

.MLIST SEQ,LOC,BIN,TOC
.REM_

IDENTIFICATION

PRODUCT CODE: AC-F172A-MC

PRODUCT NAME: CRIIAAO .IST STND-ALN SF

DATE CREATED: 15 NOVEMBER 1978

MAINTAINER: C.S.S. LOW VOLUME PRODUCTS GROUP

AUTHOR: ROBERT J. COLLINS / W. 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 (©) 1978 BY DIGITAL EQUIPMENT CORPORATION

1.0 ABSTRACT

THE DIP11-A STAND-ALONE 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 8K AND A CONSOLE TERMINAL
- B. DIP11-A (IIST) INTERFACE

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

IF A PAPER TAPE IS SUPPLIED, LOAD WITH AN ABS LOADER. IF 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
- SR00 (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).

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 HARDWARE CONFIGURATION).

7.0 OPERATION

ONCE STARTED, 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.

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

140

.LIST LOC,BIN,SEQ

```

148 .ENABLE ABS,AMA
153 .TITLE MAINDEC-11-CRIIA
(1) .*COPYRIGHT (C) 1977
(1) .*DIGITAL EQUIPMENT CORP.
(1) .*MAYNARD, MASS. 01754
(1) .*
(1) .*PROGRAM BY ROBERT J. COLLINS
(1) .*
(1) .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) .*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
(1) .*
154 .SBTTL OPERATIONAL SWITCH SETTINGS
(1) .*
(1) .* SWITCH USE
(1) .* -----
(1) .* 15 HALT ON ERROR
(1) .* 14 LOOP ON TEST
(1) .* 13 INHIBIT ERROR TYPEOUTS
(1) .* 11 INHIBIT ITERATIONS
(1) .* 10 BELL ON ERROR
(1) .* 9 LOOP ON ERROR
155 .SBTTL BASIC DEFINITIONS
(1) .*INITIAL ADDRESS OF THE STACK POINTER *** 1'00 ***
(1) 001100 STACK= 1100
(1) .EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
(1) .EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
(1) .*MISCELLANEOUS DEFINITIONS
(1) 000C11 HT= 11 ;;CODE FOR HORIZONTAL TAB
(1) 000012 LF= 12 ;;CODE FOR LINE FEED
(1) 000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
(1) 000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
(1) 177776 PS= 177776 ;;PROCESSOR STATUS WORD
(1) .EQUIV PS,PSW
(1) 177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
(1) 177772 PIRQ 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
(1) 177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
(1) 177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
(1) .*GENERAL PURPOSE REGISTER DEFINITIONS
(1) 000000 R0= %0 ;;GENERAL REGISTER
(1) 000001 R1 %1 ;;GENERAL REGISTER
(1) 000002 R2= %2 ;;GENERAL REGISTER
(1) 000003 R3= %3 ;;GENERAL REGISTER
(1) 000004 R4= %4 ;;GENERAL REGISTER
(1) 000005 R5= %5 ;;GENERAL REGISTER
(1) 000006 R6= %6 ;;GENERAL REGISTER
(1) 000007 R7 %7 ;;GENERAL REGISTER
(1) 000006 SP=R6 ;;STACK POINTER
(1) 000007 PC=R7 ;;PROGRAM COUNTER
(1) .*PRIORITY LEVEL DEFINITIONS
(1) 000000 PRO 0 ;;PRIORITY LEVEL 0
(1) 000040 PR1= 40 ;;PRIORITY LEVEL 1
(1) 000100 PR2= 100 ;;PRIORITY LEVEL 2

```

BASIC DEFINITIONS

(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 KVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
```

```

157
158      .SBTTL MEMORY MANAGEMENT DEFINITIONS
(1)
(1)      ;*KT11 VECTOR ADDRESS
(1)
(1)      000250      MMVEC= 250
(1)
(1)      ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1)      177572      SR0= 177572
(1)      177574      SR1= 177574
(1)      177576      SR2= 177576
(1)      172516      SR3  172516
(1)
(1)      ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS
(1)
(1)      172300      KIPDR0= 172300
(1)      172302      KIPDR1= 172302
(1)      172304      KIPDR2= 172304
(1)      172306      KIPDR3= 172306
(1)      172310      KIPDR4= 172310
(1)      172312      KIPDR5= 172312
(1)      172314      KIPDR6= 172314
(1)      172316      KIPDR7= 172316
(1)
(1)      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
(1)
(1)      172340      KIPAR0= 172340
(1)      172342      KIPAR1= 172342
(1)      172344      KIPAR2= 172344
(1)      172346      KIPAR3= 172346
(1)      172350      KIPAR4= 172350
(1)      172352      KIPAR5= 172352
(1)      172354      KIPAR6= 172354
(1)      172356      KIPAR7= 172356
(1)
159      .SBTTL TRAP CATCHER
(1)
(1)      000000      .-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)      ;*      PC=YYYYYY UNEXPECTED TRAP TO XXX
(1)      ;*AND RETURN TO THE PROGRAM AT PC-YYYYYY+2
(1)      ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
(1)      ;*      YYYYYY=PC AT TIME OF TRAP
(1)      ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
(1)      ;*      CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.
(1)
(1)      000000 000000      $40CAT: HALT      ;:HALT
(1)      000002 000737      BR      .-100      ;:BRANCH TO 177700 & TIME OUT (NOT ON
(1)                                     ;:11/05)

```



