EXC REGISTER TO BE CLEAR  
1719 : ALL OTHER FLAGS TO BE CLEAR (PGF, STF, DCF, UIF)  
1720 ;

1721 ;\*\*\*\*\*  
(3) ;\*TEST 50 CHECK OPERATION OF PARITY & FRAMING ERRORS, IM(3:0)=1  
(3) ;\*\*\*\*\*

(2) 016062 000004  
1722 016064 004737 023306 JSR PC,SETMNT ;ENABLE MAINTENANCE LOOPBACK  
1723 016070 012737 016130 001110 MOV #1\$, \$LPERR ;FIX START OF LOOP  
1724 016076 012777 000007 006626 MOV #DCFE,@ACR  
1725 016104 017737 006626 024756 MOV @DCF,\$NAPO ;SAVE IMAGE OF DCF REGISTER  
1726 016112 012777 000004 006612 MOV #IMSKE,@ACR ;POINT TO IMASK  
1727 016120 012777 000017 006610 MOV #17,@IMSK ;SET ALL INTERRUPT MASKS  
1728 016126 005000 CLR R0 ;INIT FRAME, TYPE, PGTE DATA SETUP  
1729  
1730 016130 010037 001200 1\$: MOV R0,\$TMP1  
1731 016134 004737 023506 JSR PC,PARCAL ;CALCULATE PARITY OF DATA  
1732 016140 113737 001200 001201 MOVB \$TMP1,\$TMP1+1  
1733 016146 106037 001201 RORB \$TMP1+1 ;POSITION DATA  
1734 016152 106037 001201 RORB \$TMP1+1  
1735 016156 106037 001201 RORB \$TMP1+1  
1736 016162 106037 001201 RORB \$TMP1+1  
1737 016166 012777 000000 006536 MOV #PGTEE,@ACR  
1738 016174 013777 001200 006534 MOV \$TMP1,@PGTE ;LOAD TRANSMIT ENABLES  
1739 016202 006301 ASL R1 ;GET PARITY  
1740 016204 042701 177775 2\$: BIC #177775,R1  
1741 016210 016437 021762 001124 MOV \$IDTAB(4),\$GDDAT ;GET SELF-ID  
1742 016216 050137 001124 BIS R1,\$GDDAT  
1743 016222 052701 000001 BIS #1,R1  
1744 016226 012777 000001 006476 MOV #PGCSE,@ACR  
1745 016234 010177 006476 MOV R1,@PGCS ;LOAD PARITY, SET GO  
1746 016240 005001 CLR R1  
1747 016242 052737 004000 001124 BIS #PRDY,\$GDDAT  
1748 016250 105377 006456 3\$: DECB @ACR  
1749 016254 017737 006456 001126 MOV @PGCS,\$BDDAT  
1750 016262 032737 004000 001126 BIT #PRDY,\$BDDAT ;WAIT FOR READY  
1751 016270 001005 BNE 4\$  
1752 016272 005301 DEC R1 ;BUT DON'T WAIT TOO LONG  
1753 016274 001365 BNE 3\$  
1754 016276 104044 ERROR+ 44  
1755 016300 004737 023526 JSR PC,DMPREG  
1756 016304 023737 001126 001124 4\$: CMP \$BDDAT,\$GDDAT ;IS PGCS OK?  
1757 016312 001403 BEQ 5\$  
1758 016314 104044 ERROR+ 44  
1759 016316 004737 023526 JSR PC,DMPREG  
1760 016322 012737 000000 001124 5\$: MOV #0,\$GDDAT ;GET EXPECTED RTE FLAGS  
1761 016330 012777 000010 006374 MOV #EXCE,@ACR  
1762 016336 017737 006374 001126 MOV @EXC,\$BDDAT ;GET EXC REGISTER  
1763 016344 001403 BEQ 6\$  
1764 016346 104044 ERROR+ 44  
1765 016350 004737 023526 JSR PC,DMPREG  
1766 016354 6\$:

```

1767 016354 005037 001124 7$: CLR $GDDAT
1768 016360 012777 000005 006344 MOV #PGFE,@ACR
1769 016366 017737 006344 001126 MOV @PGF,$BDDAT ;PGF SHOULD BE 0
1770 016374 001403 BEQ 8$
1771 016376 104044 ERROR+ 44
1772 016400 004737 023526 JSR PC,DMPREG
1773 016404 017737 006326 001126 8$: MOV @STF,$BDDAT ;GET STF
1774 016412 001403 BEQ 9$ ;STF SHOULD BE 0
1775 016414 104044 ERROR+ 44
1776 016416 004737 023526 JSP PC,DMPREG
1777 016422 013737 024756 001124 9$: MOV SNAPO,$GDDAT ;GET EXPECTED DATA FOR DCF
1778 016430 017737 006302 001126 MOV @DCF,$BDDAT ;GET DCF REGISTER
1779 016436 023737 001126 001124 CMP $BDDAT,$GDDAT ;CHECK THAT THERE IS NO CHANGE
1780 016444 001403 BEQ 10$
1781 016446 104044 ERROR+ 44
1782 016450 004737 023526 JSR PC,DMPREG
1783 016454 005200 10$: INC R0 ;INCREMENT TEST DATA
1784 016456 012777 000015 006246 MOV #MTCE,@ACR ;POINT TO MTC
1785 016464 042777 006000 006244 BIC #MTYP+MFRM,@MTC ;CLEAR TYPE AND FRAME ERROR
1786 016472 032700 000400 BIT #400,R0
1787 016476 001403 BEQ 11$ ;IF R0(8)=!, SET MTYPE
1788 016500 052777 004000 006230 BIS #MTYP,@MTC -
1789 016506 032700 001000 11$: BIT #1000,R0
1790 016512 001403 BEQ 12$
1791 016514 052777 002000 006214 BIS #MFRM,@MTC ;IF R0(9)-1, SET FRAMING ERROR
1792 016522 020027 002000 12$: CMP R0,#2000
1793 016526 103600 BLO 1$ ;LOOP UNTIL ALL COMBOS OF DATA, TYPE &
1794 FRAMING HAVE BEEN TESTED.
  
```

```

1796
1797
1798
(3)
(3)
(2) 016530 000004
1799 016532 004737 023306
1800 016536 013701 024732
1801 016542 013702 024736
1802 016546 012711 000000
1803 016552 016412 021772
1804 016556 012712 000001
1805 016562 005311
1806 016564 005212
1807 016566 016437 021762 001124
1808 016574 062737 144010 001124
1809 016602 012711 000001
1810 016606 011237 001126
1811 016612 032737 004000 001126
1812 016620 001770
1813 016622 023737 001124 001126
1814 016630 001403
1815 016632 104044
1816 016634 004737 023526
1817
1818 016640 012711 000005
1819 016644 012712 000017
1820 016650 017746 006070
1821 016654 017746 006062
1822 016660 012777 016730 006054
1823 016666 012777 000340 006050
1824 016674 012711 000001
1825 016700 012712 000004
1826 016704 005037 177776
1827 016710 000240
1828 016712 012737 000340 177776
1829 016720 104044
1830 016722 004737 023526
1831 016726 000401
1832 016730 022626
1833 016732 012677 006004
1834 016736 012677 006002
1835 016742 005311
1836 016744 012712 100000
1837 016750 042737 140010 001124
1838 016756 005311
1839 016760 011237 001126
1840 016764 023737 001124 001126
1841 016772 001403
1842 016774 104044
1843 016776 004737 023526
1844
1845 017002

```

```

;TEST ALL THE PGCS ERROR BITS FOR PROPER OPERATION
:*****
:*TEST 51 TEST GRJ FOR PROPER OPERATION
:*****
TST51: SCOPE
        JSR      PC,SETMNT      ;CLR IIST AND ENTER MAINT. MODE
        MOV      ACR,R1        ;R1=ACR
        MOV      ADR,R2        ;R2=ADR
        MOV      #PGTEE,@R1    ;SELECT PGTE
        MOV      PGITAB(4),@R2 ;LOAD PGTE WITH ENABLE
        MOV      #GO,@R2      ;TRANSMIT
        DEC      @R1           ;SELECT PGCS
        INC      @R2           ;SET GO AGAIN
        MOV      SIDTAB(4),%GDDAT;LOAD EXPECTED
        ADD      #ERR+GRJ+PRDY+IP,%GDDAT
1$:     MOV      #PGCSE,@R1    ;SELECT PGCS
        MOV      @R2,%BDDAT    ;READ PGCS
        BIT      #PRDY,%BDDAT  ;TEST FOR RDY
        BEQ      1$           ;LOOP UNTIL SET
        CMP      %GDDAT,%BDDAT ;CHECK STATUS
        BEQ      2$           ;SKIP IF OK
        ERROR+  44           ;ERR,GRJ, OR IP DID NOT SET ON
        JSR      PC,DMPREG
2$:     MOV      #PGFE,@R1    ;A GRJ ERROR
        MOV      #17,@R2      ;SELECT PGF
        MOV      @111P,-(SP)   ;CLEAR PGF
        MOV      @111V,-(SP)   ;SAVE OLD
        MOV      #3$,@111V    ;PI DATA
        MOV      #340,@111P    ;LOAD NEW
        MOV      #PGCSE,@R1    ;PI DATA
        MOV      #IE,@R2      ;SELECT PGCS
        CLR      PS           ;ENABLE PI
        NOP                    ;ALLOW PI
        MOV      #340,PS      ;STALL
        ERROR+  44           ;LOCKOUT PI
        JSR      PC,DMPREG    ;NO PI ON ERR WITH IE-1
        BR      4$
3$:     POPPOP
4$:     MOV      (SP)+,@111V   ;SKIP
        MOV      (SP)+,@111P   ;FIX STACK
        DEC      @R1          ;RESTORE OLD
        MOV      #ERR,@R2     ;PI INFO
        MOV      #ERR+GRJ+IP,%GDDAT ;SELECT PGCS
        DEC      @R1          ;CLEAR ERR BY WRITING 1 TO IT
        MOV      @R2,%BDDAT   ;SELECT PGCS
        BIC      #ERR+GRJ+IP,%GDDAT ;READ PGCS
        DEC      @R1          ;CHECK STATUS
        MOV      @R2,%BDDAT   ;SKIP IF OK
        CMP      %GDDAT,%BDDAT ;WRITING A ONE TO ERR DID NOT
        BEQ      5$           ;CLEAR ERR,GRJ, AND ,P
        ERROR+  44
        JSR      PC,DMPREG
5$:

```

```

1847
(3)
(3)
(2) 017002 000004
1848 017004 004737 023306
1849 017010 013701 024732
1850 017014 013702 024736
1851 017020 012700 000001
1852 017024 012703 000000
1853 017030 010311
1854 017032 016412 021772
1855 017036 012712 000001
1856 017042 010311
1857 017044 010112
1858 017046 016437 021762 001124
1859 017054 062737 124010 001124
1860 017062 012711 000001
1861 017066 011237 001126
1862 017072 032737 004000 001126
1863 017100 001770
1864 017102 023737 001124 001126
1865 017110 001403
1866 017112 104044
1867 017114 004737 023526
1868
1869 017120 012711 000005
1870 017124 012712 000017
1871 017130 012711 000001
1872 017134 012712 100000
1873 017140 042737 120010 001124
1874 017146 012711 000001
1875 017152 011237 001126
1876 017156 023737 001124 001126
1877 017164 001403
1878 017166 104044
1879 017170 004737 023526
1880
1881 017174

```

```

:*****
:*TEST 52 TEST PG RMR FOR PROPER OPERATION
:*****
TST52: SCOPE
JSR PC,SETMNT :CLR IIST AND ENTER MAINT. MODE
MOV ACR,R1 :R1=ACR
MOV ADR,R2 :R2=ADR
MOV #GO,R0 :R0=GO
MOV #PGTEE,R3 :R3=PGTE SELECT CODE
MOV R3,@R1 :SELECT PGTE
MOV PGITAB(4),@R2 :LOAD PGTE WITH ENABLE
MOV #GO,@R2 :TRANSMIT
MOV R3,@R1 :SELECT PGTE
MOV R1,@R2 :ACCESS PGTE AGAIN
MOV SIDTAB(4),%GDDAT :LOAD EXPECTED
ADD #ERR+PGMR+PRDY+IP,%GDDAT
1$: MOV #PGCSE,@R1 :SELECT PGCS
MOV @R2,%BDDAT :READ PGCS
BIT #PRDY,%BDDAT :TEST FOR RDY
BEQ 6 :LOOP UNTIL SET
CMP %GDDAT,%BDDAT :CHECK STATUS
BEQ 2$ :SKIP IF OK
ERROR+ 44 :FRR,PGMR, OR IP DID NOT SET ON
JSR PC,DMPREG
1868
2$: MOV #PGFE,@R1 :A PGMR ERROR
MOV #17,@R2 :SELECT PGF
MOV #PGCSE,@R1 :CLEAR PGF
MOV #ERR,@R2 :SELECT PGCS
MOV #ERR+PGMR+IP,%GDDAT :CLEAR ERR BY WRITING 1 TO IT
BIC #ERR+PGMR+IP,%GDDAT
MOV #PGCSE,@R1 :SELECT PGCS
MOV @R2,%BDDAT :READ PGCS
CMP %GDDAT,%BDDAT :CHECK STATUS
BEQ 5$ :SKIP IF OK
ERROR+ 44 :WRITING A ONE TO ERR DID NOT
JSR PC,DMPREG
5$: :CLEAR ERR,PGMR, AND IP

```

```

1883 (3)
1884 (3)
1885 (2) 017174 000004
1886 017176 004737 023476
1887 017202 013701 024732
1888 017206 013702 024736
1889 017212 012711 000002
1890 017216 016412 021772
1891 017222 012712 000001
1892 017226 016437 021762 001124
1893 017234 062737 114010 001124
1894 017242 012711 000003
1895 017246 032712 000010
1896 017252 001773
1897 017254 012711 000002
1898 017260 005212
1899 017262 012711 000001
1900 017266 011237 001126
1901 017272 023737 001124 001126
1902 017300 001403
1903 017302 104044
1904 017304 004737 023526
1905 017310 012746 100000
1906 017314 005216
1907 017316 001376
1908 017320 005026
1909 017322 012711 000006
1910 017326 012712 000017
1911 017332 012711 000001
1912 017336 012712 100000
1913 017342 042737 110010 001124
1914 017350 012711 000001
1915 017354 011237 001126
1916 017360 023737 001124 001126
1917 017366 001403
1918 017370 104044
1919 017372 004737 023526
1920 017376
1921 017376 000207

```

```

*****
*TEST 53 TEST ST RMR FOR PROPER OPERATION
*****
T53: SCOPE
      JSR PC,SSTMNT ;CLR IIST AND ENTER MAINT. MODE
      MOV ACR,R1 ;R1=ACR
      MOV ADR,R2 ;R2=ADR
      MOV #STEE,@R1 ;SELECT STEE
      MOV STITAB(4),@R2 ;LOAD STEE WITH ENABLE
      MOV #ENB,@R2 ;START TIMER
      MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED
      ADD #ERR+STMR+PRDY+IP,$GDDAT
1$: MOV #STCSE,@R1 ;SELECT STCS
      BIT #TMO,@R2 ;TEST FOR TMO
      BEQ 1$ ;LOOP UNTIL SET
      MOV #STEE,@R1 ;SELECT STEE
      INC @R2 ;ACCESS STEE AGAIN
      MOV #PGCSE,@R1 ;SELECT PGCS
      MOV @R2,$BDDAT ;READ PGCS
      CMP $GDDAT,$BDDAT ;CHECK STATUS
      BEQ 2$ ;SKIP IF OK
      ERROR+ 44 ;ERR,STMR, OR IP DID NOT SET ON
      JSR PC,DMPREG ;A STMR ERROR
2$: MOV #100000,-(SP) ;PUSH VALUE
4$: INC (SP) ;STALL
      BNE 4$
      CLR (SP)+
      MOV #STFE,@R1 ;SELECT STF
      MOV #17,@R2 ;CLEAR STF
      MOV #PGCSE,@R1 ;SELECT PGCS
      MOV #ERR,@R2 ;CLEAR ERR BY WRITING 1 TO IT
      BIC #ERR+STMR+IP,$GDDAT
      MOV #PGCSE,@R1 ;SELECT PGCS
      MOV @R2,$BDDAT ;READ PGCS
      CMP $GDDAT,$BDDAT ;CHECK STATUS
      BEQ 5$ ;SKIP IF OK
      ERROR+ 44 ;WRITING A ONE TO ERR DID NOT
      JSR PC,DMPREG ;CLEAR ERR,STMR, AND IP
5$: RTS PC ;:EXIT

```



1923 :ISSUE A PROGRAM-GENERATED REQUEST WITH INTERRUPT MASKS CLEAR WITH ALL COMBINATIONS OF  
 1924 :TRANSMIT ENABLES AND CHECK FOR:  
 1925 : CORRECT CONTENTS OF PGCS (RDY, IP, SELF-ID)  
 1926 : RTE BITS IN THE EXC REGISTER TO BE CLEAR  
 1927 : ALL OTHER FLAGS TO BE CLEAR ( STF, DCF, UIF)  
 1928 :

1929 017400

PGONL:  
 :\*\*\*\*\*  
 :\*TEST 54 CHECK OPERATION OF PG REQUESTS IN NORMAL MODE  
 :\*\*\*\*\*

1930 (3)  
 (3)  
 (2) 017400 000004  
 1931 017402 012777 100000 005322  
 1932 017410 005737 025020  
 1933 017414 001002  
 1934 017416 000137 020162  
 1935 017422 004737 023450  
 1936 017426 012737 017452 001110  
 1937 017434 012777 000007 005270  
 1938 017442 017737 005270 024756  
 1939 017450 005000  
 1940 017452 004737 023450  
 1941 017456 012777 000015 005246  
 1942 017464 012777 000010 005244  
 1943 017472 010037 001200  
 1944 017476 023727 025020 000003  
 1945 017504 001013  
 1946 017506 016401 021772  
 1947 017512 010146  
 1948 017514 006301  
 1949 017516 006301  
 1950 017520 006301  
 1951 017522 006301  
 1952 017524 052601  
 1953 017526 005101  
 1954 017530 040137 001200  
 1955 017534  
 1956 017534 010046  
 1957 017536 013700 001200  
 1958 017542 004737 023506  
 1959 017546 012600  
 1960 017550 113737 001200 001201  
 1961 017556 106037 001201  
 1962 017562 106037 001201  
 1963 017566 106037 001201  
 1964 017572 106037 001201  
 1965 017576 012777 000000 005126  
 1966 017604 013777 001200 005124  
 1967 017612 006301  
 1968 017614 005101  
 1969 017616 042701 177775  
 1970 017622 016437 021762 001124  
 1971 017630 050137 001124  
 1972 017634 052701 000001  
 1973 017640 012777 000001 005064  
 1974 017646 010177 005064  
 1975 017652 005001

IST54: SCOPE  
 MOV #CLR,@ACR ;CLEAR THE IIST  
 TST CONFIG ;ARE WE PREVENTED FROM ONLINE? (CONFIG-0)  
 BNE .+6 ;SKIP IF NOT  
 JMP 10\$ ;IF YES, GO TO NEXT TEST.  
 JSR PC,CLEARX ;CLEAR THE IIST AND FLAGS  
 MOV #1\$,SLPERR ;FIX START OF LOOP  
 MOV #DCF,@ACR  
 MOV @DCF,SNAP0 ;SAVE IMAGE OF DCF REGISTER  
 CLR R0 ;INIT PGTE DATA SETUP  
 1\$: JSR PC,CLEARX ;CLEAR IIST AND FLAGS  
 MOV #MTC,@ACR ;SELECT MTC  
 MOV #DSBT,@MTC ;DISABLE BOOTS  
 MOV R0,\$TMP1  
 CMP CONFIG,#3 ;SHOULD WE PREVENT XMIT OF ALL CODES?  
 BNE 11\$ ;NO IF NOT CONFIG #3.  
 MOV PCITAB(4),R1 ;YES -- GET MASK FOR SELF.  
 MOV R1,-(SP) ;SAVE INTR ENB  
 ASL R1 ;SHIFT 4X TO BOOT ENB  
 ASL R1  
 ASL R1  
 ASL R1  
 BIS (SP)+,R1 ;GET INTR ENB MASK  
 COM R1  
 BIC R1,\$TMP1 ;CLEAR BITS FOR DESTINATION OTHER THAN SELF.  
 11\$:  
 MOV R0,-(SP) ;SAVE COUNT VALUE  
 MOV \$TMP1,R0 ;GET ENABLE FOR PARITY GENERATOR  
 JSR PC,PARCAL ;CALCULATE PARITY OF DATA  
 MOV (SP)+,R0 ;RESTORE COUNT VALUE.  
 MOV \$TMP1,\$TMP1+1  
 RORB \$TMP1+1 ;POSITION DATA  
 RORB \$TMP1+1  
 RORB \$TMP1+1  
 RORB \$TMP1+1  
 MOV #PGTEE,@ACR  
 MOV \$TMP1,@PGTE ;LOAD TRANSMIT ENABLES  
 ASL R1 ;GET PARITY  
 COM R1 ;FIX PAR BIT  
 2\$: BIC #177775,R1  
 MOV SIDTAB(4),\$GDDAT ;GET SELF-ID  
 BIS R1,\$GDDAT  
 BIS #1,R1  
 MOV #PGCE,@ACR  
 MOV R1,@PGCS ;LOAD PARITY, SET GO  
 CLR R1

1976	017654	052737	004000	001124		BIS	#PRDY,\$GDDAT	
1977	017662	036437	021772	001200		BIT	PGITAB(4),\$TMP1	:TEST FOR PGI ENABLE
1978	017670	001403				BEQ	3\$	:SKIP IF CLEAR
1979	017672	052737	000010	001124		BIS	#IP,\$GDDAT	:SET IP IF RELEVANT
1980	017700	105377	005026		3\$:	DECB	@ACR	
1981	017704	017737	005026	001126		MOV	@PGCS,\$BDDAT	
1982	017712	032737	004000	001126		BIT	#PRDY,\$BDDAT	:WAIT FOR READY
1983	017720	001005				BNE	4\$	
1984	017722	005301				DEC	R1	:BUT DON'T WAIT TOO LONG
1985	017724	001365				BNE	3\$	
1986	017726	104044				ERROR+	44	:PRDY DID NOT SET ON A PG XFER
1987	017730	004737	023526			JSR	PC,DMPREG	
1988	017734	023737	001126	001124	4\$:	CMP	\$BDDAT,\$GDDAT	:IS PGCS OK?
1989	017742	001403				BEQ	5\$	
1990	017744	104044				ERROR+	44	:PGCS STATUS ERROR ON PG XFER
1991	017746	004737	023526			JSR	PC,DMPREG	
1992	017752	005037	001124		5\$:	CLR	\$GDDAT	:SET EXPECTED RTE FLAGS
1993	017756	012777	000010	004746		MOV	#EXCE,@ACR	
1994	017764	017737	004746	001126		MOV	@EXC,\$BDDAT	:GET EXC REGISTER
1995	017772	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	
1996	020000	001403				BEQ	6\$	
1997	020002	104044				ERROR+	44	:EXC NOT 0 ON A PG XFER
1998	020004	004737	023526			JSR	PC,DMPREG	
1999	020010	036437	021772	001200	6\$:	BIT	PGITAB(4),\$TMP1	:CHECK FOR LEGAL PGF
2000	020016	001403				BEQ	61\$	:SKIP IF NO
2001	020020	056437	021772	001124		BIS	PGITAB(4),\$GDDAT	:LOAD PGF
2002	020026	036437	022002	001200	61\$:	BIT	PGBTAB(4),\$TMP1	:CHECK FOR LEGAL PGB
2003	020034	001403				BEQ	62\$	:SKIP IF NO
2004	020036	056437	022002	001124		BIS	PGBTAB(4),\$GDDAT	:LOAD PGB
2005	020044	012777	000005	004660	62\$:	MOV	#PGFE,@ACR	
2006	020052	017737	004660	001126		MOV	@PGF,\$BDDAT	:READ PGF
2007	020060	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	:CHECK PGF
2008	020066	001403				BEQ	7\$	
2009	020070	104044				ERROR+	44	:PGF ERROR ON PG XFER
2010	020072	004737	023526			JSR	PC,DMPREG	
2011	020076	005037	001124		7\$:	CLR	\$GDDAT	:EXPECTED = 0
2012	020102	017737	004630	001126		MOV	@STF,\$BDDAT	:GET STF
2013	020110	001403				BEQ	8\$	:STF SHOULD BE 0
2014	020112	104044				ERROR+	44	:STF NO 0 ON A PG XFER
2015	020114	004737	023526			JSR	PC,DMPREG	
2016	020120	013737	024756	001124	8\$:	MOV	SNAPO,\$GDDAT	:GET EXPECTED DATA FOR DCF
2017	020126	017737	004604	001126		MOV	@DCF,\$BDDAT	:GET DCF REGISTER
2018	020134	023737	001126	001124		CMP	\$BDDAT,\$GDDAT	:CHECK THAT THERE IS NO CHANGE
2019	020142	001403				BEQ	9\$	
2020	020144	104044				ERROR+	44	:DCF ERROR ON A PG XFER
2021	020146	004737	023526			JSR	PC,DMPREG	
2022	020152	105200			9\$:	INCB	R0	:INCREMENT TEST DATA
2023	020154	001402				BEQ	10\$	:SKIP WHEN DONE
2024	020156	000137	017452			JMP	1\$	
2025	020162				10\$:			

```

2027 ;FORCE A BREAK USING THE MAINTENANCE REGISTER AND TEST
2028 ;THE DCF REGISTER
2029
2030 .....
(3) ;*TEST 55 TEST THE DCF REGISTER
(3) .....
(2) TST55: SCOPE
2031 020162 000004 JSR PC,CLEARX ;INIT THE IIST
2032 020164 004737 023450 MOV #DSBT,$TMP1 ;SET VALUE FOR MTC TO CLEAR BREAK.
2033 020170 012737 000010 001200 MOV #0,$TMP2 ;SET VALUE FOR CLEARING UNWANTED BITS.
2034 020176 012737 000000 001202 TST CONFIG ;SEE WHAT CONFIGURATION.
2035 020204 005737 025020 BNE 11$ ;IF NOT MAINT., SKIP ON.
2036 020210 001005 JSR PC,SETMNT ;IF MAINT. ONLY, SET MAINT. LOOPBACK.
2037 020212 004737 023306 MOV #MLEN+DSBT,$TMP1 ;NEW VALUE FOR CLEARING BRK.
2038 020216 012737 000012 001200 11$: MOV BRKTAB(4),R1 ;GET BRK BIT FOR OWN LINE.
2039 020224 016401 022002 BIS DCFTAB(4),R1 ;AND DCF BIT.
2040 020230 056401 021772 COM R1 ;MAKE 'BIC' MASK.
2041 020234 005101 BIC #170360,R1
2042 020236 042701 170360 MOV R1,$TMP2 ;STORE THE VALUE.
2043 020242 010137 001202 MOV #DCFE,@ACR ;SELECT DCF
2044 020246 012777 000007 004456 MOV @DCF,$SNAPO ;SAVE THE DCF CONTENTS
2045 020254 017737 004456 024756 MOV $SNAPO,$GDDAT ;COPY DCF
2046 020262 013737 024756 001124 CMP CONFIG,#1 ;ANY OTHER INTERFACES?
2047 020270 023727 025020 000001 BNE 12$
2048 020276 001003 MOV #7400,$GDDAT ;SAY EXPECTED HAS OTHER BRK BITS SET.
2049 020300 012737 007400 001124 12$: TST CONFIG ;IN MAINT. MODE?
2050 020306 005737 025020 BNE 13$
2051 020312 001002 CLR $GDDAT ;YES -- ALL DCF BITS CLEAR.
2052 020314 005037 001124 001124 13$: BIC BRKTAB(4),$GDDAT ;REMOVE SID BRK
2053 020320 046437 022002 001124 MOV $SNAPO,$BDDAT ;READ ACTUAL
2054 020326 013737 024756 001126 CMP $GDDAT,$BDDAT ;CHECK FOR LEGAL DCF
2055 020334 023737 001124 001126 BEQ 1$ ;SKIP IF OK
2056 020342 001403 ERROR+ 44 ;INCORRECT DCF FOR SELECTED IIST
2057 020344 104044 JSR PC,DMPREG
2058 020346 004737 023526 1$: MOV #MTC,@ACR ;SELECT MTC
2059 020352 012777 000015 004352 MOV #DSBT+MDS,@MTC ;FORCE A BREAK
2060 020360 012777 000011 004350 MOV #DCFE,@ACR ;SELECT DCF
2061 020366 012777 000007 004336 MOV @DCF,$BDDAT ;READ DCF
2062 020374 017737 004336 001126 BIC $TMP2,$BDDAT ;CLEAR BITS NOT OUR OWN.
2063 020402 043737 001202 001126 MOV BRKTAB(4),$GDDAT ;SET RELEVANT BRK BIT
2064 020410 016437 022002 001124 BIS DCFTAB(4),$GDDAT ;SET RELEVANT DCF BIT
2065 020416 056437 021772 001124 CMP $GDDAT,$BDDAT ;CHECK DCF
2066 020424 023737 001124 001126 BEQ 2$ ;SKIP IF OK
2067 020432 001403 ERROR+ 44 ;INCORRECT DCF AFTER FORCING A BREAK
2068 020434 104044 JSR PC,DMPREG
2069 020436 004737 023526 2$: MOV #MTC,@ACR ;SELECT MTC
2070 020442 012777 000015 004262 MOV $TMP1,@MTC ;CLEAR BRK BIT
2071 020450 013777 001200 004260 MOV #DCFE,@ACR ;SELECT DCF
2072 020456 012777 000007 004246 MOV @DCF,$BDDAT ;READ DCF
2073 020464 017737 004246 001126 BIC $TMP2,$BDDAT
2074 020472 043737 001202 001126 BIC BRKTAB(4),$GDDAT ;CLEAR OUT BRK BIT
2075 020500 046437 022002 001124 CMP $GDDAT,$BDDAT ;CHECK DCF
2076 020506 023737 001124 001126 BEQ 3$ ;SKIP IF BRK=0
2077 020514 001403 ERROR+ 44 ;INCORRECT DCF AFTER REMOVING THE BRK CONDITION
2078 020516 104044 JSR PC,DMPREG
2079 020520 004737 023526 3$: MOV SIDTAB(4),$GDDAT ;LOAD SELF ID #
2079 020524 016437 021762 001124

```

```

2080 020532 052737 004010 001124      BIS      #IP+PRDY,$GDDAT ;ADD OTHER EXPECTED DATA
2081 020540 012777 000007 004164      MOV      #DCFE,@ACR ;SELECT DCF
2082 020546 013777 001202 004162      MOV      $TMP2,@DCF ;CLEAR UNWANTED FLAGS.
2083 020554 012777 000001 004150      MOV      #PGCSE,@ACR ;SELECT PGCS
2084 020562 017737 004150 001126      MOV      @PGCS,$BDDAT ;READ PGCS
2085 020570 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CHECK PGCS FOR IP
2086 020576 001403      BEQ      4$ ;SKIP IF OK
2087 020600 104044      ERROR+ 44 ;DCF FLAG DID NOT SET IP
2088 020602 004737 023526      JSR      PC,DMPREG
2089 020606 017746 004130      4$: MOV      @111V,-(SP) ;SAVE OLD PI
2090 020612 017746 004126      MOV      @111P,-(SP) ;DATA
2091 020616 012777 020672 004116      MOV      #5$,@111V ;LOAD NEW
2092 020624 012777 000340 004112      MOV      #340,@111P ;PI DATA
2093 020632 012777 000001 004072      MOV      #PGCSE,@ACR ;SELECT PGCS
2094 020640 012777 000004 004070      MOV      #IE,@PGCS ;ENABLE PI
2095 020646 005037 177776      CLR      PS ;ALLOW PI
2096 020652 000240      NOP ;STALL
2097 020654 012737 000340 177776      MOV      #340,PS ;LOCKOUT PI
2098 020662 104044      ERROR+ 44 ;NO PI ON A DCF FLAG
2099 020664 004737 023526      JSR      PC,DMPREG
2100 020670 000401      BR ;SKIP
2101 020672 022626      5$: POPPOP ;FIX STACK
2102 020674 012677 004044      6$: MOV      (SP)+,@111P ;RESTORE OLD
2103 020700 012677 004036      MOV      (SP)+,@111V ;PI DATA
2104 020704 012737 000000 001124      MOV      #0,$GDDAT ;RELOAD EXPECTED DCF VALUE
2105 020712 012777 000007 004012      MOV      #DCFE,@ACR ;SELECT DCF
2106 020720 016477 021772 004010      MOV      DCFTAB(4),@DCF ;CLEAR DCF FLAG
2107 020726 005377 004000      DEC      @ACR ;RESELECT DCF
2108 020732 017737 004000 001126      MOV      @DCF,$BDDAT ;READ DCF
2109 020740 043737 001202 001126      BIC      $TMP2,$BDDAT ;MASK OUT JUNK.
2110 020746 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CHECK DCF
2111 020754 001403      BEQ      7$ ;SKIP IF DCF=0
2112 020756 104044      ERROR+ 44 ;WRITING A 1 TO DCF FLAG DID NOT CLEAR IT
2113 020760 004737 023526      JSR      PC,DMPREG
2114 020764 004737 023450      7$: JSR      PC,CLEARX ;INIT. THE INTERFACE.
2115 020770 012777 000004 003734      MOV      #IMSKE,@ACR ;SELECT THE IMSK REGISTER.
2116 020776 012777 000017 003732      MOV      #17,@IMSK ;SET ALL INTERRUPT MASKS.
2117 021004 012777 000015 003720      MOV      #MTC,@ACR ;SELECT MTC.
2118 021012 012777 000012 003716      MOV      #DSBT+MLEN,@MTC ;DO MAINT. LOOP TO CLEAR BREAKS.
2119 021020 027777 003706 003704      CMP      @ACR,@ACR ;DELAY
2120 021026 027777 003700 003676      CMP      @ACR,@ACR ;
2121 021034 027777 003672 003670      CMP      @ACR,@ACR ;
2122 021042 012777 000011 003666      MOV      #DSBT+MDS,@MTC ;CAUSE BREAK
2123 021050 027777 003656 003654      CMP      @ACR,@ACR ;
2124 021056 027777 003650 003646      CMP      @ACR,@ACR ;
2125 021064 012737 000000 001124      MOV      #0,$GDDAT ;SAY GOOD IS 0
2126 021072 012777 000007 003632      MOV      #DCFE,@ACR ;SELECT DCF
2127 021100 017737 003632 001126      MOV      @DCF,$BDDAT ;GET DCF
2128 021106 042737 007400 001126      BIC      #7400,$BDDAT ;CLEAR BREAKS
2129 021114 023737 001126 001124      CMP      $BDDAT,$GDDAT ;VERIFY
2130 021122 001403      BEQ      8$
2131 021124 104044      ERROR+ 44
2132 021126 004737 023526      JSR      PC,DMPREG
2133 021132 000207      8$: RTS      PC ;EXIT
    
```

```

2135 ;ISSUE A SANITY TIMER REQUEST WITH INTERRUPT MASKS CLEAR WITH ALL COMBINATIONS OF
2136 ;TRANSMIT ENABLES AND CHECK FOR:
2137 ;CORRECT CONTENTS OF PGCS (RDY, IP, SELF-ID)
2138 ;RTE BITS IN THE EXC REGISTER TO BE CLEAR
2139 ;ALL OTHER FLAGS TO BE CLEAR (PGF, DCF, UIF)
2140

```

2141 021134

STONL:

```

2142 ;*****
(3) ;*TEST 56 CHECK OPERATION OF ST REQUESTS IN NORMAL MODE
(3) ;*****

```

```

(2) 021134 000004
2143 021136 012777 100000 003566 1ST56: SCOPE
2144 021144 004737 023450 MOV #CLR,@ACR ;CLEAR THE IIST
2145 021150 005737 025020 JSR PC,CLEARX
2146 021154 001001 TST CONFIG
2147 021156 000207 BNE .+4
2148 021160 012737 021204 001110 RTS PC ;EXIT IF WE CAN'T XMIT
2149 021166 012777 000007 003536 MOV #1$,$LPERR ;FIX START OF LOOP
2150 021174 017737 003536 024756 MOV @DCF,$NAPO ;SAVE IMAGE OF DCF REGISTER
2151 021202 005000 CLR R0 ;INIT STTE DATA SETUP
2152 021204 012777 100000 003520 1$: MOV #CLR,@ACR ;CLEAR THE IIST
2153 021212 012777 000005 003512 MOV #PGFE,@ACR ;SELECT PGF
2154 021220 012777 007400 003510 MOV #7400,@PGF ;CLEAR ALL
2155 021226 012777 007400 003502 MOV #7400,@STF ;FLAGS
2156 021234 012777 000017 003474 MOV #17,@DCF ;CLEAR BRK
2157 021242 012777 000015 003462 MOV #MTC,@ACR ;SELECT MTC
2158 021250 012777 000010 003460 MOV #DSBT,@MTC ;DISABLE BOOTS
2159 021256 010037 001200 MOV R0,$TMP1
2160 021262 023727 025020 000003 CMP CONFIG,#3 ;SHOULD WE PREVENT XMIT OF ALL CODES?
2161 021270 001013 BNE 11$ ;NO IF NOT CONFIG #3.
2162 021272 016401 021772 MOV STTAB(4),R1 ;YES -- GET MASK FOR SELF.
2163 021276 010146 MOV R1,-(SP) ;SAVE INTR ENB
2164 021300 006301 ASL R1 ;SHIFT 4X TO BOOT ENB
2165 021302 006301 ASL R1
2166 021304 006301 ASL R1
2167 021306 006301 ASL R1
2168 021310 052601 BIS (SP)+,R1 ;GET INTR ENB MASK
2169 021312 005101 COM R1
2170 021314 040137 001200 BIC R1,$TMP1 ;CLEAR BITS FOR DESTINATION OTHER THAN SELF.
2171 021320 11$:
2172 021320 010046 MOV R0,-(SP) ;SAVE COUNT VALUE
2173 021322 013700 001200 MOV $TMP1,R0 ;GET ENABLE FOR PARITY GENERATOR
2174 021326 004737 023506 JSR PC,PARCAL ;CALCULATE PARITY OF DATA
2175 021332 012600 MOV (SP)+,R0 ;RESTORE COUNT VALUE.
2176 021334 113737 001200 001201 MOV $TMP1,$TMP1+1
2177 021342 106037 001201 RORB $TMP1+1 ;POSITION DATA
2178 021346 106037 001201 RORB $TMP1+1
2179 021352 106037 001201 RORB $TMP1+1
2180 021356 106037 001201 RORB $TMP1+1
2181 021362 012777 000002 003342 MOV #STEE,@ACR
2182 021370 013777 001200 003340 MOV $TMP1,@STEE ;LOAD TRANSMIT ENABLES
2183 021376 006301 ASL R1 ;GET PARITY
2184 021400 005101 COM R1 ;FIX PAR BIT
2185 021402 042701 177775 2$: BIC #177775,R1
2186 021406 010137 001124 MOV R1,$GDDAT ;LOAD PARITY
2187 021412 052701 000001 BIS #1,R1

```

```

2188 021416 012777 000003 003306      MOV      #STCSE,@ACR
2189 021424 010177 003306              MOV      R1,@STCS      ;LOAD PARITY, ENB, AND COUNT
2190 021430 005001              CLR      R1
2191 021432 052737 177411 001124      BIS      #TMO+177401,$GDDAT
2192 021440 012777 000004 003264      MOV      #IMSKE,@ACR      ;SELECT IMSK
2193 021446 105377 003260 3$:      DECB    @ACR
2194 021452 017737 003260 001126      MOV      @STCS,$BDDAT
2195 021460 032737 000010 001126      BIT      #TMO,$BDDAT      ;WAIT FOR TMO
2196 021466 001005              BNE     4$
2197 021470 005301              DEC     R1      ;BUT DON'T WAIT TOO LONG
2198 021472 001365              BNE     3$
2199 021474 104044      ERROR+  44      ;TMO DID NOT SET ON A ST XFER
2200 021476 004737 023526              JSR    PC,DMPREG
2201 021502 012777 000003 003222 4$:      MOV      #STCSE,@ACR      ;SELECT STCS
2202 021510 017737 003222 001126      MOV      @STCS,$BDDAT      ;READ STCS
2203 021516 023737 001126 001124      CMP     $BDDAT,$GDDAT      ;IS STCS OK?
2204 021524 001403              BEQ     5$
2205 021526 104044      ERROR+  44      ;STCS STATUS ERROR ON ST XFER
2206 021530 004737 023526              JSR    PC,DMPREG
2207 021534 005037 001124 5$:      CLR     $GDDAT      ;SET EXPECTED RTE FLAGS
2208 021540 012777 000010 003164      MOV      #EXCE,@ACR
2209 021546 017737 003164 001126      MOV      @EXC,$BDDAT      ;GET EXC REGISTER
2210 021554 023737 001124 001126      CMP     $GDDAT,$BDDAT
2211 021562 001403              BEQ     6$
2212 021564 104044      ERROR+  44      ;EXC NOT 0 ON A ST XFER
2213 021566 004737 023526              JSR    PC,DMPREG
2214 021572 036437 021772 001200 6$:      BIT     STITAB(4),$TMP1      ;CHECK FOR STF
2215 021600 001403              BEQ     61$      ;SKIP IF 0
2216 021602 056437 021772 001124      BIS     STITAB(4),$GDDAT      ;SET STF
2217 021610 036437 022002 001200 61$:      BIT     STBTAB(4),$TMP1      ;CHECK FOR STB
2218 021616 001403              BEQ     62$      ;SKIP IF 0
2219 021620 056437 022002 001124      BIS     STBTAB(4),$GDDAT      ;SET STB
2220 021626 012777 000006 003076 62$:      MOV      #STFE,@ACR
2221 021634 017737 003076 001126      MOV      @STF,$BDDAT      ;READ STF
2222 021642 023737 001124 001126      CMP     $GDDAT,$BDDAT      ;CHECK DATA
2223 021650 001403              BEQ     7$
2224 021652 104044      ERROR+  44      ;STF ERROR ON ST XFER
2225 021654 004737 023526              JSR    PC,DMPREG
2226 021660 005037 001124 7$:      CLR     $GDDAT      ;EXPECTED = 0
2227 021664 012777 000005 003040      MOV      #PGFE,@ACR      ;SELECT PGF
2228 021672 017737 003040 001126      MOV      @PGF,$BDDAT      ;GET PGF
2229 021700 001403              BEQ     8$      ;PGF SHOULD BE 0
2230 021702 104044      ERROR+  44      ;PGF NO 0 ON A ST XFER
2231 021704 004737 023526              JSR    PC,DMPREG
2232 021710 013737 024756 001124 8$:      MOV      SNAPO,$GDDAT      ;GET EXPECTED DATA FOR DCF
2233 021716 012777 000007 003006      MOV      #DCFE,@ACR      ;SELECT DCF
2234 021724 017737 003006 001126      MOV      @DCF,$BDDAT      ;GET DCF REGISTER
2235 021732 023737 001126 001124      CMP     $BDDAT,$GDDAT      ;CHECK THAT THERE IS NO CHANGE
2236 021740 001403              BEQ     9$
2237 021742 104044      ERROR+  44      ;DCF ERROR ON A ST XFER
2238 021744 004737 023526              JSR    PC,DMPREG
2239 021750 105200 9$:      INCB   R0      ;INCREMENT TEST DATA
2240 021752 001402              BEQ     10$      ;SKIP WHEN DONE
2241 021754 000137 021204              JMP     1$
2242 021760 000207 10$:      RTS     PC      ;EXIT
  
```

```
2244 .SBTTL LINE CONTROL BIT TABLES
2245
2246 021762 000000 SIDTAB: SID0 ;SID TABLE
2247 021764 000400 SID1
2248 021766 001000 SID2
2249 021770 001400 SID3
2250
2251 021772 PGITAB: ;PROGRAM INTERRUPT ENABLE TABLE
2252 021772 STITAB: ;SANITY INTERRUPT ENABLE TABLE
2253 021772 IMITAB: ;INTERRUPT MASK INHIBIT TABLE
2254 021772 PIFTAB: ;PGM INTERRUPT FLAG TABLE
2255 021772 SIFTAB: ;SANITY INTERRUPT FLAG TABLE
2256 021772 DCFTAB: ;DCLO INTERRUPT FLAG TABLE
2257 021772 000001 RTETAB: RTE0 ;INVALID INPUT FLAG TABLE
2258 021774 000002 RTE1
2259 021776 000004 RTE2
2260 022000 000010 RTE3
2261
2262 022002 PGBTAB: ;PROGRAM BOOT ENABLE TABLE
2263 022002 STBTAB: ;SANITY BOOT ENABLE TABLE
2264 022002 BMITAB: ;BOOT MASK INHIBIT TABLE
2265 022002 PBFTAB: ;PGM BOOT FLAG TABLE
2266 022002 SBFTAB: ;SANITY BOOT FLAG TABLE
2267 022002 BRKTAB: ;BREAK STATE TABLE
2268 022002 000400 UVITAB: U10 ;UNEXPECTED INPUT FLAG TABLE
2269 022004 001000 U11
2270 022006 002000 U12
2271 022010 004000 U13
```

2273 022012

2274  
2275  
2276  
2277  
2278  
2279  
2280  
2281

ENDTST:

:ON FIRST PASS, DETERMINE SANITY-TIMER COUNT RATE (256 US, 8.192 MS,  
:OR 16.384 MS) AND PRINT IT OUT.  
:ON SUBSEQUENT PASSES, CHECK FOR CONSISTENCY.  
:  
:MAINTENANCE LOOPBACK IS USED, WITH HARDWARE SELF-ID.

(3)  
(3)

\*\*\*\*\*  
:TES 57 DETERMINE SANITY TIMER COUNT RATE  
:\*\*\*\*\*

(2) 022012 000004

TST57: SCOPE

2282 022014 012737 000340 177776  
2283 022022 004737 023306  
2284 022026 012777 000015 002676  
2285 022034 012777 000013 002674  
2286 022042 117704 002666  
2287 022046 010437 024754  
2288 022052 006304  
2289 022054 012777 000002 002650  
2290 022062 016477 021772 002646  
2291 022070 017746 002650  
2292 022074 017746 002642  
2293 022100 012777 022234 002634  
2294 022106 012777 000340 002630  
2295 022114 005000  
2296 022116 013701 024732  
2297 022122 013702 024736  
2298 022126 012703 000001  
2299 022132 012704 004000  
2300 022136 012705 000007  
2301 022142 012711 000003  
2302 022146 012712 000401  
2303 022152 005037 177776

MOV #340,PSW ;RAISE PROCESSOR PRIORITY  
JSR PC,SETMNT ;INIT THE INTERFACE  
MOV #MTC,@ACR ;SELECT MTC REGISTER  
MOV #DSBT+MLN+MDS,@MTC ;SET LOOPBACK, DISABLE BOOT & DRIVERS  
MOVB @ACRH,R4 ;GET HARDWARE SELF-ID  
MOV R4,LINMBR ;PUT IT INTO LINE NUMBER WORD  
ASL R4 ;MAKE A WORD INDEX  
MOV #STEE,@ACR ;SELECT STEE (S.T. TRANSMIT ENABLE)  
MOV STITAB(R4),@STEE ;SET XMIT ENABLE FOR SELF  
MOV @I1IP,-(SP) ;SAVE OLD INTR VECTOR INFO  
MOV @I1IV,-(SP)  
MOV #4,@I1IV ;SETUP NEW INTR VECTOR  
MOV #340,@I1IP  
CLR R0 ;CLEAR CHARACTER COUNT  
MOV ACR,R1 ;R1 = ACR ADDRESS  
MOV ADR,R2 ;R2 = ADR ADDRESS  
MOV #PGCSE,R3 ;R3 = PGCS ACCESS CODE  
MOV #PRDY,R4 ;R4 = PG RDY BIT MASK  
MOV #PTP+IE+GO,R5 ;R5 = VALUE FOR LOADING PGCS  
MOV #STCSE,(R1) ;SELECT STCS  
MOV #401,(R2) ;START SANITY TIMER FOR DOING 2 COUNTS  
CLR PSW ;ALLOW INTERRUPTS

2304  
2305 022156 010311  
2306 022160 010512  
2307 022162 005200  
2308 022164 001404  
2309 022166 010311  
2310 022170 030412  
2311 022172 001371  
2312 022174 000774  
2313

1\$: MOV R3,(R1) ;SELECT PGCS  
MOV R5,(R2) ;XMIT NULL PG COMMAND, SET INTR ENABLE  
INC R0 ;COUNT # OF CHARS TRANSMITTED  
BEQ 3\$ ;BUT DON'T DO FOREVER.  
2\$: MOV R3,(R1) ;SELECT PGCS  
BIT R4,(R2) ;LOOK FOR PG RDY  
BNE 1\$ ;XMIT AGAIN IF SET  
BR 2\$ ;LOOP IF NOT

2314  
2315 022176 012737 000340 177776  
2316 022204 012737 177410 001124  
2317 022212 012711 000003  
2318 022216 011237 001126  
2319 022222 104044  
2320 022224 004737 023526  
2321 022230 000137 023260  
2322

:COME HERE IF NO TIMEOUT INTR SEEN AFTER 64K TRANSMITS.  
3\$: MOV #340,PSW ;RAISE PROCESSOR TO LOCKOUT INTERRUPTS  
MOV #177400+TMO,\$GDDAT ;SETUP EXPECTED DATA  
MOV #STCSE,(R1) ;SELECT STCS  
MOV (R2),\$BDDAT ;GET CONTENTS OF STCS FOR PRINTOUT  
ERROR+ 44 ;NO TIMEOUT SEEN AFTER 64K TRANSMITS.  
JSR PC,DMPREG  
JMP 99\$ ;THEN JUST EXIT

2323  
2324 022234 062706 000004  
2325 022240 010046

:COME HERE ON INTERRUPT, CHECK THAT IT'S THE CORRECT ONE.  
4\$: ADD #4,SP ;POP STACK  
MOV R0,-(SP) ;SAVE COUNT



```

2326 022242 010311          MOV      R3,(R1)          ;SELECT PGCS
2327 022244 012705 000062  MOV      #50.,R5         ;WAIT FOR FINAL PG RDY
2328 022250 005305          DEC      R5
2329 022252 001376          BNE     .-2
2330 022254 011237 001126  MOV      (R2),%BDDAT     ;GET PGCS CONTENTS
2331 022260 011137 001124  MOV      (R1),%GDDAT     ;GET SELF-ID
2332 022264 042737 000017 001124  BIC     #17,%GDDAT      ;CLEAR CODE
2333 022272 052737 004016 001124  BIS     #PRDY+IE+IP+PTP,%GDDAT ;SET UP EXPECTED VALUE
2334 022300 023737 001126 001124  CMP     %BDDAT,%GDDAT   ;VERIFY PGCS CONTENTS
2335 022306 001403          BEQ     5$
2336 022310 104044          ERROR+ 44              ;PGCS INCORRECT AFTER INTERRUPT
2337 022312 004737 023526  JSR     PC,DMPREG
2338 022316 005037 001124 5$:  CLR     %GDDAT          ;SETUP EXPECTED VALUE FOR PGF, DCF, AND EXC
2339 022322 012777 000005 002402  MOV     #PGFE,@ACR      ;SELECT PGF
2340 022330 017737 002402 001126  MOV     @PGF,%BDDAT     ;GET PG RECEIVE FLAGS
2341 022336 001403          BEQ     6$
2342 022340 104044          ERROR+ 44              ;PGF NOT 0 AFTER INTERRUPT
2343 022342 004737 023526  JSR     PC,DMPREG
2344 022346 012777 000007 002356 6$:  MOV     #DCF,%ACR      ;SELECT DCF
2345 022354 017737 002356 001126  MOV     @DCF,%BDDAT     ;GET FLAGS & TEST THEM
2346 022362 001403          BEQ     7$
2347 022364 104044          ERROR+ 44              ;DCF NOT 0 AFTER INTERRUPT
2348 022366 004737 023526  JSR     PC,DMPREG
2349 022372 012777 000010 002332 7$:  MOV     #EXCE,@ACR     ;SELECT EXC
2350 022400 017737 002332 001126  MOV     @EXC,%BDDAT     ;GET FLAGS & TEST THEM
2351 022406 001403          BEQ     8$
2352 022410 104044          ERROR+ 44              ;EXC NOT CLEAR AFTER INTERRUPT
2353 022412 004737 023526  JSR     PC,DMPREG
2354 022416 012737 000017 001124 8$:  MOV     #17,%GDDAT     ;SETUP EXPECTED VALUE FOR STF
2355 022424 012777 000006 002300  MOV     #STFE,@ACR      ;SELECT STF
2356 022432 017737 002300 001126  MOV     @STF,%BDDAT     ;GET STF
2357 022440 023737 001126 001124  CMP     %BDDAT,%GDDAT   ;CHECK VALUE
2358 022446 001403          BEQ     9$
2359 022450 104044          ERROR+ 44              ;STF INCORRECT AFTER INTERRUPT
2360 022452 004737 023526  JSR     PC,DMPREG
2361
2362 022456 012637 001126 9$:  MOV     (SP)+,%BDDAT    ;GET COUNT VALUE
2363 022462 005737 025022  TST     STCNT          ;IS THERE A VALID COUNT?
2364 022466 001402          BEQ     .+6
2365 022470 000137 023200  JMP     20$            ;IF YES, GO VERIFY CONSISTENCY.
2366 022474 012737 000011 001124  MOV     #9.,%GDDAT     ;NO -- SO START CHECKING VALUE.
2367 022502 023737 001126 001124  CMP     %BDDAT,%GDDAT   ;IS COUNT .LT 9.?
2368 022510 002002          BGE     .+6           ;GO ON IF NO
2369 022512 000137 023170  JMP     12$            ;ERROR IF YES (COUNTER TOO FAST)
2370 022516 012737 000024 001124  MOV     #20.,%GDDAT
2371 022524 023737 001126 001124  CMP     %BDDAT,%GDDAT
2372 022532 003047          BGT     10$           ;GREATER THAN 20.?
2373 022534 013737 001126 025022  MOV     %BDDAT,STCNT    ;NO, SO BETWEEN 9 & 21 -- VALID
2374 022542 012737 000024 025026  MOV     #20.,MAXCNT    ;SAVE MAX LIMIT
2375 022550 012737 000011 025024  MOV     #9.,MINCNT     ;AND MIN LIMIT, FOR FUTURE TESTING.
2376 022556 005737 000042  TST     @#42          ;ARE WE IN ACT?
2377 022562 001402          BEQ     .+6           ;BEQ IF NO.
2378 022564 000137 023260  JMP     99$           ;IF YES, DON'T PRINT.
2379 022570 104401 022576  TYPE   ,65$          ;:TYPE ASCII STRING
(1) 022574 000424          BR      64$          ;:GET OVER THE ASCII
(1) ;:65$: .ASCII <200>/SANITY TIMER SET FOR 256US PER COUNT./<200>

```

```
(1) 022646 000137 023260 64$: JMP 99$
2380 022646 000137 023260
2381
2382 022652 012737 000500 001124 10$: MOV #320.,$GDDAT
2383 022660 023737 001126 001124 CMP $BDDAT,$GDDAT ;CHECK FOR COUNT BETWEEN 320.
2384 022666 002540 BLT 12$
2385 022670 012737 001200 001124 MOV #640.,$GDDAT
2386 022676 023727 001126 001200 CMP $BDDAT,#640. ;AND 640.
2387 022704 003045 BGT 11$
2388 022706 013737 001126 025022 MOV $BDDAT,STCCNT ;SAVE COUNT
2389 022714 012737 000500 025024 MOV #320.,MINCNT ;AND LIMITS
2390 022722 012737 001200 025026 MOV #640.,MAXCNT
2391 022730 005737 000042 TST @#42 ;ARE WE IN ACT?
2392 022734 001151 BNE 99$ ;IF YES, DON'T PRINT.
2393 022736 104401 022744 TYPE ,67$ ;;TYPE ASCIZ STRING
(1) 022742 000425 BR 66$ ;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <200>/SANITY TIMER SET FOR 8.192MS PER COUNT./<200>
(1) 66$:
2394 023016 000520 BR 99$
2395
2396 023020 012737 001200 001124 11$: MOV #640.,$GDDAT
2397 023026 023737 001126 001124 CMP $BDDAT,$GDDAT ;COUNT .LT 640.?
2398 023034 002455 BLT 12$ ;ERROR IF YES.
2399 023036 012737 002400 001124 MOV #1280.,$GDDAT
2400 023044 023727 001126 002400 CMP $BDDAT,#1280. ;GREATER THAN 1280.?
2401 023052 003046 BGT 12$ ;ERROR IF YES.
2402 023054 013737 001126 025022 MOV $BDDAT,STCCNT ;SAVE COUNT
2403 023062 012737 001200 025024 MOV #640.,MINCNT ;AND LIMITS
2404 023070 012737 002400 025026 MOV #1280.,MAXCNT
2405 023076 005737 000042 TST @#42 ;ARE WE IN ACT?
2406 023102 001066 BNE 99$
2407 023104 104401 023112 TYPE ,69$ ;;TYPE ASCIZ STRING
(1) 023110 000426 BR 68$ ;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <200>/SANITY TIMER SET FOR 16.384MS PER COUNT./<200>
(1) 68$:
2408 023166 000434 BR 99$
2409
2410 ;HERE ON ERROR
2411 023170 104044 12$: ERROR+ 44 ;CAN'T RESOLVE SANITY TIMER VALUE
2412 023172 004737 023526 JSR PC,DMPREG ;COUNTS NOT WITHIN LIMITS
2413 023176 000430 BR 99$
2414
2415 ;HERE ON SUBSEQUENT PASSES TO CHECK CONSISTENCY
2416 023200 013737 025026 001124 20$: MOV MAXCNT,$GDDAT ;GET EXPECTED MAX LIMIT FOR TYPEOJT.
2417 023206 013737 025022 024750 MOV STCCNT,DISPLY ;DISPLAY COUNT AS FOUND ON FIRST PASS.
2418 023214 023737 001126 001124 CMP $BDDAT,$GDDAT ;SEE THAT COUNT NOT EXCEEDED.
2419 023222 003404 BLE 21$ ;OK IF .LE MAX LIMIT.
2420 023224 104044 ERROR+ 44 ;COUNT GREATER THAN MAX OF RANGE.
2421 023226 004737 023526 JSR PC,DMPREG
2422 023232 000412 BR 99$
2423 023234 013737 025024 001124 21$: MOV MINCNT,$GDDAT ;GET EXPECTED MIN LIMIT.
2424 023242 023737 001126 001124 CMP $BDDAT,$GDDAT ;CHECK
2425 023250 002003 BGE 99$ ;OK IF ACTUAL .GE MIN LIMIT.
2426 023252 104044 ERROR+ 44 ;COUNT LESS THAN MIN OF RANGE.
2427 023254 004737 023526 JSR PC,DMPREG
2428
```

MAINDEC-11-CR11A-B MACY11 30A(1052) 18-JUN-79 14:32 PAGE 58-3  
CR11AB.P11 18-JUN-79 14:24 T57 DETERMINE SANITY TIMER COUNT RATE

SEQ 0082

2429	023260	012677	001456	99%:	MOV	(SP)+,@111V	:RESTORE VECTOR
2430	023264	012677	001454		MOV	(SP)+,@111P	:
2431	023270	012777	100000 001434		MOV	#CLR,@ACR	:INIT THE INTERFACE
2432							
2433							
2434	023276	000005			RESET		
2435	023300	000240			NOP		
2436	023302	000137	032052		JMP	\$ECP	

```

2438          .SBTTL TEST AND UTILITY SUBROUTINES
2439
2440          ;SET THE IIST TO INTERNAL LOOPBACK MAINTENENCE MODE USING
2441          ;THE SID VALUE POINTED TO BY R4
2442
2443 023306 005037 023446          SETMNT: CLR          MNTBTS          ;CLEAR EXTRA MNT BITS
2444 023312 012777 100000 001412 SETMNT1: MOV          #CLR,@ACR          ;RESET THE INTERFACE
2445 023320 016400 021762          MOV          SIDTAB(4),R0        ;GET SID VALUE
2446 023324 062700 000016          ADD          #TMT,R0          ;ADD LOOPBACK + BOOT DISABLE TO MNT. SID
2447 023330 063700 023446          ADD          MNTBTS,R0        ;ADD ANY EXTRA MNT BITS
2448 023334 027777 001372 001370  CMP          @ACR,@ACR          ;SHORT DELAY TO ALLOW RESET TO WORK
2449 023342 027777 001364 001362  CMP          @ACR,@ACR
2450 023350 012777 000015 001354  MOV          #MTC,@ACR          ;CHOOSE MTC
2451 023356 010077 001354          MOV          R0,@MTC ;LOAD MTC
2452 023362 012777 000001 001342  MOV          #PGCSE,@ACR        ;SELECT PGCS
2453 023370 017737 001342 024750  MOV          @PGCS,DISPLY        ;READ PGCS
2454 023376 032737 004000 024750  BIT          #PRDY,DISPLY        ;CHECK FOR RDY
2455 023404 001003          BNE          1$                ;SKIP IF SET
2456 023406 104044          ERROR+ 44                      ;DEVICE CLR DID NOT SET PGCS RDY
2457 023410 004737 023526          JSR          PC,DMPREG
2458 023414          1$:
2459 023414 012777 000005 001310  SETMNX: MOV          #PGFE,@ACR        ;SELECT PGF
2460 023422 012777 007400 001306  MOV          #7400,@PGF          ;CLEAR BOOT FLAGS
2461 023430 012777 007400 001300  MOV          #7400,@STF          ;CLEAR BOOT FLAGS
2462 023436 012777 000017 001272  MOV          #17,@DCF           ;CLEAR DCF FLAGS
2463 023444 000207          RTS          PC                ;EXIT
2464
2465 023446 000000          MNTBTS: 0
2466
2467 023450 012746 023414          CLEARX: MOV          #SETMNX,-(SP) ;GET INTERMEDIATE RETURN FOR FLAG CLEARING
2468
2469          ;ROUTINE TO INIT THE IIST, THEN DELAY A SHORT TIME
2470
2471 023454 012777 100000 001250  CLEAR:  MOV          #100000,@ACR    ;RESET THE INTERFACE
2472 023462 012746 000030          MOV          #30,-(SP)          ;GET A LOOP COUNTER
2473 023466 005316          1$:  DEC          (SP)
2474 023470 001376          BNE          1$
2475 023472 005726          TST          (SP)+
2476 023474 000207          RTS          PC
2477
2478          ;SET MAINTENENCE MODE SO THAT SANITY TIMER OPERATIONS ACT
2479          ;LIKE PROGRAM GENERATED OPERATIONS
2480
2481 023476 012737 004000 023446  SSTMNT: MOV          #MTYP,MNTBTS    ;SET TYP TO A 1
2482 023504 000702          BR          SETMNT1           ;EXIT THROUGH SETMNT

```

2484  
2485  
2486  
2487  
2488 023506 010046  
2489 023510 005001  
2490 023512 006300  
2491 023514 005501  
2492 023516 005700  
2493 023520 001374  
2494 023522 012600  
2495 023524 000207

;PARITY CALCULATION SUBROUTINE  
;COUNTS THE NUMBER OF 1 BITS IN R0 IN R1

PARCAL: MOV R0,-(SP)  
CLR R1  
1\$: ASL R0  
ADC R1  
TST R0  
BNE 1\$  
MOV (SP)+,R0  
RTS PC

```

2497          ;DUMP ALL REGISTERS ROUTINE
2498
2499 023526 032777 000001 155404 DMPREG: BIT    #BIT0,@SWR    ;TEST BIT0
2500 023534 001001          BNE    1$          ;SKIP IF SET
2501 023536 000207          RTS     PC          ;EXIT IF 0
2502 023540 104401 042002          1$:   TYPE   ,HDRMES
2503 023544 012737 000011 023642   MOV    #9.,TCR    ;DO 9 TIMES
2504 023552 017746 001154          MOV    @ACR,-(SP) ;:PUSH @ACR ON STACK
2505 023556 005077 001150          CLR    @ACR      ;:INIT ADDRESS
2506 023562          2$:
   (1) 023562 017746 001150          MOV    @ADR,-(SP) ;:SAVE @ADR FOR TYPEOUT
   (1) 023566 104402          TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2507 023570 104401 042112          TYPE   ,TWOSP
2508 023574 005337 023642          DEC    TCR      ;DONE 9?
2509 023600 001370          BNE    2$      ;DO 9
2510 023602 104401 042131          TYPE   ,CARET
2511 023606 012677 001120          MOV    (SP)+,@ACR ;:POP STACK INTO @ACR
2512 023612 005777 155322          TST    @SWR     ;CHECK FOR ERR HLT
2513 023616 100002          BPL    3$      ;SKIP IF 0
2514 023620 000000          HALT
2515 023622 104407          CKSWR
2516 023624 032777 001000 155306 3$:   BIT    #BIT9,@SWR ;CHECK FOR LOOP ON ERROR
2517 023632 001402          BEQ    4$      ;SKIP IF CLEAR
2518 023634 013716 001110          MOV    $LPERR,(SP) ;:IF LOOP ON ERROR, RETURN TO PRESET LOCATION
2519 023640 000207          4$:   RTS     PC          ;EXIT
2520
2521 023642 000000          TCR:   0
    
```

```

2523
2524 ;ROUTINE TO GET DEVICE PARAMETERS
2525
2526 GETPAR:
(1) 023644 104401 023652 TYPE .65$ ;:TYPE ASCIZ STRING
(1) 023650 000440 BR 64$ ;:GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <200>/DO YOU WANT TO CHANGE DEVICE PARAMETERS (CSR, ETC.)? [Y OR N]/
(1) 023752 64$:
2527 023752 104410 RDCHR
2528 023754 022627 000131 CMP (SP)+,#'Y ;SEE WHAT ANSWER IS.
2529 023760 001402 BEQ .+6 ;IF Y(ES), ASK FOR PARAMETERS.
2530 023762 000137 024504 JMP 3$ ;IF ANYTHING ELSE, ASSUME NO.
2531 023766 104401 023774 TYPE .67$ ;:TYPE ASCIZ STRING
(1) 023772 000444 BR 66$ ;:GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <200>/IN THE FOLLOWING THREE PRINTOUTS, TYPE THE NEW DATA OR 0 IF NO CHA
(1) 024104 66$:
2532 024104 104401 041706 10$: TYPE ,ASKDVA
2533 024110 013746 024732 MOV ACR,-(SP) ;:SAVE ACR FOR TYPEOUT
(1) 024114 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2534 024116 104401 042110 TYPE ,SPACE
2535 024122 104412 RDOCT
2536 024124 012600 MOV (SP)+,RO ;GET VALUE TYPED IN.
2537 024126 001402 BEQ 11$ ;IF ZERO, DON'T CHANGE.
2538 024130 010037 024732 MOV RO,ACR ;STORE NEW VALUE
2539 024134 023727 024732 160000 11$: CMP ACR,#160000 ;CHECK FOR LEGALITY
2540 024142 103760 BLO 10$ ;LOOP IF BAD
2541 024144 032737 000003 024732 BIT #3,ACR ;CHECK FOR MOD. 4
2542 024152 001354 BNE 10$ ;LOOP IF BAD
2543 024154 013737 024732 024734 MOV ACR,ACRH ;LOAD OTHER ADDRESS SLOTS
2544 024162 005237 024734 INC ACRH
2545 024166 013737 024734 024736 MOV ACRH,ADR
2546 024174 005237 024736 INC ADR
2547 024200 013737 024736 024740 MOV ADR,ADRH
2548 024206 005237 024740 INC ADRH
2549 024212 104401 041726 1$: TYPE ,ASKPIV
2550 024216 013746 024742 MOV IIIV,-(SP) ;:SAVE IIIV FOR TYPEOUT
(1) ;:1
(1) 024222 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 024224 006 .BYTE 6 ;:TYPE 6 DIGITS
(1) 024225 000 .BYTE 0 ;:SUPPRESS LEADING ZEROS
2551 024226 104401 042110 TYPE ,SPACE
2552 024232 104412 RDOCT
2553 024234 012600 MOV (SP)+,RO ;GET VALUE TYPED IN.
2554 024236 001402 BEQ 12$ ;IF 0, DON'T CHANGE.
2555 024240 010037 024742 MOV RO,IIIV ;LOAD NEW VALUE
2556 024244 023727 024742 002000 12$: CMP IIIV,#2000 ;CHECK FOR LEGALITY
2557 024252 103357 BHIS 1$ ;LOOP IF BAD
2558 024254 032737 000003 024742 BIT #3,IIIV ;CHECK FOR MOD. 4
2559 024262 001353 BNE 1$ ;LOOP IF BAD
2560 024264 013737 024742 024744 MOV IIIV,IIIP ;LOAD PIL
2561 024272 062737 000002 024744 ADD #2,IIIP ;WITH IIIV+2
2562 024300 104401 041761 2$: TYPE ,ASKBRL
2563 024304 013746 024746 MOV IIIL,-(SP) ;:SAVE IIIL FOR TYPEOUT
(1) ;:1
(1) 024310 104403 TYPOS ;:GO TYPE--OCTAL ASCII
(1) 024312 006 .BYTE 6 ;:TYPE 6 DIGITS
    
```

```

(1) 024313 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2564 024314 104401 042110 TYPE ,SPACE
2565 024320 104412 RDOCT
2566 024322 012600 MOV (SP)+,R0 ;GET VALUE TYPED IN.
2567 024324 001402 BEQ 21$ ;IF ZERO, DON'T CHANGE.
2568 024326 010037 024746 MOV R0,111L ;GET BR LEVEL
2569 024332 023727 024746 000004 21$: CMP 111L,#4 ;CHECK FOR LEGALITY
2570 024340 103757 BLO 2$ ;LOOP IF BAD
2571 024342 023727 024746 000007 CMP 111L,#7 ;CHECK
2572 024350 101353 BHI 2$ ;LOOP IF BAD
2573 024352 000137 024504 JMP 3$ ;JUMP OVER HELP MSG.
2574 024356 104401 024364 30$:
(1) 024362 000423 TYPE ,69$ ;;TYPE ASCIZ STRING
(1) ;:69$: BR 68$ ;;GET OVER THE ASCIZ
(1) .ASCIZ <200>/TEST CONFIGURATION-SELECT CODES ARE:/
(1) 68$:
2575 024432 012737 002762 001176 MOV #MTAB,$TMPO ;GET START OF MESSAGE POINTER TABLE.
2576 024440 104401 001225 31$: TYPE ,$CRLF ;OUTPUT NEWLINE
2577 024444 017737 154526 024454 MOV @TMPO,32$ ;GET ADDRESS OF TEXT LINE TO PRINT.
2578 024452 104401 TYPE
2579 024454 000000 32$: .WORD 0
2580 024456 062737 000002 001176 ADD #2,$TMPO ;BUMP POINTER
2581 024464 023727 001176 003004 CMP $TMPO,#CNFMAX ;AT END?
2582 024472 103762 BLO 31$ ;IF NOT, GO FOR MORE.
2583 024474 104401 001225 TYPE ,$CRLF ;EXTRA CRLF
2584 024500 104401 001225 TYPE ,$CRLF ;NEW LINE
2585 024504 3$:
(1) 024504 104401 024512 TYPE ,71$ ;;TYPE ASCIZ STRING
(1) 024510 000437 BR 70$ ;;GET OVER THE ASCIZ
(1) ;:71$: .ASCIZ <200>/ENTER TEST CONFIGURATION SELECTION (0-10) OR 111 FOR HELP: /
(1) 70$:
2586 024610 104412 RDOCT
2587 024612 012637 025020 MOV (SP)+,CONFIG ;GET VALUE
2588 024616 023737 025020 003004 CMP CONFIG,CNFMAX ;SEE IF VALID
2589 024624 101402 BLOS .+6
2590 024626 000137 024356 JMP 30$ ;IF NOT, DO HELP MSG.
2591 024632 000207 RTS PC ;EXIT
    
```



```

2593
2594 ;ROUTINE TO REPORT THE SELF ID OF THE IIST
2595
2596 024634 005737 001236 GETSID: TST $PASS ;PASS 1?
2597 024640 001401 BEQ 1$ ;SKIP IF YES
2598 024642 000207 RTS PC ;EXIT IF NOT
2599 024644 005737 000042 1$: TST @#42 ;ARE WE IN ACT?
2600 024650 001002 BNE 2$ ;IF YES, DON'T PRINT
2601 024652 104401 042115 TYPE ,DMPSID
2602 024656 012777 100000 000046 2$: MOV #CLR,@ACR ;CLEAR INTERFACE
2603 024664 017700 000042 MOV @ACR,RO ;GET SID
2604 024670 000300 SWAB RO ;INTO B2-0
2605 024672 042700 177770 BIC #177770,RO ;CLEAR B15-3
2606 024676 010037 024730 MOV RO,SIDNUM ;SAVE SELF ID NUM.
2607 024702 005737 000042 TST @#42 ;IN ACT
2608 024706 001007 BNE 3$ ;DON'T PRINT IF YES.
2609 024710 010046 MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
(1) ;;1
(1) 024712 104403 TYPOS ;;GO TYPE--OCTAL ASCII
(1) 024714 006 .BYTE 6 ;;TYPE 6 DIGITS
(1) 024715 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
2610 024716 104401 042131 TYPE ,CARET
2611 024722 104401 042131 TYPE ,CARET
2612 024726 000207 3$: RTS PC ;EXIT
2613
2614 024730 000000 SIDNUM: 0
  
```

```

2616 .SBTTL DEVICE ADDRESS PARAMETER BLOCK
2617
2618 024732 177500 ACR: 177500 ;ACCESS CONTROL REGISTER
2619 024734 177501 ACRH: 177501 ;HIGH BYTE OF ACR
2620
2621 024736 ADR: ;ACCESS DATA REGISTER
2622 024736 PGTE: ;PGM-GEN. XMIT ENABLES REGISTER
2623 024736 PGCS: ;PGM-GEN. CSR
2624 024736 STTE: ;SANITY TIMER XMIT ENABLES REGISTER
2625 024736 STCS: ;SANITY TIMER CSR
2626 024736 IMSK: ;INPUT MASKS REGISTER
2627 024736 PGF: ;PGM-GEN. I/B FALAGS REGISTER
2628 024736 STF: ;SANITY TIMER FLAGS REGISTER
2629 024736 DCF: ;DCLO/DISCONNECT FLAGS REGISTER
2630 024736 EXC: ;EXCEPTIONS REGISTER
2631 024736 177502 MTC: 177502 ;MAINTENANCE REGISTER
2632 024740 ADRH: ;ADR HIGH BYTE
2633 024740 PGCSH: ;PGCS HIGH BYTE
2634 024740 STCSH: ;STCS HIGH BYTE
2635 024740 IMSKH: ;IMSK HIGH BYTE
2636 024740 PGFH: ;PGF HIGH BYTE
2637 024740 DCFH: ;DCF HIGH BYTE
2638 024740 EXCH: ;EXC HIGH BYTE
2639 024740 177503 MTCH: 177503 ;MTC HIGH BYTE
2640 024742 000260 I1IV: 260 ;I1ST INTERRUPT VECTOR ADDRESS
2641 024744 000262 I1IP: 262 ;I1ST INTERRUPT PRIORITY ADDRESS
2642 024746 000006 I1IL: 6 ;I1ST INTERRUPT LEVEL
2643
2644
2645 ;CONSTANTS AND VARIABLES
2646
2647 024750 000000 DISPLY: 0
2648 024752 000000 TIMPER: 0
2649 024754 000000 LINMBR: 0
2650 024756 000020 SNAP0: .BLKW 16.
2651
2652 025016 000000 PARFLG: .WORD 0
2653
2654 025020 000000 CONFIG: .WORD 0
2655
2656 025022 000000 STCCNT: .WORD 0 ;COUNT OF # PG CHARACTERS TRANSMITTED DURING
2657 ;TWO COUNTS OF THE SANITY TIMER.
2658 025024 000000 MINCNT: .WORD 0 ;MINIMUM ALLOWED VALUE OF STCCNT.
2659 025026 000000 MAXCNT: .WORD 0 ;MAXIMUM ALLOWED VALUE OF STCCNT.

```

```

2661 .SBTTL MAINTENANCE-MODE SELF-BOOT TEST
2662
-2663 ;MAINTENANCE-MODE BOOT TEST.
2664 ;USES MAINTENANCE LOOPBACK TO GENERATE BOOT TO THIS CPU.
2665 ;RECEIVE CHANNEL AND TYPE (PG OR ST) SELECTED VIA OPERATOR INPUT.
2666 ; (0-3) FOR PG BOOTS, (10-13) FOR ST BOOTS.
2667 ;THIS TEST VERIFIES OPERATION OF THE BOOT-PULSE COMBINING LOGIC,
2668 ;THE BOOT ONE-SHOT, THE DRIVER, AND THE CABLE.
2669
2670 025030 000004 MMBOOT: SCOPE
2671 025032 004737 024634 JSR PC,GETSID ;GET SELF-ID
2672 025036 005227 177777 INC #-1 ;FIRST TIME?
2673 025042 001402 BEQ ;+6
2674 025044 000137 025476 JMP 1$ ;IF NOT, SKIP EXPLANATION.
2675
2676 025050 104401 025056 TYPE ,65$ ;;TYPE ASCIZ STRING
(1) 025054 000434 BR 64$ ;;GET OVER THE ASCIZ
(1) ;;65$: .ASCIZ <CRLF>/AFTER SELECTING THE OPERATING CODE, FOLLOWED BY (CR),/
(1) 64$:
2677 025146 104401 025154 TYPE ,67$ ;;TYPE ASCIZ STRING
(1) 025152 000431 BR 66$ ;;GET OVER THE ASCIZ
(1) ;;67$: .ASCIZ <CRLF>/ THE OPERATOR MUST VERIFY THAT THE PROPER BOOT/
(1) 66$:
2678 025236 104401 025244 TYPE ,69$ ;;TYPE ASCIZ STRING
(1) 025242 000426 BR 68$ ;;GET OVER THE ASCIZ
(1) ;;69$: .ASCIZ <CRLF>/ ACTION TOOK PLACE. OPERATING CODES ARE:/
(1) 68$:
2679 025320 104401 025326 TYPE ,71$ ;;TYPE ASCIZ STRING
(1) 025324 000431 BR 70$ ;;GET OVER THE ASCIZ
(1) ;;71$: .ASCIZ <CRLF>/ 0-3 SELECTS PROGRAMMED BOOT, RECEIVERS 0-3/
(1) 70$:
2680 025410 104401 025416 TYPE ,73$ ;;TYPE ASCIZ STRING
(1) 025414 000430 BR 72$ ;;GET OVER THE ASCIZ
(1) ;;73$: .ASCIZ <CRLF>/ 10-13 SELECTS SANITY BOOT, RECEIVERS 0-3/<CRLF>
(1) 72$:
2681
2682 025476 005777 153444 1$: TST @TKB ;FLUSH KEYBOARD
2683 025502 104401 025510 TYPE ,75$ ;;TYPE ASCIZ STRING
(1) 025506 000412 BR 74$ ;;GET OVER THE ASCIZ
(1) ;;75$: .ASCIZ <CRLF>/ENTER BOOT CODE: /
(1) 74$:
2684 025534 104412 RDOCT ;GET OCTAL INPUT
2685 025536 011637 024754 MOV (SP),LINMBR ;SAVE VALUE
2686 025542 012600 MOV (SP)+,RO ;AND A COPY
2687 025544 032700 177764 BIT #177764,RO ;IS IT A VALID CODE?
2688 025550 001427 BEQ 2$ ;IF YES, GO ON.
2689 025552 104401 025560 TYPE ,77$ ;;TYPE ASCIZ STRING
(1) 025556 000423 BR 76$ ;;GET OVER THE ASCIZ
(1) ;;77$: .ASCIZ <CRLF>/ILLEGAL CODE; MUST BE 0-3 OR 10-13/<CRLF>
(1) 76$:
2690 025626 000723 BR 1$ ;GO BACK FOR ANOTHER
2691
2692 025630 012737 025630 001110 2$: MOV #2$,SLPERR ;SET UP FOR ERROR LOOP
2693 025636 004737 023306 JSR PC,SETMNT ;INIT, THEN SETUP MAINT. MODE
2694 025642 012777 000015 177062 MOV #MICE,@ACR ;SELECT MTC
2695 025650 012777 000003 177060 MOV #MLEN+MDS,@MTC ;SET PROPER MODE.

```

```

2696 025656 013700 024754      MOV      LINMBR,R0      ;GET CODE
2697 025662 010037 024750      MOV      R0,DISPLY     ;MAKE COPY
2698 025666 042737 000010 024754  BIC      #10,LINMBR
2699 025674 042700 000010      BIC      #10,R0        ;MASK OUT TYPE BIT
2700 025700 006300      ASL      R0            ;MAKE WORD INDEX
2701 025702 016001 022002      MOV      PGBTAB(R0),R1 ;GET SELECTED BIT
2702 025706 005101      COM      R1            ;INVERT TO CONFIGURE MASKS.
2703 025710 012777 000004 177014  MOV      #IMSKE,@ACR   ;SELECT IMASK REG.
2704 025716 010177 177014      MOV      R1,@IMSK      ;SET ALL MASKS EXCEPT ONE.
2705 025722 010137 001176      MOV      R1,$TMP0      ;$TMP0 IS MASK VALUE
2706 025726 016037 022002 001124  MOV      PGBTAB(R0),$GDDAT ;BOOT ENABLE
2707 025734 013701 024730      MOV      SIDNUM,R1     ;GET SELF-ID.
2708 025740 006301      ASL      R1            ;MAKE WORD INDEX
2709 025742 032737 000010 024750  BIT      #10,DISPLY    ;WHAT TYPE OF BOOT?
2710 025750 001036      BNE      5$           ;BRANCH IF ST
2711
2712                                ;DO PROGRAMMED BOOT:
2713 025752 012777 000000 176752  MOV      #PGTEE,@ACR   ;SELECT PGTE REGISTER.
2714 025760 016177 022002 176750  MOV      PGBTAB(R1),@PGTE ;ENABLE BOOT TO SELF.
2715 025766 012777 000001 176742  MOV      #GO,@PGCS    ;INITIATE BOOT SEQUENCE.
2716 025774 005002      CLR      R2
2717 025776 062702 000001      ADD      #1,R2        ;STALL
2718 026002 001375      BNE      -4
2719 026004 012777 000010 176720  MOV      #EXCE,@ACR   ;SELECT EXC
2720 026012 017737 176720 001200  MOV      @EXC,$TMP1   ;GET EXC INTO $TMP1
2721 026020 012777 000005 176704  MOV      #PGFE,@ACR   ;SELECT PGF
2722 026026 017737 176704 001126  MOV      @PGF,$BDDAT  ;GET IT
2723 026034 104044      ERROR   +44          ;NO BOOT OCCURRED
2724 026036 004737 023526      JSR      PC,DMPREG
2725 026042 000137 025476      JMP      1$           ;TRY SOME MORE?
2726
2727                                ;DO SANITY-TIMER BOOT (AND HALT):
2728 026046 012777 000002 176656  5$: MOV      #STTEE,@ACR ;SELECT ST TRANSMIT ENABLES
2729 026054 016177 022002 176654  MOV      STBTAB(R1),@STTE ;SET TO XMIT TO SELF
2730 026062 012777 000005 176646  MOV      #LKE+ENB,@STCS ;ENABLE CLOCK AND HALT
2731 026070 005002      CLR      R2
2732 026072 005777 176634  6$: TST      @ACR        ;STALL
2733 026076 005777 176630      TST      @ACR
2734 026102 005777 176624      TST      @ACR
2735 026106 005777 176620      TST      @ACR
2736 026112 005202      INC      R2
2737 026114 001366      BNE      6$
2738 026116 012777 000010 176606  MOV      #EXCE,@ACR   ;SELECT EXC
2739 026124 017737 176606 001200  MOV      @EXC,$TMP1   ;GET IT
2740 026132 012777 000006 176572  MOV      #STFE,@ACR   ;SELECT STF
2741 026140 017737 176572 001126  MOV      @STF,$BDDAT  ;SELECT STF
2742 026146 104044      ERROR   +44          ;NO BOOT (OR EVEN HALT) OCCURRED
2743 026150 004737 023526      JSR      PC,DMPREG
2744 026154 000137 025476      JMP      1$           ;TRY FOR ANOTHER?
    
```

```

2746 .SBTTL MULTIPROCESSOR BOOT TEST
2747
2748 ;ALLOWS OPERATOR TO INITIATE BOOTS TO OTHER CPU'S.
2749 ;IF THE PROGRAM IN THE BOOTSTRAP ROM SETS THE DIP11-A
2750 ;INTERRUPT ENABLE BIT, THIS TEST SHOULD BE STARTED IN EACH
2751 ;CPU. THE BOOTED CPU CAN THEN BE INTERRUPTED TO REPORT THE FACT.
2752
2753 MPDIBT: SCOPE
2754 026160 000004 JSR PC,GETSID ;GET SELF-ID
2755 026162 004737 024634 MPDBT1: MOV #PR7,PSW ;LOCK OUT INTERRUPTS
2756 026166 012737 000340 177776 JSR PC,CLEARX ;INIT THE INTERFACE & FLAGS
2757 026200 012777 026714 176534 MOV #MPDINT,@I11V ;SET UP IIST VECTOR
2758 026206 012777 000340 176532 MOV #PR7,@I11L ;SET TO PRI 7 ON INTERRUPT
2759
2760 026214 012777 000001 176510 18: MOV #PGCSE,@ACR ;SELECT PGCS REGISTER
2761 026222 012777 000004 176506 MOV #IE,@PGCS ;SET INTR ENB TO CATCH ERRORS
2762 026230 012706 001100 MOV #STACK,SP ;SETUP STACK POINTER
2763 026234 005037 177776 CLR PSW ;ALLOW INTERRUPTS
2764 026240 005777 152702 TST @STKB ;FLUSH KBD BUFFER OF OLD STUFF
2765 026244 104401 026252 TYPE ,65$ ;TYPE ASCIZ STRING
(1) 026250 000415 BR 64$ ;GET OVER THE ASCIZ
(1) ;:65$: .ASCIZ <CRLF>/ENTER # OF CPU TO BOOT: /
(1) 026304 64$:
2766
2767 026304 005037 001124 10$: CLR $GDDAT ;EXPECT PGF AND STF TO BE CLEAR
2768 026310 012777 000005 176414 MOV #PGFE,@ACR ;SELECT PGF REGISTER
2769 026316 017737 176414 001126 MOV @PGF,$BDDAT ;GET PGF AND TEST IT
2770 026324 001411 BEQ 11$ ;OK IF 0 (WE SHOULDN'T SEE FLAGS HERE)
2771 026326 104044 ERROR +44 ;ERROR -- PGF FLAGS PRESENT BUT NO BOOTING
2772 026330 004737 023526 JSR PC,DMPREG
2773 026334 012777 000005 176370 MOV #PGFE,@ACR ;SELECT PGF AGAIN
2774 026342 012777 007417 176366 MOV #7417,@PGF ;CLEAR THE FLAGS
2775 026350 012777 000006 176354 11$: MOV #STFE,@ACR ;SELECT STF REGISTER
2776 026356 017737 176354 001126 MOV @STF,$BDDAT ;GET STF AND TEST IT
2777 026364 001411 BEQ 12$ ;OK IF 0
2778 026366 104044 ERROR +44 ;ERROR -- STF FLAGS PRESENT
2779 026370 004737 023526 JSR PC,DMPREG
2780 026374 012777 000006 176330 MOV #STFE,@ACR ;SELECT STF AGAIN
2781 026402 012777 007417 176326 MOV #7417,@STF ;CLEAR THE FLAGS
2782 026410 105777 152530 12$: TSTB @STKS ;LOOK FOR KBD INPUT
2783 026414 100333 BPL 10$ ;LOOP BACK IF NONE.
2784
2785 026416 104412 RDOCT ;GET OCTAL INPUT WHEN PRESENT
2786 026420 011637 024750 MOV (SP),DISPLY ;SAVE THE # OF CPU TO BE BOOTED
2787 026424 022627 000003 CMP (SP)+,#3 ;SEE IF IT'S LEGAL (0-3)
2788 026430 101427 BLOS 2$ ;BR IF YES - IF NOT, SAY SO
2789 026432 104401 026440 TYPE ,67$ ;TYPE ASCIZ STRING
(1) 026436 000423 BR 66$ ;GET OVER THE ASCIZ
(1) ;:67$: .ASCIZ <CRLF>/ILLEGAL CODE; MUST BE IN RANGE 0-3/<CRLF>
(1) 026506 66$:
2790 026506 000642 BR 18 ;GO ASK FOR IT AGAIN
2791
2792 ;COME HERE ON VALID INPUT FROM KEYBOARD.
2793
2794 026510 013700 024750 2$: MOV DISPLY,RO ;GET CPU SELECT CODE
2795 026514 006300 ASL RO ;MAKE IT A WORD INDEX

```

```

2796 026516 012777 000000 176206      MOV      #PGTEE,@ACR      ;SELECT PG XMIT ENABLES REGISTER
2797 026524 016077 022002 176204      MOV      PGTAB(R0),@PGTE ;SELECT CPU TO BOOT
2798 026532 052777 000001 176176      BIS      #GO,@PGCS      ;SEND THE BOOT
2799 026540 012702 000004          MOV      #4,R2          ;INITIALIZE OUTER COUNTER FOR STALL
2800 026544 005003          CLR      R3            ;ZERO THE INNER COUNTER
2801 026546 005777 176160      3$:     CLR      R3            ;ZERO THE INNER COUNTER
2802 026552 005777 176154      4$:     TST      @ACR      ;STALL TO ALLOW BOOTING TO OCCUR
2803 026556 005777 176150      TST      @ACR      ;AND INIT TO FINISH ON TARGET CPU.
2804 026562 005777 176144      TST      @ACR      ;EACH "TST @ACR" SHOULD BURN UP AT
2805 026566 005777 176140      TST      @ACR      ;LEAST 2 MICROSECONDS
2806 026572 005777 176134      TST      @ACR      ;TO ALLOW AT LEAST 4 SECONDS OF STALL.
2807 026576 005203          INC      R3            ;BUMP INNER COUNTER.
2808 026600 001362          BNE     4$            ;LOOP 'TIL IT GOES TO ZERO.
2809 026602 005302          DEC      R2            ;BUMP OUTER COUNTER
2810 026604 001357          BNE     3$            ;LOOP 'TIL IT COUNTS TO ZERO.
2811 026606 013700 024750      MOV      DISPLY,R0     ;GET TARGET CPU # AGAIN
2812 026612 006300          ASL     R0            ;MAKE WORD INDEX
2813 026614 012777 000000 176110      MOV      #PGTEE,@ACR  ;SELECT PGTE AGAIN
2814 026622 016077 021772 176106      MOV      PGITAB(R0),@PGTE ;SELECT CPU TO BE INTERRUPTED
2815 026630 052777 000001 176100      BIS      #GO,@PGCS    ;SEND THE INTERRUPT
2816 026636 104401 026644      TYPE    ,69$         ;:TYPE ASCIZ STRING
(1) 026642 000416      BR      68$         ;:GET OVER THE ASCIZ
(1)          ;:69$: .ASCIZ <CRLF>/BOOT & INTR SENT TO CPU # /
(1) 026700      68$:          MOV      DISPLY,-(SP) ;PRINT THE CPU.
2817 026700 013746 024750      MOV      DISPLY,-(SP) ;PRINT THE CPU.
2818 026704 104403          TYPOS
2819 026706 001 000          .BYTE 1,0
2820 026710 000137 026214      JMP     1$            ;GO FOR MORE.
2821
2822          ;COME HERE ON INTERRUPT
2823
2824 026714 012706 001100      MPDINT: MOV      #STACK,SP   ;REESTABLISH STACK, ASSUMING BOOT CHANGED IT.
2825 026720 012777 000010 176004      MOV      #EXCE,@ACR   ;SELECT EXC (EXCEPTIONS) REGISTER
2826 026726 005037 001124          CLR      $GDDAT      ;EXPECT NO UIF OR RTE FLAGS.
2827 026732 017737 176000 001126      MOV      @EXC,$BDDAT  ;GET AND TEST EXC
2828 026740 001411          BEQ     1$            ;OK, BRANCH IF 0.
2829 026742 104044          ERROR   +44         ;ERROR - EXC IS NONZERO.
2830 026744 004737 023526          JSR     PC,DMPREG
2831 026750 012777 000010 175754      MOV      #EXCE,@ACR   ;SELECT EXC AGAIN
2832 026756 012777 007417 175752      MOV      #7417,@EXC   ;CLEAR THE FLAGS
2833 026764 012777 000007 175740      1$:     MOV      #DCFE,@ACR  ;SELECT DCF REGISTER.
2834 026772 017737 175740 001126      MOV      @DCF,$BDDAT  ;GET DCF
2835 027000 013737 001126 001124      MOV      $BDDAT,$GDDAT ;COPY IT
2836 027006 042737 000017 001124      BIC     #17,$GDDAT    ;EXPECT NO DCF FLAGS, BUT BRKS OK.
2837 027014 023737 001126 001124      CMP     $BDDAT,$GDDAT ;CHECK THE DATA
2838 027022 001411          BEQ     2$            ;BRANCH IF OK
2839 027024 104044          ERROR   +44         ;ERROR - DCF FLAGS NONZERO
2840 027026 004737 023526          JSR     PC,DMPREG
2841 027032 012777 000007 175672      MOV      #DCFE,@ACR   ;SELECT DCF AGAIN
2842 027040 012777 000017 175670      MOV      #17,@DCF     ;CLEAR THE FLAGS.
2843
2844 027046 012777 000006 175656      2$:     MOV      #STFE,@ACR   ;SELECT SANITY-TIMER FLAGS REGISTER
2845 027054 005037 001124          CLR      $GDDAT      ;EXPECT NO ST FLAGS
2846 027060 017737 175652 001126      MOV      @STF,$BDDAT  ;GET AND TEST STF CONTENTS
2847 027066 001411          BEQ     3$            ;BR IF ZERO
2848 027070 104044          ERROR   +44         ;ERROR -- STF IS NOT ZERO
    
```

```

2849 027072 004737 023526          JSR    PC,DMPREG
2850 027076 012777 000006 175626    MOV    #STFE,@ACR      ;SELECT STF AGAIN
2851 027104 012777 007417 175624    MOV    #7417,@STF     ;CLEAR ALL STF FLAGS
2852
2853 027112 005037 001124          3$:   CLR    $GDDAT         ;EXPECT NO FLAGS IN PGTE (BOOTING SHOULD CLEAR)
2854 027116 012777 000000 175606    MOV    #PGTEE,@ACR    ;SELECT PGTE REGISTER
2855 027124 017737 175606 001126    MOV    @PGTE,$BDDAT   ;GET AND TEST PGTE
2856 027132 001403          BEQ    4$              ;BR IF 0
2857 027134 104044          ERROR  +44            ;ERROR -- PGTE NOT ZERO
2858 027136 004737 023526          JSR    PC,DMPREG
2859
2860 027142 012777 000002 175562    4$:   MOV    #STTEE,@ACR    ;SELECT SITE REGISTER
2861 027150 017737 175562 001126    MOV    @STTE,$BDDAT   ;GET SITE AND TEST FOR ZERO
2862 027156 001403          BEQ    5$              ;OK OF ZERO
2863 027160 104044          ERROR  +44            ;ERROR - SITE NOT ZERO
2864 027162 004737 023526          JSR    PC,DMPREG
2865
2866 027166 012777 000003 175536    5$:   MOV    #STCSE,@ACR    ;SELECT STCS REGISTER
2867 027174 017737 175536 001126    MOV    @STCS,$BDDAT   ;GET STCS AND TEST FOR 0
2868 027202 001403          BEQ    6$              ;OK IF ZERO
2869 027204 104044          ERROR  +44            ;ERROR - STCS NOT ZERO
2870 027206 004737 023526          JSR    PC,DMPREG
2871
2872 027212 012737 004014 001124    6$:   MOV    #IP+IE+PRDY,$GDDAT ;EXPECT BITS IN PGCS
2873 027220 013700 024730          MOV    SIDNUM,RO      ;GET SELF-ID
2874 027224 006300          ASL    RO              ;MAKE WORD INDEX
2875 027226 056037 021762 001124    BIS    SIDTAB(RO),$GDDAT ;INSERT SELF-ID INTO EXPECTED DATA
2876 027234 012777 000001 175470    MOV    #PGCSE,@ACR    ;SELECT PGCS REGISTER
2877 027242 017737 175470 001126    MOV    @PGCS,$BDDAT   ;GET PGCS
2878 027250 023737 001126 001124    CMP    $BDDAT,$GDDAT  ;CHECK THE RESULT
2879 027256 001403          BEQ    7$              ;BRANCH IF OK
2880 027260 104044          ERROR  +44            ;ERROR -- PGCS INCORRECT
2881 027262 004737 023526          JSR    PC,DMPREG
2882
2883 027266 012777 000001 175436    7$:   MOV    #PGCSE,@ACR    ;SELECT PGCS REGISTER
2884 027274 012777 100000 175434    MOV    #100000,@PGCS  ;CLEAR ERR AND IE
2885 027302 012777 000005 175422    MOV    #PGFE,@ACR    ;SELECT PGF REGISTER
2886 027310 017737 175422 024750    MOV    @PGF,DISPLY    ;GET PGF
2887 027316 005037 001126          CLR    $BDDAT
2888 027322 005037 001124          CLR    $GDDAT
2889 027326 113737 024751 001124    MOV    DISPLY+1,$GDDAT ;GET PG BOOT FLAGS
2890 027334 113737 024750 001126    MOV    DISPLY,$BDDAT  ;GET PG INTR FLAGS
2891 027342 023737 001126 001124    CMP    $BDDAT,$GDDAT  ;PG BOOT & INTR FLAGS SHOULD BE EQUAL
2892 027350 001403          BEQ    8$              ;BRANCH IF THEY ARE.
2893 027352 104044          ERROR  +44            ;ERROR -- PG BOOT & INTR FLAGS NOT SAME
2894 027354 004737 023526          JSR    PC,DMPREG
2895
2896 027360 013700 001124          8$:   MOV    $GDDAT,RO      ;GET BOOT FLAGS
2897 027364 001004          BNE    9$              ;BRANCH IF NONZERO
2898 027366 104044          ERROR  +44            ;ERROR -- NO BOOT FLAGS
2899 027370 004737 023526          JSR    PC,DMPREG
2900 027374 000427          BR     12$             ;CAN' REPORT BOOT
2901
2902 027376 005046          9$:   CLR    -(SP)          ;CLEAR COUNTER
2903 027400 000241          10$:  CLC                    ;CLEAR C-BIT IN PREP. FOR ROTATE
2904 027402 006000          ROR    RO              ;SHIFT FLAGS INTO C-BIT

```

2905	027404	103402				BCS	11\$		:FOUND IT IF C SET
2906	027406	005216				INC	(SP)		:IF NOT, GO AGAIN
2907	027410	000773				BR	10\$		
2908	027412				11\$:				
(1)	027412	104401	027420			TYPE	.65\$		::TYPE ASCIZ STRING
(1)	027416	000412				BR	64\$		::GET OVER THE ASCIZ
(1)					::65\$:	.ASCIZ	<CRLF>/BOOTED FROM CPU #/		
(1)	027444				64\$:				
2909	027444	104403				TYPOS			:OUTPUT THE CPU #
2910	027446	001	000			.BYTE	1,0		
2911	027450	104401	001225			TYPE	,SCLF		:AND A NEWLINE
2912	027454	012777	000005	175250	12\$:	MOV	#PGFE,@ACR		:SELECT PGF
2913	027462	012777	007417	175246		MOV	#7417,@PGF		:CLEAR FLAGS
2914	027470	012706	001100			MOV	#STACK,SP		
2915	027474	000137	026166			JMP	MPDBT1		



```
2917          .SBTTL  SANITY-TIMER TIMEOUT HALT/FREEZE TEST
2918
2919          ;ALLOWS OPERATOR TO START THE SANITY TIMER AND VERIFY IF HALT OCCURS.
2920
2921 027500 000004          STHLT:  SCOPE
2922 027502 004737 024634          JSR    PC,GETSID          ;GET SELF-ID
2923 027506 004737 023450          2$:   JSR    PC,CLEARX        ;INIT
2924 027512 005777 151430          TST   @STKB           ;FLUSH KEYBOARD OF OLD STUFF.
2925 027516 104401 027524          TYPE  ,65$           ;;TYPE ASCIZ STRING
(1) 027522 000434          BR    64$            ;;GET OVER THE ASCIZ
(1)          ;;65$:  .ASCIZ <CRLF>/TYPE ANY CHARACTER WHEN READY TO INITIATE HALT ACTION;/
(1) 027614          64$:   TYPE  ,67$           ;;TYPE ASCIZ STRING
2926 027614 104401 027622          BR    66$            ;;GET OVER THE ASCIZ
(1) 027620 000425          ;;67$:  .ASCIZ <CRLF># THEN VERIFY THAT CPU IS HALTED/FROZEN.#
(1) 027674          66$:   RDCHR          ;GET INPUT
2927 027674 104410          MOV   #STACK,SP
2928 027676 012706 001100          MOV   #STCSE,@ACR    ;SELECT STCS REGISTER
2929 027702 012777 000003 175022          MOV   #LKE+ENB,@STCS ;SET LOCK-ENABLE AND START CLOCK
2930 027710 012777 000005 175020          CLR   R0             ;INIT STALL COUNTER
2931 027716 005000          1$:   TST   @ACR
2932 027720 005777 175006          TST   @ACR
2933 027724 005777 175002          TST   @ACR
2934 027730 005777 174776          INC   R0             ;STALL
2935 027734 005200          BNE  1$             ;LOOP
2936 027736 001370
2937
2938          ;ERROR IF WE FALL THROUGH -- NO HALT OCCURRED
2939          ; OR MAYBE OPERATOR JUST HIT "CONTINUE"
2940 027740 104044          ERROR +44
2941 027742 004737 023526          JSR   PC,DMPREG
2942 027746 000657          BR   2$
```

```

2944          .SBTTL  MANUAL CONTROL-PANEL TEST
2945
2946          ;TEST MANUAL CONTROL PANEL (INHIBIT INPUTS) VIA DIALOG.
2947          ;FOR TIME BEING, WE JUST PRINT OUT DCF REGISTER WHEN IT CHANGES.
2948
2949 027750 000004          MCP:  SCOPE
2950 027752 004737 023450      JSR    PC,CLEARX          ;INIT THE INTERFACE
2951 027756 004737 024634      JSR    PC,GETSID         ;SET SELF-ID
2952 027762 104401 027770      TYPE   ,65$            ;;TYPE ASCIZ STRING
      (1) 027766 000415          BR     64$              ;;GET OVER THE ASCIZ
      (1)                          ;;65$: .ASCIZ <CRLF>/INITIAL DCF CONTENTS = /
      (1)                          64$:
2953 030022 012777 000007 174702  MOV    #DCFE,@ACR        ;SELECT DCF
2954 030030 017737 174702 024750  MOV    @DCF,DISPLY       ;GET INITIAL CONTENTS OF DCF
2955 030036 013746 024750      MOV    DISPLY,-(SP)
2956 030042 104402          TYPOC          ;TYPE IT OUT.
2957
2958 030044 012777 000007 174660 1$:  MOV    #DCFE,@ACR        ;SELECT DCF
2959 030052 017737 174660 001124  MOV    @DCF,$GDDAT       ;GET IT
2960 030060 023737 001124 024750  CMP    $GDDAT,DISPLY     ;COMPARE WITH OLD VALUE
2961 030066 001766          BEQ    1$                ;JUST LOOK AGAIN IF EQUAL
2962
2963 030070 104401 030076          TYPE   ,67$            ;;TYPE ASCIZ STRING
      (1) 030074 000411          BR     66$              ;;GET OVER THE ASCIZ
      (1)                          ;;67$: .ASCIZ <CRLF>/DCF CHANGED TO /
      (1)                          66$:
2964 030120 013746 001124          MOV    $GDDAT,-(SP)
2965 030124 104402          TYPOC          ;PRINT IT
2966 030126 012777 000007 174576  MOV    #DCFE,@ACR        ;SELECT DCF
2967 030134 012777 000017 174574  MOV    #17,@DCF         ;CLEAR FLAGS
2968 030142 013737 001124 024750  MOV    $GDDAT,DISPLY     ;MAKE NEW OLD
2969 030150 042737 000017 024750  BIC    #17,DISPLY
2970 030156 000732          BR     1$

```

```

2972          .SBTTL  SCOPE LOOPS
2973
2974          ;ROUTINES FOR CYCLING THE LOGIC REPEATEDLY.
2975
2976 030160 000004          SLOOPS: SCOPE
2977 030162 004737 024634      JSR    PC,GETSID      ;GET SELF-ID INTO SIDNUM
2978 030166 005227 177777      INC    #-1          ;FIRST TIME SINCE LOADING?
2979 030172 001402          BEQ    1$           ;IF YES, PRINT EXPLANATIONS.
2980 030174 000137 031232      JMP    SLOOP1      ;IF NO, JUST GO ASK FOR WHAT TO DO.
2981
2982 030200 005777 150742      1$:   TST    @STKB      ;FLUSH KEYBOARD OF STUFF HANGING AROUND.
2983 030204 104401 030214      TYPE  ,SLMSG      ;TYPE MESSAGE.
2984 030210 000137 031232      JMP    SLOOP1      ;NOW GO SEE WHAT TO DO
2985
2986 030214 052200 042510 051440  SLMSG: .ASCII <CRLF>/THE SCOPE LOOPS ARE USED FOR CYCLING THE/
      030222 047503 042520 046040
      030230 047517 051520 040440
      030236 042522 052440 042523
      030244 020104 047506 020122
      030252 054503 046103 047111
      030260 020107 044124 105
2987 030265 040 047514 044507      .ASCII / LOGIC WHILE MONITORING WITH AN/<200>/ OSCILLOSCOPE./
      030272 020103 044127 046111
      030300 020105 047515 044516
      030306 047524 044522 043516
      030314 053440 052111 020110
      030322 047101 020200 051517
      030330 044503 046114 051517
      030336 047503 042520 056
2988 030343 200 042523 042514      .ASCII <CRLF>/SELECT THE LOOP FROM THE FOLLOWING MENU BY TYPING/
      030350 052103 052040 042510
      030356 046040 047517 020120
      030364 051106 046517 052040
      030372 042510 043040 046117
      030400 047514 044527 043516
      030406 046440 047105 020125
      030414 054502 052040 050131
      030422 047111 107
2989 030425 200 044440 051524      .ASCII <200>/ ITS OCTAL CODE FOLLOWED BY CARRIAGE RETURN./
      030432 047440 052103 046101
      030440 041440 042117 020105
      030446 047506 046114 053517
      030454 042105 041040 020131
      030462 040503 051122 040511
      030470 042507 051040 052105
      030476 051125 027116
2990 030502 052200 042510 043040      .ASCII <CRLF>/THE FOLLOWING OPERATIONS AND THEIR RESPECTIVE CODES ARE AVAILABLE
      030510 046117 047514 044527
      030516 043516 047440 042520
      030524 040522 044524 047117
      030532 020123 047101 020104
      030540 044124 044505 020122
      030546 042522 050123 041505
      030554 044524 042526 041440
      030562 042117 051505 040440
      030570 042522 040440 040526
  
```

```

030576 046111 041101 042514
030604 072
2991 030605 200 020200 030040 .ASCII <CRLF><CRLF>/ 0 GENERATE BOOT PULSES (AT J2 PIN B AND J3 PIN 2)/
030612 020040 042507 042516
030620 040522 042524 041040
030626 047517 020124 052520
030634 051514 051505 024040
030642 052101 045040 020062
030650 044520 020116 020102
030656 047101 020104 031512
030664 050040 047111 031040
030672 051
2992 030673 200 020040 020061 .ASCII <CRLF>/ 1 TRANSMIT PG INTERRUPTS TO SELF ON IIST BUS/
030700 052040 040522 051516
030706 044515 020124 043520
030714 044440 052116 051105
030722 052522 052120 020123
030730 047524 051440 046105
030736 020106 047117 044440
030744 051511 020124 052502
030752 123
2993 030753 200 020040 020062 .ASCII <CRLF>/ 2 TRANSMIT ST INTERRUPTS TO SELF ON IIST BUS/
030760 052040 040522 051516
030766 044515 020124 052123
030774 044440 052116 051105
031002 052522 052120 020123
031010 047524 051440 046105
031016 020106 047117 044440
031024 051511 020124 052502
031032 123
2994 031033 200 020040 020063 .ASCII <CRLF>/ 3 TRANSMIT PG INTERRUPTS TO ALL IIST'S/
031040 052040 040522 051516
031046 044515 020124 043520
031054 044440 052116 051105
031062 052522 052120 020123
031070 047524 040440 046114
031076 044440 051511 023524
031104 123
2995 031105 200 020040 020064 .ASCII <CRLF>/ 4 TRANSMIT PG BOOTS TO ALL IIST'S/
031112 052040 040522 051516
031120 044515 020124 043520
031126 041040 047517 051524
031134 052040 020117 046101
031142 020114 044511 052123
031150 051447
2996 031152 100200 047524 043440 .ASCII2 <CRLF><CRLF>/TO GET OUT OF THE LOOP, TYPE ANY CHARACTER./<CRLF>
031160 052105 047440 052125
031166 047440 020106 044124
031174 020105 047514 050117
031202 020054 054524 042520
031210 040440 054516 041440
031216 040510 040522 052103
031224 051105 100056 000
2997 031232 .EVEN
2998
2999 031232 005777 147710 SLOOP1: TST @STKE ;FLUSH KEYBOARD
    
```

```

3000 031236 104401 031244      TYPE      ,65$      ;;TYPE ASCIZ STRING
(1) 031242 000423      BR      64$      ;;GET OVER THE ASCIZ
(1) 031312      ;;65$: .ASCIZ <CRLF>/ENTER CODE FOR DESIRED LOOP (0-4): /
(1) 64$:      RDOCT      ;GET INPUT
3001 031312 104412      MOV      (SP)+,R0      ;GET VALUE
3002 031314 012600      CMP      R0,#4      ;SEE IF LEGAL
3003 031316 020027 000004      BLOS     1$      ;IF 0-4, THEN OK, GO DO IT.
3004 031322 101426      TYPE      ,67$      ;;TYPE ASCIZ STRING
3005 031324 104401 031332      BR      66$      ;;GET OVER THE ASCIZ
(1) 031330 000422      ;;67$: .ASCIZ /ILLEGAL CODE; MUST BE IN RANGE 0-4/
(1) 66$:      BR      SLOOP1      ;GO ASK AGAIN.
3006 031376 000715      1$:      ASL      R0      ;MAKE WORD INDEX.
3007 031400 006300      JMP      @SLDISP(R0) ;JUMP VIA DISPATCH TABLE.
3008 031402 000170 031406      ;DISPATCH TABLE
3009 031402 000170 031406      SLDISP: SCBOOT ;0 BOOT PULSES
3010 031406 031420      SCPI$F ;1 PG INTERRUPTS TO SELF
3011 031410 032000      SCST$F ;2 ST INTERRUPTS TO SELF
3012 031412 031554      SCPIAL ;3 PG INTERRUPTS TO ALL
3013 031414 032026      SCPBAL ;4 PG BOOTS TO ALL
3014 031416 032040
3015 031416 032040
3016 031416 032040
3017
3018
3019      ;BOOT PULSES (ASSUMES BOOT CABLE IS DISCONNECTED)
3020
3021 031420 004737 023450      SCBOOT: JSR      PC,CLEARX ;INIT THE INTERFACE
3022 031424 012777 000015 173300      MOV      #MTC,@ACR ;SELECT MTC
3023 031432 012777 000002 173276      MOV      #MLN,@MTC ;SET MAINT. LOOPBACK
3024 031440 005000      CLR      R0 ;R0 CYCLES THE MASKS SO EACH RECEIVER OPERATES IN TURN.
3025
3026 031442 010001      1$:      MOV      R0,R1 ;GET MASK CYCLE CONTROL
3027 031444 006301      ASL      R1 ;MAKE WORD INDEX
3028 031446 042701 177770      BIC      #177770,R1 ;ZAP JUNK
3029 031452 016102 022002      MOV      PGBTAB(R1),R2 ;GET BIT TO ENABLE
3030 031456 005102      COM      R2 ;SET ALL OTHER MASKS
3031 031460 012777 000004 173244      MOV      #IM$KE,@ACR ;SELECT IM$K REGISTER
3032 031466 010277 173244      MOV      R2,@IM$K ;LOAD MASKS, LEAVING ONE CLEAR
3033 031472 013701 024730      MOV      SIDNLM,R1 ;GET SELF-ID
3034 031476 006301      ASL      R1 ;MAKE WORD INDEX
3035 031500 012777 000000 173224      MOV      #PGTEE,@ACR ;SELECT PGTE
3036 031506 016177 022002 173222      MOV      PGBTAB(R1),@PGTE ;SET TO XMIT TO SELF
3037 031514 012777 000001 173214      MOV      #GO,@PGLS ;TRANSMIT
3038 031522 012701 000040      MOV      #40,R1 ;INIT STALL COUNTER
3039 031526 013702 024732      MOV      ACR,R2 ;GET ACR ADDRESS
3040 031532 005712      2$:      TST      (R2) ;STALL
3041 031534 005301      DEC      R1
3042 031536 001375      BNE      2$
3043 031540 005200      INC      R0 ;WHEN DONE, CYCLE MASK
3044 031542 105777 147376      TSTB     @STKS ;SEE IF KBD ACTIVE
3045 031546 100335      BPL      1$ ;IF NO, KEEP LOOPING
3046 031550 000137 031232      JMP      SLOOP1 ;IF YES, EXIT LOOP
3047
3048
3049      ;SEND REAL SANITY-TIMER INTERRUPTS TO SELF:

```

```

3050 031554 004737 023450          SCSTSF: JSR    PC,CLEARX      ;INIT THE INTERFACE
3051 031560 013700 024736          1$:    MOV    ADR,R0        ;GET ADR ADDRESS
3052 031564 114001                   MOV    -(R0),R1        ;BUMP ADDRESS AND GET SELF-ID
3053 031566 006301                   ASL    R1              ;MAKE WORD INDEX
3054 031570 112740 000002          MOV    #STEE,-(R0)     ;SELECT SITE REGISTER
3055 031574 016160 021772 000002  MOV    STTAB(R1),2(R0) ;SET ENABLE TO XMIT TO SELF
3056 031602 012760 000011 000002  MOV    #TMO+ENB,2(R0) ;CLEAR TMO, ENABLE CLOCK FOR 1 COUNT.
3057 031610 012701 023420          MOV    #10000.,R1     ;SET DELAY COUNTER
3058
3059 031614 105777 147324          2$:    TSTB   @STKS      ;KEYBOARD ACTIVE?
3060 031620 100002                   BPL    .+6            ;IF NOT, SKIP
3061 031622 000137 031232          JMP    SLOOP1        ;IF YES, EXIT LOOP
3062 031626 012710 000003          MOV    #STCSE,(R0)   ;SELECT STCS
3063 031632 032760 000010 000002  BIT    #TMO,2(R0)    ;TMO SET?
3064 031640 001003                   BNE    3$            ;SKIP IF YES
3065 031642 005301                   DEC    R1             ;IF NOT, JUST WAIT
3066 031644 001363                   BNE    2$            ;BUT NOT TOO LONG
3067 031646 000742                   BR     SCSTSF        ;NO TIMEOUT -- GO TO INIT
3068
3069 031650 012701 000040          3$:    MOV    #40,R1     ;SET TO STALL FOR XMIT
3070
3071 031654 005301                   4$:    DEC    R1       ;STALL
3072 031656 001376                   BNE    4$
3073 031660 000737                   BR     1$            ;DO IT AGAIN
3074
3075
3076                                ;GENERAL TRANSMIT ROUTINE:
3077 031662 000000          XMITS: .WORD 0        ;BIT TO TRANSMIT
3078
3079 031664 004737 023450          SCPXMT: JSR    PC,CLEARX      ;INIT THE INTERFACE
3080 031670 012777 000000 173034  MOV    #PGTEE,@ACR    ;SELECT THE ACR
3081 031676 013700 031662          MOV    XMITS,R0      ;GET BITS TO TRANSMIT
3082 031702 004737 023506          JSR    PC,PARCAL     ;CALCULATE PARITY
3083 031706 006301                   ASL    R1             ;POSITION PARITY BIT
3084 031710 052701 177774          BIS    #177774,R1
3085 031714 005101                   COM    R1             ;PARITY & GO
3086 031716 012777 000000 173006 1$:    MOV    #PGTEE,@ACR    ;SELECT PGTE
3087 031724 013777 031662 173004  MOV    XMITS,@PGTE   ;LOAD XMITS
3088 031732 010177 173000          MOV    R1,@PGCS     ;SEND
3089 031736 012702 000020          MOV    #20,R2       ;LOOP COUNTER IN CASE OF HANG
3090 031742 105777 147176          2$:    TSTB   @STKS      ;KEYBOARD ACTIVE?
3091 031746 100002                   BPL    3$            ;CONTINUE OF NOT
3092 031750 000137 031232          JMP    SLOOP1        ;EXIT LOOP IF YES
3093 031754 012777 000001 172750 3$:    MOV    #PGCSE,@ACR   ;READY?
3094 031762 032777 004000 172746  BIT    #PRDY,@PGCS   ;IF YES, DO AGAIN
3095 031770 001352                   BNE    1$
3096 031772 005302                   DEC    R2            ;IF NOT, COUNT DOWN
3097 031774 001362                   BNE    2$            ;LOOP
3098 031776 000732                   BR     SCPXMT        ;NO PRDY
3099
3100
3101                                ;PG INTERRUPTS TO SELF:
3102 032000 017701 172726          SCPISF: MOV    @ACR,R1    ;GET SELF-ID
3103 032004 000301                   SWAB   R1
3104 032006 006301                   ASL    R1
3105 032010 042701 177770          BIC    #177770,R1
    
```

```
3106 032014 016137 021772 031662      MOV    PGITAB(R1),XMIT5 ;GET BIT TO TRANSMIT
3107 032022 000137 031664              JMP    SCPXMT           ;GO DO IT
3108
3109                                     ;PG INTERRUPTS TO ALL:
3110 032026 012737 000017 031662  SCPIAL: MOV    #17,XMIT5 ;SET ALL INTR XMIT ENABLES
3111 032034 000137 031664              JMP    SCPXMT
3112
3113                                     ;PG BOOTS TO ALL:
3114 032040 012737 007400 031662  SCPBAL: MOV    #7400,XMIT5 ;SET ALL BOOT XMIT ENABLE
3115 032046 000137 031664              JMP    SCPXMT           ;GO TO MAIN LOOP
```





```

(1)                               ;; OTHERWISE CONTINUE
(1) 032176 021627 001002          CMP      (SP),#1002          ;; UNEXPECTED TRAP OR INTERRUPT
(1) 032202 101002                  BHI      1$                ;; ARE TRAPPED HERE VIA IOT
(1) 032204 0G0137 032446          JMP      $ERROR           ;; GO PROCESS UNEXPECTED TRAP
(1) 032210 032777 040000 146722 1$: BIT      #BIT14,@SWR        ;; LOOP ON PRESENT TEST?
(1) 032216 001104                  BNE      $OVER           ;; YES IF SW14=1
(1)                               ;##### START OF CODE FOR THE XOR TESTER#####
(1) 032220 000416                  $XTSTR: BR      6$        ;; IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)                               ;; THIS INSTRUCTION TO A 'NOP' (NOP-240)
(1) 032222 013746 000004          MOV      @#ERRVEC,-(SP)   ;; SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 032226 012737 032246 000004  MOV      #5$,@#ERRVEC    ;; SET FOR TIMEOUT
(1) 032234 005737 177060          TST      @#177060        ;; TIME OUT ON XOR?
(1) 032240 012637 000004          MOV      (SP)+,@#ERRVEC  ;; RESTORE THE ERPOR VECTOR
(1) 032244 000453                  BR       $SVLAD          ;; GO TO THE NEXT TEST
(1) 032246 022626                  5$:  CMP      (SP)+,(SP)+  ;; CLEAR THE STACK AFTER A TIME OUT
(1) 032250 012637 000004          MOV      (SP)+,@#ERRVEC  ;; RESTORE THE ERROR VECTOR
(1) 032254 000413                  BR       7$             ;; LOOP ON THE PRESENT TEST
(1) 032256                  6$: ;##### END OF CODE FOR THE XOR TESTER#####
(1) 032256 105737 001103          2$:  TSTB     $ERFLG      ;; HAS AN ERROR OCCURRED?
(1) 032262 001421                  BEQ      3$             ;; BR IF NO
(1) 032264 123737 001115 001103  CMPB     $ERMAX,$ERFLG   ;; MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 032272 101015                  BHI      3$             ;; BR IF NO
(1) 032274 032777 001000 146636  BIT      #BIT09,@SWR    ;; LOOP ON ERROR?
(1) 032302 001404                  BEQ      4$             ;; BR IF NO
(1) 032304 013737 001110 001106  7$:  MOV      $LPERR,$LPADR  ;; SET LOOP ADDRESS TO LAST SCOPE
(1) 032312 000446                  BR       $OVER
(1) 032314 105037 001103          4$:  CLRB     $ERFLG      ;; ZERO THE ERROR FLAG
(1) 032320 005037 001214          CLR      $TIMES         ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1) 032324 000415                  BR       1$             ;; ESCAPE TO THE NEXT TEST
(1) 032326 032777 004000 146604  3$:  BIT      #BIT11,@SWR    ;; INHIBIT ITERATIONS?
(1) 032334 001011                  BNE      1$             ;; BR IF YES
(1) 032336 005737 001236          TST      $PASS          ;; IF FIRST PASS OF PROGRAM
(1) 032342 001406                  BEQ      1$             ;; INHIBIT ITERATIONS
(1) 032344 005237 001104          INC      $ICNT          ;; INCREMENT ITERATION COUNT
(1) 032350 023737 001214 001104  CMP      $TIMES,$ICNT   ;; CHECK THE NUMBER OF ITERATIONS MADE
(1) 032356 002024                  BGE      $OVER         ;; BR IF MORE ITERATION REQUIRED
(1) 032360 012737 000001 001104  1$:  MOV      #1,$ICNT     ;; REINITIALIZE THE ITERATION COUNTER
(1) 032366 013737 032444 001214  MOV      $MXCNT,$TIMES  ;; SET NUMBER OF ITERATIONS TO DO
(1) 032374 105237 001102          $SVLAD: INCB    $STNM     ;; COUNT TEST NUMBERS
(1) 032400 113737 001102 001234  MOVB    $STNM,$TESTN    ;; SET TEST NUMBER IN APT MAILBOX
(1) 032406 011637 001106          MOV      (SP),$LPADR    ;; SAVE SCOPE LOOP ADDRESS
(1) 032412 011637 001110          MOV      (SP),$LPERR    ;; SAVE ERROR LOOP ADDRESS
(1) 032416 005037 001216          CLR      $ESCAPE       ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 032422 112737 000001 001115  MOVB    #1,$ERMAX      ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 032430 013777 001102 146504  $OVER: MOV      $STNM,@DISPLAY ;; DISPLAY TEST NUMBER
(1) 032436 013716 001106          MOV      $LPADR,(SP)   ;; FUDGE RETURN ADDRESS
(1) 032442 000002                  RTI                    ;; FIXES PS
(1) 032444 000001          $MXCNT: 1              ;; MAX. NUMBER OF ITERATIONS

```

3120

.SBTTL ERROR HANDLER ROUTINE

```

(1)
(2)
(1) ;*****
(1) ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
(1) ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
(1) ;*AND GO TO $ERRTYP ON ERROR
(1) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1) ;*SW15=1 HALT ON ERROR

```

```
(1) ;*SW13=1          INHIBIT ERROR TYPEOUTS
(1) ;*SW10=1          BELL ON ERROR
(1) ;*SW09=1          LOOP ON ERROR
(1) ;*CALL
(1) ;*              ERROR N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
(1) $ERROR:
(1) 032446          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(1) 032446 104407 7$: INCB $ERFLG      ;;SET THE ERROR FLAG
(1) 032450 105237 001103 BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
(1) 032454 001775 MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 032456 013777 001102 146456 BIT #BIT10,@SWR ;;BELL ON ERROR?
(1) 032464 032777 002000 146446 BEQ 1$          ;;NO - SKIP
(1) 032472 001402 TYPE ,SBELL ;;RING BELL
(1) 032474 104401 001220 INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
(1) 032500 005237 001112 1$: MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 032504 011637 001116 SUB #2,$ERRPC
(1) 032510 162737 000002 001116 MOV #2,$ERRPC
(1) 032516 117737 146374 001114 MOV #2,$ERRPC ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 032524 032777 020000 146406 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
(1) 032532 001055 BNE 20$          ;;SKIP TYPEOUTS
(1) 032534 021627 001002 CMP (SP),#1002 ;;IF RETURN PC LESS THAN 1002
(1) 032540 101046 BHI 12$          ;;ERROR IS ILLEGAL TRAP
(1) ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
(1) 032542 016637 000004 001116 MOV 4(SP),$ERRPC ;;GET PC AT TIME OF FALSE TRAP
(1) 032550 162737 000002 001116 SUB #2,$ERRPC ;;ADJUST PC
(1) 032556 104401 032622 TYPE ,10$          ;;TYPE HEADER
(2) 032562 013746 001116 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 032566 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032570 104401 032630 TYPE ,11$
(1) 032574 162716 000004 SUB #4,(SP) ;;GET FALSE TRAP VECTOR ADDR
(1) 032600 011637 001116 MOV (SP),$ERRPC
(2) 032604 013746 001116 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 032610 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 032612 104401 001225 TYPE ,SCRLF
(1) 032616 022626 CMP (SP)+,(SP)+ ;;POP FALSE TRAP VECTOR PC&ADDR
(1) 032620 000422 BR 20$
(1) 032622 050200 036503 000040 10$: .ASCIZ <200>'PC= '
(1) 032630 020040 047125 054105 11$: .ASCIZ ' UNEXPECTED TRAP TO '
(1) 032636 042520 052103 042105
(1) 032644 052040 040522 020120
(1) 032652 047524 000040
(1) .EVEN
(1) 032656          12$: JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 032656 004737 032770 TYPE ,SCRLF
(1) 032662 104401 001225
(1) 032666          20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 032666 122737 000001 001250 BNE 2$          ;;NO,SKIP APT ERROR REPORT
(1) 032674 001007 MOV #1,$ITEMB ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 032676 113737 001114 032710 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
(1) 032704 004737 035732
(1) 032710 000          21$: .BYTE 0
(1) 032711 000          .BYTE 0
(1) 032712 000777          22$: BR 22$          ;;APT ERROR LOOP
(1) 032714 005777 146220 2$: TST @SWR ;;HALT ON ERROR
(1) 032720 100002 BPL 3$          ;;SKIP IF CONTINUE
(1) 032722 000000 HALT          ;;HALT ON ERROR!
```

```
(1) 032724 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
(1) 032726 032777 001000 146204 3$: BIT      #BIT09,@SWR  ;;LOOP ON ERROR SWITCH SET?
(1) 032734 001402          BEQ      4$          ;;BR IF NO
(1) 032736 013716 001110          MOV      $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
(1) 032742 005737 001216          4$: TST      $ESCAPE  ;;CHECK FOR AN ESCAPE ADDRESS
(1) 032746 001402          BEQ      5$          ;;BR IF NONE
(1) 032750 013716 001216          MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1) 032754          5$:
(1) 032754 022737 032140 000042  CMP      #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
(1) 032762 001001          BNE      6$          ;;BRANCH IF NO
(1) 032764 000000          HALT          ;;YES
(1) 032766          6$:
(1) 032766 000002          RTI          ;;RETURN
```

3121 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```
(1)
(2)
(1)
(1)
(1)
(1)
(1)
*****
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
```

```
(1) 032770          $ERRTYP:
(1) 032770 104401 001225          TYPE      ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 032774 010046          MOV      RO,-(SP)          ;;SAVE RO
(1) 032776 005000          CLR      RO              ;;PICKUP THE ITEM INDEX
(1) 033000 153700 001114          BISB     @#$ITEMB,RO
(1) 033004 001004          BNE      1$              ;;IF ITEM NUMBER IS ZERO, JUST
(1)
(2) 033006 013746 001116          MOV      $ERRPC,-(SP)    ;;TYPE THE PC OF THE ERROR
(2)
(2) 033012 104402          TYPOC          ;;SAVE $ERRPC FOR TYPEOUT
(1) 033014 000445          BR      10$            ;;ERROR ADDRESS
(1) 033016 005300          1$: DEC      RO          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 033020 006300          ASL      RO          ;;GET OUT
(1) 033022 006300          ASL      RO          ;;ADJUST THE INDEX SO THAT IT WILL
(1) 033024 006300          ASL      RO          ;;WORK FOR THE ERROR TABLE
(1) 033026 062700 001354          ADD      #$ERRTB,RO    ;;FORM TABLE POINTER
(1) 033032 012037 033042          MOV      (RO)+,2$      ;;PICKUP "ERROR MESSAGE" POINTER
(1) 033036 001404          BEQ      3$          ;;SKIP TYPEOUT IF NO POINTER
(1) 033040 104401          TYPE          ;;TYPE THE "ERROR MESSAGE"
(1) 033042 000000          2$: .WORD     0          ;;"ERROR MESSAGE" POINTER GOES HERE
(1) 033044 104401 001225          TYPE      ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 033050 012037 033060          3$: MOV      (RO)+,4$      ;;PICKUP "DATA HEADER" POINTER
(1) 033054 001404          BEQ      5$          ;;SKIP TYPEOUT IF 0
(1) 033056 104401          TYPE          ;;TYPE THE "DATA HEADER"
(1) 033060 000000          4$: .WORD     0          ;;"DATA HEADER" POINTER GOES HERE
(1) 033062 104401 001225          TYPE      ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
(1) 033066 010146          5$: MOV      R1,-(SP)      ;;SAVE R1
(1) 033070 012001          MOV      (RO)+,R1      ;;PICKUP "DATA TABLE" POINTER
(1) 033072 001415          BEQ      9$          ;;BR IF NO DATA TO BE TYPED
(1) 033074 012000          MOV      (RO)+,RO      ;;PICKUP "DATA FORMAT" POINTER
(1) 033076 105720          6$: TSTB     (RO)+        ;;"OCTAL" OR "DECIMAL"
(1) 033100 001003          BNE      7$          ;;BR IF DECIMAL
(2) 033102 013146          MOV      @ (R1)+,-(SP)  ;;SAVE @ (R1)+ FOR TYPEOUT
(2) 033104 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 033106 000402          BR      8$
(1) 033110          7$:
```







```

(1) 033674 104401 034507          TYPE      ,SMNEW      ;;PROMPT FOR NEW SWR
(1) 033700 005046          19$: CLR      -(SP)      ;;CLEAR COUNTER
(1) 033702 005046          CLR      -(SP)      ;;THE NEW SWR
(1) 033704 105777 145234      7$: TSTB     @STKS      ;;CHAR THERE?
(1) 033710 100375          BPL      7$         ;;IF NOT TRY AGAIN
(1)
(1) 033712 117746 145230          MOVB     @STKB, -(SP)  ;;PICK UP CHAR
(1) 033716 042716 177600          BIC     #'^C177,(SP)  ;;MAKE IT 7-BIT ASCII
(1)
(1)
(1) 033722 021627 000025          9$: CMP     (SP),#25   ;;IS IT A CONTROL-U?
(1) 033726 001005          BNE     10$         ;;BRANCH IF NOT
(1) 033730 104401 034464          TYPE     ,SCNTLU     ;;YES, ECHO CONTROL-U (^U)
(1) 033734 062706 000006      20$: ADD     #6,SP     ;;IGNORE PREVIOUS INPUT
(1) 033740 000757          BR      19$         ;;LET'S TRY IT AGAIN
(1)
(1)
(1) 033742 021627 000015          10$: CMP     (SP),#15   ;;IS IT A <CR>?
(1) 033746 001022          BNE     16$         ;;BRANCH IF NO
(1) 033750 005766 000004          TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
(1) 033754 001403          BEQ     11$         ;;BRANCH IF YES
(1) 033756 016677 000002 145154      MOV     2(SP),@SWR    ;;SAVE NEW SWR
(1) 033764 062706 000006      11$: ADD     #6,SP     ;;CLEAR UP STACK
(1) 033770 104401 001225          14$: TYPE     ,SCRLF   ;;ECHO <CR> AND <LF>
(1) 033774 123727 001135 000001      CMPB    $INTAG,#1    ;;RE-ENABLE TTY KBD INTERRUPTS?
(1) 034002 001003          BNE     15$         ;;BRANCH IF NOT
(1) 034004 012777 000100 145132      MOV     #100,@STKS   ;;RE-ENABLE TTY KBD INTERRUPTS
(1) 034012 000002          RTI     ;;RETURN
(1) 034014 004737 034732          16$: JSR     PC,$TYPEC  ;;ECHO CHAR
(1) 034020 021627 000060          CMP     (SP),#60    ;;CHAR < 0?
(1) 034024 002420          BLT     18$         ;;BRANCH IF YES
(1) 034026 021627 000067          CMP     (SP),#67    ;;CHAR > 7?
(1) 034032 003015          BGT     18$         ;;BRANCH IF YES
(1) 034034 042726 000060          BIC     #60,(SP)+   ;;STRIP-OFF ASCII
(1) 034040 005766 000002          TST     2(SP)       ;;IS THIS THE FIRST CHAR
(1) 034044 001403          BEQ     17$         ;;BRANCH IF YES
(1) 034046 006316          ASL     (SP)        ;;NO, SHIFT PRESENT
(1) 034050 006316          ASL     (SP)        ;;CHAR OVER TO MAKE
(1) 034052 006316          ASL     (SP)        ;;ROOM FOR NEW ONE.
(1) 034054 005266 000002          17$: INC     2(SP)     ;;KEEP COUNT OF CHAR
(1) 034060 056616 177776          BIS     -2(SP),(SP)  ;;SET IN NEW CHAR
(1) 034064 000707          BR      7$         ;;GET THE NEXT ONE
(1) 034066 104401 001224          18$: TYPE     ,SQUES   ;;TYPE ?<CR><LF>
(1) 034072 000720          BR      20$         ;;SIMULATE CONTROL-U
(1)
(1) .DSABL  LSB

```

```

(2) .....
(1) *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) *CALL:
(1) *      RDCHR      ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1) *      RETURN HERE ;;CHARACTER IS ON THE STACK
(1) *
(1) *
(1) *

```





```

(1) 034342 000726          BR      1$          ;;GO START OVER
(1) 034344 122713 000022 8$:  CMPB   #22,(R3)  ;;IS CHARACTER A '^R'?
(1) 034350 001011          BNE    3$          ;;BRANCH IF NO
(1) 034352 105013          CLRB   (R3)        ;;CLEAR THE CHARACTER
(1) 034354 104401 001225          TYPE  ,SCLRF      ;;TYPE A 'CR' & 'LF'
(1) 034360 104401 034450          TYPE  ,STYIN      ;;TYPE THE INPUT STRING
(1) 034364 000717          BR      2$          ;;GO PICKUP ANOTHER CHACTER
(1) 034366 104401 001224 4$:  TYPE  ,SQUES      ;;TYPE A '?'
(1) 034372 000712          BR      1$          ;;CLEAR THE BUFFER AND LOOP
(1) 034374 111337 034446 3$:  MOVB   (R3),9$     ;;ECHO THE CHARACTER
(1) 034400 104401 034446          TYPE  ,9$
(1) 034404 122723 000015          CMPB   #15,(R3)+  ;;CHECK FOR RETURN
(1) 034410 001305          BNE    2$          ;;LOOP IF NOT RETURN
(1) 034412 105063 177777          CLRB   -1(R3)     ;;CLEAR RETURN (THE 15)
(1) 034416 104401 001226          TYPE  ,SLF        ;;TYPE A LINE FEED
(1) 034422 005726          TST   (SP)+       ;;CLEAN RUBOUT KEY FROM THE STACK
(1) 034424 012603          MOV    (SP)+,R3   ;;RESTORE R3
(1) 034426 011646          MOV    (SP),-(SP) ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 034430 016666 000004 000002 MOV    4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
(1) 034436 012766 034450 000004 MOV    #STYIN,4(SP)
(1) 034444 000002          RTI
(1) 034446 000          9$:  .BYTE  0          ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 034447 000          .BYTE  0          ;;TERMINATOR
(1) 034450 000014          $TYIN: .BLKB 12.    ;;RESERVE 12. BYTES FOR TTY INPUT
(1) 034464 052536 005015 000  $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
(1) 034471 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
(1) 034476 005015 053523 020122 $MSWR:  .ASCIZ <15><12>/SWR - /
(1) 034504 020075 000          $MNEW:  .ASCIZ / NEW - /
(1) 034507 040 047040 053505
(1) 034514 036440 000040
    
```

```

3125 .SBTTL TYPE ROUTINE
(1)
(2)
(1) *****
(1) *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1) *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1) *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1) *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1) *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) *
(1) *CALL:
(1) *1) USING A TRAP INSTRUCTION
(1) * TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) *OR
(1) * TYPE
(1) * MESADR
(1) *
(1) $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(1) 034520 105737 001157 BPL 1$ ;;BR IF YES
(1) 034524 100002 HALT ;;HALT HERE IF NO TERMINAL
(1) 034526 000000 BR 3$ ;;LEAVE
(1) 034530 000430 1$: MOV R0, -(SP) ;;SAVE R0
(1) 034532 010046 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
(1) 034534 017600 000002 001250 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 034540 122737 000001 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
(1) 034546 001011 BITB #APTSPool,$ENVM ;;SPOOL MESSAGE TO APT
(1) 034550 132737 000100 001251
    
```

```

(1) 034556 001405          BEQ      62$          ;;NO,GO CHECK FOR CONSOLE
(1) 034560 010037 034570   MOV      RO,61$       ;;SETUP MESSAGE ADDRESS FOR APT
(1) 034564 004737 035722   JSR      PC,$ATY3     ;;SPOOL MESSAGE TO APT
(1) 034570 000000          .WORD    0            ;;MESSAGE ADDRESS
(1) 034572 132737 000040 001251 62$:   BITB    #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 034600 001003          BNE      60$          ;;YES,SKIP TYPE OUT
(1) 034602 112046          MOVB    (RO)+,-(SP)   ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 034604 001005          BNE      4$           ;;BR IF IT ISN'T THE TERMINATOR
(1) 034606 005726          TST     (SP)+         ;;IF TERMINATOR POP IT OFF THE STACK
(1) 034610 012600          MOV     (SP)+,RO     ;;RESTORE RO
(1) 034612 062716 000002   3$:   ADD     #2,(SP)    ;;ADJUST RETURN PC
(1) 034616 000002          RTI                    ;;RETURN
(1) 034620 122716 000011   4$:   CMPB    #HT,(SP)    ;;BRANCH IF <HT>
(1) 034624 001430          BEQ     8$           ;;BRANCH IF NOT <CRLF>
(1) 034626 122716 000200   CMPB    #CRLF,(SP)
(1) 034632 001006          BNE     5$           ;;POP <CR><LF> EQUIV
(1) 034634 005726          TST     (SP)+         ;;TYPE A CR AND LF
(1) 034636 104401          TYPE
(1) 034640 001225          $CRLF
(1) 034642 105037 034776   CLRB    $CHARCNT     ;;CLEAR CHARACTER COUNT
(1) 034646 000755          BR      2$           ;;GET NEXT CHARACTER
(1) 034650 004737 034732   5$:   JSR     PC,$TYPEC    ;;GO TYPE THIS CHARACTER
(1) 034654 123726 001156   6$:   CMPB    $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 034660 001350          BNE     2$           ;;IF NO GO GET NEXT CHAR.
(1) 034662 013746 001154   MOV     $NULL,-(SP)  ;;GET # OF FILLER CHARS. NEEDED
(1)                                ;;AND THE NULL CHAR.
(1) 034666 105366 000001   7$:   DECB    1(SP)       ;;DOES A NULL NEED TO BE TYPED?
(1) 034672 002770          BLT     6$           ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 034674 004737 034732   JSR     PC,$TYPEC    ;;GO TYPE A NULL
(1) 034700 105337 034776   DECB    $CHARCNT     ;;DO NOT COUNT AS A COUNT
(1) 034704 000770          BR      7$           ;;LOOP
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1) 034706 112716 000040   8$:   MOVB    #' ,(SP)    ;;REPLACE TAB WITH SPACE
(1) 034712 004737 034732   9$:   JSR     PC,$TYPEC    ;;TYPE A SPACE
(1) 034716 132737 000007 034776   BITB    #7,$CHARCNT  ;;BRANCH IF NOT AT
(1) 034724 001372          BNE     9$           ;;TAB STOP
(1) 034726 005726          TST     (SP)+         ;;POP SPACE OFF STACK
(1) 034730 000724          BR      2$           ;;GET NEXT CHARACTER
(1) 034732 105777 144212   $TYPEC: TSTB    @$TIPS  ;;WAIT UNTIL PRINTER IS READY
(1) 034736 100375          BPL     $TYPEC
(1) 034740 116677 000002 144204   MOVB    2(SP),@$TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 034746 122766 000015 000002   CMPB    #CR,2(SP)   ;;IS CHARACTER A CARRIAGE RETURN?
(1) 034754 001003          BNE     1$           ;;BRANCH IF NO
(1) 034756 105037 034776   CLRB    $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
(1) 034762 000406          BR      $TYPEX       ;;EXIT
(1) 034764 122766 000012 000002 1$:   CMPB    #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
(1) 034772 001402          BEQ     $TYPEX       ;;BRANCH IF YES
(1) 034774 105227          INCB    (PC)+        ;;COUNT THE CHARACTER
(1) 034776 000000          $CHARCNT: .WORD    0 ;;CHARACTER COUNT STORAGE
(1) 035000 000207          $TYPEX: RTS      PC

```

3126  
 (1)  
 (2)

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE  
 ;:.....

```

(1) ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1) ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1) ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1) ;*CALL:
(1) ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1) ;*   TYPOS   ;;CALL FOR TYPEOUT
(1) ;*   .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1) ;*   .BYTE  M              ;;M=1 OR 0
(1) ;*                                     ;;1=TYPE LEADING ZEROS
(1) ;*                                     ;;0=SUPPRESS LEADING ZEROS
(1) ;*
(1) ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1) ;*$TYPOS OR $TYPOC
(1) ;*CALL:
(1) ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1) ;*   TYPON   ;;CALL FOR TYPEOUT
(1) ;*
(1) ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1) ;*CALL:
(1) ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
(1) ;*   TYPOC   ;;CALL FOR TYPEOUT
(1) ;*
(1) 035002 017646 000000 $TYPOS: MOV @ (SP),-(SP) ;;PICKUP THE MODE
(1) 035006 116637 000001 035225 MOV 1(SP), $OFILL ;;LOAD ZERO FILL SWITCH
(1) 035014 112637 035227 MOV  (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
(1) 035020 062716 000002 ADD #2, (SP) ;;ADJUST RETURN ADDRESS
(1) 035024 000406 BR $TYPON
(1) 035026 112737 000001 035225 $TYPOC: MOV #1, $OFILL ;;SET THE ZERO FILL SWITCH
(1) 035034 112737 000006 035227 MOV #6, $OMODE+1 ;;SET FOR SIX(6) DIGITS
(1) 035042 112737 000005 035224 $TYPON: MOV #5, $OCNT ;;SET THE ITERATION COUNT
(1) 035050 010346 MOV R3, -(SP) ;;SAVE R3
(1) 035052 010446 MOV R4, -(SP) ;;SAVE R4
(1) 035054 010546 MOV R5, -(SP) ;;SAVE R5
(1) 035056 113704 035227 MOV $OMODE+1, R4 ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 035062 005404 NEG R4
(1) 035064 062704 000006 ADD #6, R4 ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 035070 110437 035226 MOV R4, $OMODE ;;SAVE IT FOR USE
(1) 035074 113704 035225 MOV $OFILL, R4 ;;GET THE ZERO FILL SWITCH
(1) 035100 016605 000012 MOV 12(SP), R5 ;;PICKUP THE INPUT NUMBER
(1) 035104 005003 CLR R3 ;;CLEAR THE OUTPUT WORD
(1) 035106 006105 1$: ROL R5 ;;ROTATE MSB INTO 'C'
(1) 035110 000404 BR 3$ ;;GO DO MSB
(1) 035112 006105 2$: ROL R5 ;;FORM THIS DIGIT
(1) 035114 006105 ROL R5
(1) 035116 006105 ROL R5
(1) 035120 010503 MOV R5, R3
(1) 035122 006103 3$: ROL R3 ;;GET LSB OF THIS DIGIT
(1) 035124 105337 035226 DECB $OMODE ;;TYPE THIS DIGIT?
(1) 035130 100016 BPL 7$ ;;BR IF NO
(1) 035132 042703 177770 BIC #177770, R3 ;;GET RID OF JUNK
(1) 035136 001002 BNE 4$ ;;TEST FOR 0
(1) 035140 005704 TST R4 ;;SUPPRESS THIS 0?
(1) 035142 001403 BEQ 5$ ;;BR IF YES
(1) 035144 005204 4$: INC R4 ;;DON'T SUPPRESS ANYMORE 0'S
(1) 035146 052703 000060 BIS #'0, R3 ;;MAKE THIS DIGIT ASCII
(1) 035152 052703 000040 5$: BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
    
```

```
(1) 035156 110337 035222      MOVB  R3,8$           ;;SAVE FOR TYPING
(1) 035162 104401 035222      TYPE  ,8$           ;;GO TYPE THIS DIGIT
(1) 035166 105337 035224      7$:  DECIB $OCNT     ;;COUNT BY 1
(1) 035172 003347           BGT   2$           ;;BR IF MORE TO DO
(1) 035174 002402           BLT   6$           ;;BR IF DONE
(1) 035176 005204           INC   R4           ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 035200 000744           BR    2$           ;;GO DO THE LAST DIGIT
(1) 035202 012605      6$:  MOV   (SP)+,R5     ;;RESTORE R5
(1) 035204 012604      MOV   (SP)+,R4     ;;RESTORE R4
(1) 035206 012603      MOV   (SP)+,R3     ;;RESTORE R3
(1) 035210 016666 000002 000004 MOV   2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
(1) 035216 012616      MOV   (SP)+,(SP)
(1) 035220 000002      RTI                   ;;RETURN
(1) 035222 000          8$:  .BYTE  0           ;;STORAGE FOR ASCII DIGIT
(1) 035223 000          .BYTE  0           ;;TERMINATOR FOR TYPE ROUTINE
(1) 035224 000      $OCNT: .BYTE  0       ;;OCTAL DIGIT COUNTER
(1) 035225 000      $OFILL: .BYTE  0       ;;ZERO FILL SWITCH
(1) 035226 000000      $OMODE: .WORD  0     ;;NUMBER OF DIGITS TO TYPE
3127 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
```

```
(1) *****
(2) *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1) *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1) *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1) *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1) *REPLACED WITH SPACES.
```

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

```
(1) 035230      $TYPDS:
(3) 035230 010046      MOV   R0,-(SP)        ;;PUSH R0 ON STACK
(3) 035232 010146      MOV   R1,-(SP)        ;;PUSH R1 ON STACK
(3) 035234 010246      MOV   R2,-(SP)        ;;PUSH R2 ON STACK
(3) 035236 010346      MOV   R3,-(SP)        ;;PUSH R3 ON STACK
(3) 035240 010546      MOV   R5,-(SP)        ;;PUSH R5 ON STACK
(1) 035242 012746 020200 MOV   #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1) 035246 016605 000020 MOV   20(SP),R5       ;;GET THE INPUT NUMBER
(1) 035252 100004      BPL   1$           ;;BR IF INPUT IS POS.
(1) 035254 005405      NEG   R5           ;;MAKE THE BINARY NUMBER POS.
(1) 035256 112766 000055 000001 MOVB  #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1) 035264 005000      1$:  CLR   R0           ;;ZERO THE CONSTANTS INDEX
(1) 035266 012703 035444      MOV   #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
(1) 035272 112723 000040      MOVB  #'',(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1) 035276 005002      2$:  CLR   R2           ;;CLEAR THE BCD NUMBER
(1) 035300 016001 035434      MOV   $DTBL(R0),R1    ;;GET THE CONSTANT
(1) 035304 160105      3$:  SUB   R1,R5         ;;FORM THIS BCD DIGIT
(1) 035306 002402      BLT   4$           ;;BR IF DONE
(1) 035310 005202      INC   R2           ;;INCREASE THE BCD DIGIT BY 1
(1) 035312 000774      BR    3$
(1) 035314 060105      4$:  ADD   R1,R5         ;;ADD BACK THE CONSTANT
(1) 035316 005702      TST   R2           ;;CHECK IF BCD DIGIT=0
(1) 035320 001002      BNE   5$           ;;FALL THROUGH IF 0
(1) 035322 105716      TSTB  (SP)          ;;STILL DOING LEADING 0'S?
(1) 035324 100407      BMI   7$           ;;BR IF YES
(1) 035326 106316      5$:  ASLB  (SP)          ;;MSD?
```



```

(3)
(3)
(3)
(3) 035510 035476
(3) 035512 034520
(3) 035514 035026
(3) 035516 035002
(3) 035520 035042
(3) 035522 035230
(1)
(3) 035524 033662
(1)
(3) 035526 033612
(3) 035530 034074
(3) 035532 034214
(3) 035534 033452
3129
(1)
(2)
(1)
(1) 035536 012737 035676 000024
(1) 035544 012737 000340 000026
(3) 035552 010046
(3) 035554 010146
(3) 035556 010246
(3) 035560 010346
(3) 035562 010446
(3) 035564 010546
(3) 035566 017746 143346
(1) 035572 010637 035702
(1) 035576 012737 035610 000024
(1) 035604 000000
(1) 035606 000776
(1)
(2)
(1)
(1) 035610 012737 035676 000024
(1) 035616 013706 035702
(1) 035622 005037 035702
(1) 035626 005237 035702
(1) 035632 001375
(3) 035634 012677 143300
(3) 035640 012605
(3) 035642 012604
(3) 035644 012603
(3) 035646 012602
(3) 035650 012601
(3) 035652 012600
(1) 035654 012737 035536 000024
(1) 035662 012737 000340 000026
(1) 035670 104401
(1) 035672 035704
(1) 035674 000002
(1) 035676 000000
(1) 035700 000776
(1) 035702 000000

: ROUTINE
:-----
$TRPAD: .WORD $TRAP2
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

$GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

$CKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
$RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
$RDOCT ;;CALL=RDOCT TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

.SBTTL POWER DOWN AND UP ROUTINES

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

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

```

( ) 035704 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'
( ) 035712 000122
(1)
313) .EVEN
(1) .SBTTL APT COMMUNICATIONS ROUTINE
(2)
(1) 035714 112737 000001 036160 $ATY1: MOV #1,$FFLG ;;TO REPORT FATAL ERROR
(1) 035722 112737 000001 036156 $ATY3: MOV #1,$MFLG ;;TO TYPE A MESSAGE
(1) 035730 000403 BR $ATYC
(1) 035732 112737 000001 036160 $ATY4: MOV #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
(1) 035740 $ATYC:
(3) 035740 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
(3) 035742 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
(1) 035744 105737 036156 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
(1) 035750 001450 BEQ 5$ ;;IF NOT: BR
(1) 035752 122737 000001 001250 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
(1) 035760 001031 BNE 3$ ;;IF NOT: BR
(1) 035762 132737 000100 001251 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1) 035770 001425 BEQ 3$ ;;IF NOT: BR
(1) 035772 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.
(1) 035776 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 036004 005737 001230 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
(1) 036010 001375 BNE 1$ ;;IF NOT: WAIT
(1) 036012 010037 001244 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
(1) 036016 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE
(1) 036020 001376 BNE 2$
(1) 036022 163700 001244 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
(1) 036026 006200 ASR R0 ;;GET MESSAGE LNTH IN WORDS
(1) 036030 010037 001246 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
(1) 036034 012737 000004 001230 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
(1) 036042 000413 BR 5$
(1) 036044 017637 000004 036070 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
(1) 036052 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
(3) 036060 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
(1) 036064 004737 034520 JSR PC,$TYPE ;;CALL TYPE MACRO
(1) 036070 000000 4$: .WORD 0
(1) 036072 5$:
(1) 036072 105737 036160 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
(1) 036076 001416 BEQ 12$ ;;IF NOT: BR
(1) 036100 005737 001250 TST $ENV ;;RUNNING UNDER APT?
(1) 036104 001413 BEQ 12$ ;;IF NOT: BR
(1) 036106 005737 001230 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
(1) 036112 001375 BNE 11$ ;;IF NOT: WAIT
(1) 036114 017637 000004 001232 MOV @4(SP),$FATAL ;;GET ERROR #
(1) 036122 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
(1) 036130 005237 001230 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
(1) 036134 105037 036160 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
(1) 036140 105037 036157 CLRB $LFLG ;;CLEAR LOG FLAG
(1) 036144 105037 036156 CLRB $MFLG ;;CLEAR MESSAGE FLAG
(3) 036150 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
(3) 036152 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
(1) 036154 000207 RTS PC ;;RETURN
(1) 036156 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
(1) 036157 000 $LFLG: .BYTE 0 ;;LOG FLAG
(1) 036160 000 $FFLG: .BYTE 0 ;;FATAL FLAG
(1) 036162 .EVEN

```

(1)	000200	APTSIZE=200
(1)	00000i	APTENV=001
(1)	000100	APTSPool=100
(1)	000040	APTCSUP=040



```

3132          .SBTTL  MESSAGES
3133
3134 036162 041501 020122 044502 EM1:  .ASCIZ      /ACR BIT3-0 LOAD-READ ERROR/
    036170 031524 030055 046040
    036176 040517 026504 042522
    036204 042101 042440 051122
    036212 051117      000
3135 036215      101 051103 041040 EM2:  .ASCIZ      /ACR BIT3-0 WERE NOT CLEARED BY RESET (ACR-CLR)/
    036222 052111 026463 020060
    036230 042527 042522 047040
    036236 052117 041440 042514
    036244 051101 042105 041040
    036252 020131 042522 042523
    036260 020124 040450 051103
    036266 041455 051114 000051
3136 036274 042511 024040 043520 EM7:  .ASCIZ      /IE (PGTE BIT2) CANNOT BE SET/
    036302 042524 041040 052111
    036310 024462 041440 047101
    036316 047516 020124 042502
    036324 051440 052105      000
3137 036331      111 020105 050050 EM10: .ASCIZ      /IE (PGTE BIT2) CANNOT BE WRITTEN TO 0/
    036336 052107 020105 044502
    036344 031124 020051 040503
    036352 047116 052117 041040
    036360 020105 051127 052111
    036366 042524 020116 047524
    036374 030040      000
3138 036377      111 020105 050050 EM11: .ASCIZ      /IE (PGTE BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)/
    036404 052107 020105 044502
    036412 031124 020051 040503
    036420 047116 052117 041040
    036426 020105 046103 040505
    036434 042522 020104 054502
    036442 051040 051505 052105
    036450 024040 041501 020122
    036456 046103 024522      000
3139 036463      120 050124 024040 EM12: .ASCIZ      /PTP (PGTE BIT1) CANNOT BE SET/
    036470 043520 042524 041040
    036476 052111 024461 041440
    036504 047101 047516 020124
    036512 042502 051440 052105
    036520      000
3140 036521      120 050124 024040 EM13: .ASCIZ      /PTP (PGTE BIT1) CANNOT BE WRITTEN TO 0/
    036526 043520 042524 041040
    036534 052111 024461 041440
    036542 047101 047516 020124
    036550 042502 053440 044522
    036556 052124 047105 052040
    036564 020117 000060
3141 036570 052120 020120 050050 EM14: .ASCIZ      /PTP (PGTE BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)/
    036576 052107 020105 044502
    036604 030524 020051 040503
    036612 047116 052117 041040
    036620 020105 046103 040505
    036626 042522 020104 054502
    036634 051040 051505 052105
  
```

3142	036642	024040	041501	020122			
	036650	046103	024522	000			
	036655	120	052107	020105	EM3:	.ASCIZ	/PGTE BIT11-8 LOAD-READ ERROR/
	036662	044502	030524	026461			
	036670	020070	047514	042101			
	036676	051055	040505	020104			
3143	036704	051105	047522	000122	EM4:	.ASCIZ	/PGTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	036712	043520	042524	041040			
	036720	052111	030461	034055			
	036726	053440	051105	020105			
	036734	047516	020124	046103			
	036742	040505	042522	020104			
	036750	054502	051040	051505			
	036756	052105	024040	041501			
	036764	020122	046103	024522			
	036772	000					
3144	036773	120	052107	020105	EM5:	.ASCIZ	/PGTE BIT3-0 LOAD-READ ERROR/
	037000	044502	031524	030055			
	037006	046040	040517	026504			
	037014	042522	042101	042440			
	037022	051122	051117	000			
3145	037027	120	052107	020105	EM6:	.ASCIZ	/PGTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	037034	044502	031524	030055			
	037042	053440	051105	020105			
	037050	047516	020124	046103			
	037056	040505	042522	020104			
	037064	054502	051040	051505			
	037072	052105	024040	041501			
	037100	020122	046103	024522			
	037106	000					
3146	037107	123	052124	020105	EM15:	.ASCIZ	/STTE BIT11-8 LOAD-READ ERROR/
	037114	044502	030524	026461			
	037122	020070	047514	042101			
	037130	051055	040505	020104			
	037136	051105	047522	000122			
3147	037144	052123	042524	041040	EM16:	.ASCIZ	/STTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	037152	052111	030461	034055			
	037160	053440	051105	020105			
	037166	047516	020124	046103			
	037174	040505	042522	020104			
	037202	054502	051040	051505			
	037210	052105	024040	041501			
	037216	020122	046103	024522			
	037224	000					
3148	037225	123	052124	020105	EM17:	.ASCIZ	/STTE BIT3-0 LOAD-READ ERROR/
	037232	044502	031524	030055			
	037240	046040	040517	026504			
	037246	042522	042101	042440			
	037254	051122	051117	000			
3149	037261	123	052124	020105	EM20:	.ASCIZ	/STTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	037266	044502	031524	030055			
	037274	053440	051105	020105			
	037302	047516	020124	046103			
	037310	040505	042522	020104			
	037316	054502	051040	051505			
	037324	052105	024040	041501			

	037332	020122	046103	024522			
	037340	000					
3150	037341	114	042513	024040	EM21:	.ASCIZ	/LKE (STCS BIT2) CANNOT BE SET/
	037346	052123	051503	041040			
	037354	052111	024462	041440			
	037362	047101	047516	020124			
	037370	042502	051440	052105			
	037376	000					
3151	037377	114	042513	024040	EM22:	.ASCIZ	/LKE (STCS BIT2) CANNOT BE WRITTEN TO 0/
	037404	052123	051503	041040			
	037412	052111	024462	041440			
	037420	047101	047516	020124			
	037426	042502	053440	044522			
	037434	052124	047105	052040			
	037442	020117	000060				
3152	037446	045514	020105	051450	EM23:	.ASCIZ	/LKE (STCS BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)/
	037454	041524	020123	044502			
	037462	031124	020051	040503			
	037470	047116	052117	041040			
	037476	020105	046103	040505			
	037504	042522	020104	054502			
	037512	051040	051505	052105			
	037520	024040	041501	020122			
	037526	046103	024522	000			
3153	037533	123	050124	024040	EM24:	.ASCIZ	/STP (STCS BIT1) CANNOT BE SET/
	037540	052123	051503	041040			
	037546	052111	024461	041440			
	037554	047101	047516	020124			
	037562	042502	051440	052105			
	037570	000					
3154	037571	123	050124	024040	EM25:	.ASCIZ	/STP (STCS BIT1) CANNOT BE WRITTEN TO 0/
	037576	052123	051503	041040			
	037604	052111	024461	041440			
	037612	047101	047516	020124			
	037620	042502	053440	044522			
	037626	052124	047105	052040			
	037634	020117	000060				
3155	037640	052123	020120	051450	EM26:	.ASCIZ	/STP (STCS BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)/
	037646	041524	020123	044502			
	037654	030524	020051	040503			
	037662	047116	052117	041040			
	037670	020105	046103	040505			
	037676	042522	020104	054502			
	037704	051040	051505	052105			
	037712	024040	041501	020122			
	037720	046103	024522	000			
3156	037725	105	041116	024040	EM27:	.ASCIZ	/ENB (STCS BIT0) CANNOT BE SET/
	037732	052123	051503	041040			
	037740	052111	024460	041440			
	037746	047101	047516	020124			
	037754	042502	051440	052105			
	037762	000					
3157	037763	105	041116	024040	EM30:	.ASCIZ	/ENB (STCS BIT0) CANNOT BE WRITTEN TO 0/
	037770	052123	051503	041040			
	037776	052111	024460	041440			
	040004	047101	047516	020124			

	040012	042502	053440	044522			
	040020	052124	047105	052040			
	040026	020117	000060				
3158	040032	047105	020102	051450	EM31:	.ASCIZ	/ENB (STCS BIT0) CANNOT BE CLEARED BY RESET (ACR CLR)/
	040040	041524	020123	044502			
	040046	030124	020051	040503			
	040054	047116	052117	041040			
	040062	020105	046103	040505			
	040070	042522	020104	054502			
	040076	051040	051505	052105			
	040104	024040	041501	020122			
	040112	046103	024522	000			
3159	040117	123	041524	020123	EM32:	.ASCIZ	/STCS BIT15-8 LOAD-READ ERROR/
	040124	044502	030524	026465			
	040132	020070	047514	042101			
	040140	051055	040505	020104			
3160	040146	051105	047522	000122	EM33:	.ASCIZ	/STCS BIT15-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	040154	052123	051503	041040			
	040162	052111	032461	034055			
	040170	053440	051105	020105			
	040176	047516	020124	046103			
	040204	040505	042522	020104			
	040212	054502	051040	051505			
	040220	052105	024040	041501			
	040226	020122	046103	024522			
	040234	000					
3161	040235	111	051515	020113	EM34:	.ASCIZ	/IMSK BIT11-8 LOAD-READ ERROR/
	040242	044502	030524	026461			
	040250	020070	047514	042101			
	040256	051055	040505	020104			
	040264	051105	047522	000122			
3162	040272	046511	045523	041040	EM35:	.ASCIZ	/IMSK BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	040300	052111	030461	034055			
	040306	053440	051105	020105			
	040314	047516	020124	046103			
	040322	040505	042522	020104			
	040330	054502	051040	051505			
	040336	052105	024040	041501			
	040344	020122	046103	024522			
	040352	000					
3163	040353	111	051515	020113	EM36:	.ASCIZ	/IMSK BIT3-0 LOAD-READ ERROR/
	040360	044502	031524	030055			
	040366	046040	040517	026504			
	040374	042522	042101	042440			
	040402	051122	051117	000			
3164	040407	111	051515	020113	EM37:	.ASCIZ	/IMSK BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	040414	044502	031524	030055			
	040422	053440	051105	020105			
	040430	047516	020124	046103			
	040436	040505	042522	020104			
	040444	054502	051040	051505			
	040452	052105	024040	041501			
	040460	020122	046103	024522			
	040466	000					
3165	040467	115	041524	041040	EM40:	.ASCIZ	/MTC BIT11-8 LOAD-READ FRROR/
	040474	052111	030461	034055			

	040502	046040	040517	026504		
	040510	042522	042101	042440		
	040516	051122	051117	000		
3166	040523	115	041524	041040	EM41: .ASCIZ	/MTC BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	040530	052111	030461	034055		
	040536	053440	051105	020105		
	040544	047516	020124	046103		
	040552	040505	042522	020104		
	040560	054502	051040	051505		
	040566	052105	024040	041501		
	040574	020122	046103	024522		
	040602	000				
3167	040603	115	041524	041040	EM42: .ASCIZ	/MTC BIT3-0 LOAD-READ ERROR/
	040610	052111	026463	020060		
	040616	047514	042101	051055		
	040624	040505	020104	051105		
	040632	047522	000122			
3168	040636	052115	020103	044502	EM43: .ASCIZ	/MTC BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	040644	031524	030055	053440		
	040652	051105	020105	047516		
	040660	020124	046103	040505		
	040666	042522	020104	054502		
	040674	051040	051505	052105		
	040702	024040	041501	020122		
	040710	046103	024522	000		
3169	040715	111	051511	020124	EM44: .ASCII	/IIST ERROR - REFER TO THE LISTING AT "PC"/
	040722	051105	047522	020122		
	040730	020055	042522	042506		
	040736	020122	047524	052040		
	040744	042510	046040	051511		
	040752	044524	043516	040440		
	040760	020124	050042	021103		
3170	040766	052200	020117	042504	.ASCIZ	<CRLF>/TO DETERMINE THE ERROR./
	040774	042524	046522	047111		
	041002	020105	044124	020105		
	041010	051105	047522	027122		
	041016	000				
3171	041017	101	051103	024040	EM45: .ASCIZ	/ACR (3:0) AUTO-INCREMENT ERROR/
3172	041024	035063	024460	040440		
	041032	052125	026517	047111		
	041040	051103	046505	047105		
	041046	020124	051105	047522		
	041054	000122				

Line	Code	Code	Code	Code	Label	Code	Code	Code	Code	Code	Code
3174	041056	040520	051523	020040	DH1:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	ACR/
	041064	020040	041520	020040							
	041072	020040	020040	054105							
	041100	041520	042124	020040							
	041106	041501	052524	046101							
	041114	020040	041501	000122							
3175	041122	040520	051523	020040	DH2:	.ASCIZ	/PASS	PC	ACR/		
	041130	020040	041520	020040							
	041136	020040	020040	041501							
	041144	000122									
3176	041146	040520	051523	020040	DH3:	.ASCIZ	/PASS	PC	PGTE/		
	041154	020040	041520	020040							
	041162	020040	020040	043520							
	041170	042524	000								
3177	041173	120	051501	020123	DH4:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	PGTE/
	041200	020040	050040	020103							
	041206	020040	020040	042440							
	041214	050130	052103	020104							
	041222	040440	052103	040525							
	041230	020114	050040	052107							
	041236	000105									
3178	041240	040520	051523	020040	DH5:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	STTE/
	041246	020040	041520	020040							
	041254	020040	020040	054105							
	041262	041520	042124	020040							
	041270	041501	052524	046101							
	041276	020040	052123	042524							
	041304	000									
3179	041305	120	051501	020123	DH6:	.ASCIZ	/PASS	PC	STTE/		
	041312	020040	050040	020103							
	041320	020040	020040	051440							
	041326	052124	000105								
3180	041332	040520	051523	020040	DH7:	.ASCIZ	/PASS	PC	STCS/		
	041340	020040	041520	020040							
	041346	020040	020040	052123							
	041354	051503	000								
3181	041357	120	051501	020123	DH8:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	STCS/
	041364	020040	050040	020103							
	041372	020040	020040	042440							
	041400	050130	052103	020104							
	041406	040440	052103	040525							
	041414	020114	051440	041524							
	041422	000123									
3182	041424	040520	051523	020040	DH9:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	IMSK/
	041432	020040	041520	020040							
	041440	020040	020040	054105							
	041446	041520	042124	020040							
	041454	041501	052524	046101							
	041462	020040	046511	045523							
	041470	000									
3183	041471	120	051501	020123	DH10:	.ASCIZ	/PASS	PC	IMSK/		
	041476	020040	050040	020103							
	041504	020040	020040	044440							
	041512	051515	000113								
3184	041516	040520	051523	020040	DH11:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	MTCE/
	041524	020040	041520	020040							



```

3200 042160 001236 001116 024754 DT3: .WORD $PASS,$ERRPC,$LINMBR,$GDDAT,$BDDAT,$DISPLY,$TMPO,$TMP1,0
      042166 001124 001126 024750
      042174 001176 001200 000000
3201 042202 001 000 000 DAF1: .BYTE 1,0,0,0,0,0,0,0
      042205 000 000 000
      042210 000 000
3202 .EVEN
3203
3204 000001 .END ;THAT'S ALL FOLKS!

```







BM2 = 002000	543#																		
BM3 = 004000	542#																		
BPTVEC = 000014	241#																		
BRKTAB 022002	2038	2052	2063	2074	2267#														
BRK0 = 000400	578#																		
BRK1 = 001000	577#																		
BRK2 = 002000	576#																		
BRK3 = 004000	575#																		
CARET 042131	2510	2610	2611	3195#															
CKSWR = 104407	2515	3119	3120	3128#															
CLEAR 023454	775	854	856	858	860	862	864	866	868	870	872	874	876						
	878	880	2471#																
CLEARX 023450	899	929	1935	1940	2031	2114	2144	2467#	2756	2923	2950	3021	3050						
	3079																		
CLR = 100000	483#	792	928	1931	2143	2152	2431	2444	2602										
CNFMAX 003004	748#	2581	2588																
CONFIG 025020	733	739	893	1932	1944	2034	2046	2049	2145	2160	2587*	2588	2654#						
CR = 000015	241#	3125																	
CRLF = 000200	241#	613#	704	716	732	2676	2677	2678	2679	2680	2683	2689	2765						
	2789	2816	2908	2925	2926	2952	2963	2986	2988	2990	2991	2992	2993						
	2994	2995	2996	3000	3125	3170	3187	3190	3191	3194	3195								
DAF1 042202	261	267	273	279	285	291	297	303	309	315	321	327	333						
	339	345	351	357	363	369	375	381	387	393	399	405	411						
	417	423	429	435	441	447	453	459	465	471	477	3201#							
DCF 024736	1040	1145	1213	1305	1631	1694	1725	1778	1938	2017	2044	2061	2072						
	2082*	2106*	2108	2127	2150	2156*	2234	2345	2462*	2629#	2834	2842*	2954						
	2959	2967*																	
DCFE = 000007	491#	1039	1144	1212	1304	1630	1724	1937	2043	2060	2071	2081	2105						
	2126	2149	2233	2344	2833	2841	2953	2958	2966										
DCFH 024740	2637#																		
DCFTAB 021772	2039	2064	2106	2256#															
DCFO = 000001	582#																		
DCF1 = 000002	581#																		
DCF2 = 000004	580#																		
DCF3 = 000010	579#																		
DDISP = 177570	241#	254	703																
DH1 041056	259	475	3174#																
DH10 041471	427	439	3183#																
DH11 041516	445	457	3184#																
DH12 041563	451	463	3185#																
DH13 041610	469	3186#																	
DH2 041122	265	3175#																	
DH3 041146	277	289	295	301	307	313	319	325	3176#										
DH4 041173	271	283	3177#																
DH5 041240	331	343	3178#																
DH6 041305	337	349	3179#																
DH7 041332	355	361	367	373	379	385	391	397	403	415	3180#								
DH8 041357	409	3181#																	
DH9 041424	421	433	3182#																
DISPLA 001142	254#	703*	3119*	3120*															
DISPLY 024750	779*	780	793*	794	803*	806	816	823	830	837	844*	846	851						
	854*	856*	858*	860*	862*	864*	866*	868*	870*	872*	874*	876*	878*						
	880*	1040*	1045*	1050*	1062*	1063	1145*	1150*	1155*	1213*	1218*	1223*	1235*						
	1236	1305*	1310*	1315*	1368*	1369	1453*	1454	1512*	1513	1526*	1528	155						
	1541*	1542	1555*	1556	1563*	1565	1572	1578*	1579	2417*	2453*	2454	2647#						
	2697*	2709	2786*	2794	2811	2817	2886*	2889	2890	2954*	2955	2960	2968*						









SBF2	=	000004	565#																
SBF3	=	000010	564#																
SBO	=	000400	527#																
SB1	=	001000	526#																
SB2	=	002000	525#																
SB3	=	004000	524#																
SCBOOT		031420	3012	3021#															
SCPBAL		032040	3016	3114#															
SCPIAL		032026	3015	3110#															
SCPISF		032000	3013	3102#															
SCPXMT		031664	3079#	3098	3107	3111	3115												
SCSTSF		031554	3014	3050#	3067														
SELPAR		002654	246	727#															
SETMNT		023306	1010	1122	1339	1385	1628	1722	1799	1848	2036	2283	2443#	2693					
SETMNX		023414	2459#	2467															
SETMNI		023312	2444#	2482															
SIDNUM		024730	990	993	2606*	2614#	2707	2873	3033										
SIDTAB		021762	1018	1028	1133	1191	1201	1293	1645	1741	1807	1858	1890	1970	2079				
			2246#	2445	2875														
SIDO	=	000000	513#	2246															
SID1	=	000400	514#	2247															
SID2	=	001000	515#	2248															
SID3	=	001400	516#	2249															
SIFTAB		021772	1264	1509	1591	2255#													
SIFO	=	000001	571#																
SIF1	=	000002	570#																
SIF2	=	000004	569#																
SIF3	=	000010	568#																
SIO	=	000001	531#																
S12	=	000002	529#	530#															
S13	=	000010	528#																
SLDISP		031406	3009	3012#															
SLMSG		030214	2983	2986#															
SLOOPS		030160	766	2976#															
SLOOP1		031232	2980	2984	2999#	3006	3046	3061	3092										
SNAPO		024756	898	912	930	957	1631*	1693	1725*	1777	1938*	2016	2044*	2045	2053				
			2150*	2232	2650#														
SPACE		042110	2534	2551	2564	3192#													
SRO	=	177572	244#	3122*															
SR1	=	177574	244#																
SR2	=	177576	244#																
SR3	=	172516	244#	3122*															
SSTMNT		023476	1183	1282	1424	1470	1506	1549	1589	1884	2481#								
STACK	=	001100	241#	703	2762	2824	2914	2928											
START		002024	703#	724	728														
STBTAB		022002	2217	2219	2263#	2729													
STCCNT		025022	718*	2363	2373*	2388*	2402*	2417	2656#										
STCS		024736	866*	868*	870*	872*	1510*	1512	1526	1552*	1555	1563	1592*	2189*	2194				
			2202	2625#	2730*	2867	2930*												
STCSE	=	000003	487#	866	868	870	872	945	1511	1525	1554	1562	1892	2188	2201				
			2301	2317	2866	2929	3062												
STCSH		024740	1594*	1600	1609	2634#													
STF		024736	1050	1155	1206	1231*	1233	1265*	1268	1298	1321*	1325	1374	1412	1435				
			1481	1689	1773	2012	2155*	2221	2356	2461*	2628#	2741	2776	2781*	2846				
			2851*																
STFE	=	000006	490#	1049	1154	1205	1230	1232	1263	1267	1297	1319	1324	1373	1411				





TST20	006202	866	868#											
TST21	006256	868#												
TST22	006342	868#												
TST23	006416	868	870#											
TST24	006472	870#												
TST25	006556	870#												
TST26	006632	870	872#											
TST27	007042	874#												
TST3	004276	854#												
TST30	007252	876#												
TST31	007462	878#												
TST32	007672	880#												
TST33	010106	892#												
TST34	010634	1008#												
TST35	011524	1121#												
TST36	012052	1182#												
TST37	012646	1281#												
TST4	004506	856#												
TST40	013174	1338#												
TST41	013470	1384#												
TST42	013716	1423#												
TST43	014212	1469#												
TST44	014440	1505#												
TST45	014710	1548#												
TST46	015136	1585#												
TST47	015334	1627#												
TST5	004716	858#												
TST50	016062	1721#												
TST51	016530	1798#												
TST52	017002	1847#												
TST53	017174	1883#												
TST54	017400	1930#												
TST55	020162	2030#												
TST56	021134	2142#												
TST57	022012	2281#												
TST6	004772	858#												
TST7	005056	858#												
TTMG0	003006	744	752#											
TTMG1	003040	744	753#											
TTMG10	003516	746	760#											
TTMG2	003130	744	754#											
TTMG3	003216	744	755#											
TTMG4	003303	745	756#											
TTMG5	003342	745	757#											
TTMG6	003414	745	758#											
TTMG7	003460	745	759#											
TWOSP	042112	2507	3193#											
TYPDS =	104405	3117	3121	3128#										
TYPE =	104401	704	716	732	736	738	2379	2393	2407	2502	2507	2510	2526	2531
		2532	2534	2549	2551	2562	2564	2574	2576	2578	2583	2584	2585	2601
		2610	2611	2676	2677	2678	2679	2680	2683	2689	2765	2789	2816	2908
		2911	2925	2926	2952	2963	2983	3000	3005	3117	3120	3121	3123	3124
		3125	3126	3127	3128#	3129								
TVPOC =	104402	2506	2533	2956	2965	3120	3121	3124	3128#					
TVPON =	104404	3128#												
TVPOS =	104403	2550	2563	2609	2818	2909	3128#							















.EQUAT	235#	241
.HEADE	237#	239
.KT11	236#	244
.SETUP	237#	701
.SWRHI	235#	240
.SWRLO	240#	
.\$ACT1	238#	250
.\$APT8	238#	254#
.\$APTH	238#	252
.\$APTY	238#	3130
.\$CMTA	235#	254
.\$EOP	235#	3117
.\$ERRO	235#	3120
.\$ERRT	235#	3121
.\$POWE	236#	3129
.\$RDOC	237#	3123
.\$READ	237#	3124
.\$SCOP	236#	3119
.\$SIZE	236#	3122
.\$TRAP	236#	3128
.\$TYPD	236#	3127
.\$TYPE	237#	3125
.\$TYPO	236#	3126
.\$40CA	235#	245

. ABS. 042212 000

ERRORS DETECTED: 0

CRIIAB,CRIIAB/CRF CRIIAB.P11  
RUN-TIME: 128 92 TO SECONDS  
RUN-TIME RATIO: 1050/232=4.5  
CORE USED: 28K (55 PAGES)