```
(1) 000004 002502          .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 ADDRESS(ES)
(1) 000200 000137 002502  JMP      @NOPAR ;;GO TO START OF PROGRAM
160 000204 000137 002512  JMP      SELPAR    ;;GO TO SELECT PARAMETERS
161
```

```

163
164 .SBTTL ACT11 HOOKS
(1)
(2) ::*****
(1) :HOOKS REQUIRED BY ACT11
(1) 000210 $SVPC= ;SAVE PC
(1) 000046 .=46
(1) 000046 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1) 000052 .=52
(1) 000052 .WORD 0 ;;2)SET LOC.52 TO ZERO
(1) 000210 .=$SVPC ;; RESTORE PC
165 -1000
166 .SBTTL APT PARAMETER BLOCK
(1)
(2) ::*****
(1) :SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2) ::*****
(1) 001000 .SX- ;;SAVE CURRENT LOCATION
(1) 000024 -24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 200 ;;FOR APT START UP
(1) 000044 .44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 $APTHDR ;;POINT TO APT HEADER BLOCK
(1) 001000 .SX ;;RESET LOCATION COUNTER
(2) ::*****
(1) :SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1) :INTERFACE SPEC.
(1)
(1) $APTHD:
(1) 001000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 001002 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 001004 $STMT: .WORD 15. ;;RUN TIM OF LONGEST TEST
(1) 001006 $PASTM: .WORD 60. ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 001010 $UNITM: .WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 001012 .WORD $END-$MAIL/2 ;;LENGTH MAILBOX-TABLE (WORDS)

```



```
(1) 001225 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
(1) 001226 000012 $LF: .ASCIIZ <12> ::LINE FEED
(2) ::*****
(2) $BTTL APT MAILBOX-ETABLE
(2) ::*****
(2) .EVEN
(2) $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 $FTABLE: ::APT ENVIRONMENT TABLE
(2) 001250 000 $ENV: .BYTE AENV ::ENVIRONMENT BYTE
(2) 001251 000 $FNVM: .BYTE AENVM ::ENVIRONMENT MODE BITS
(2) 001252 000000 $SWREG: .WORD ASWREG ::APT SWITCH REGISTER
(2) 001254 000000 $USWR: .WORD AUSWR ::JSER SWITCHES
(2) 001256 000000 $CPUOP: .WORD ACPUOP ::CPU TYPE,OPTIONS
(2) * BIT 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 $DEV: .WORD ADEV ::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
```

(2)	001326	000000	\$DDW5:	.WORD	ADDW5	::DEVICE	DESCRIPTOR	WORD#5
(2)	001330	000000	\$DDW6:	.WORD	ADDW6	::DEVICE	DESCRIPTOR	WORD#6
(2)	001332	000000	\$DDW7:	.WORD	ADDW7	::DEVICE	DESCRIPTOR	WORD#7
(2)	001334	000000	\$DDW8:	.WORD	ADDW8	::DEVICE	DESCRIPTOR	WORD#8
(2)	001336	000000	\$DDW9:	.WORD	ADDW9	::DEVICE	DESCRIPTOR	WORD#9
(2)	001340	000000	\$DDW10:	.WORD	ADDW10	::DEVICE	DESCRIPTOR	WORD#10
(2)	001342	000000	\$DDW11:	.WORD	ADDW11	::DEVICE	DESCRIPTOR	WORD#11
(2)	001344	000000	\$DDW12:	.WORD	ADDW12	::DEVICE	DESCRIPTOR	WORD#12
(2)	001346	000000	\$DDW13:	.WORD	ADDW13	::DEVICE	DESCRIPTOR	WORD#13
(2)	001350	000000	\$DDW14:	.WORD	ADDW14	::DEVICE	DESCRIPTOR	WORD#14
(2)	001352	000000	\$DDW15:	.WORD	ADDW15	::DEVICE	DESCRIPTOR	WORD#15
(2)								
(2)								
(2)	001354		\$FEND:					
(2)								

```

(1) .SBTTL ERROR POINTER TABLE
(1)
(3) ;*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 $ERRTB:
169 ;ERROR PARAMETER TABLE
170
171 ;ERROR 1
172 001354 030000 EM1 ;ACR BIT3-0 LOAD-READER ERROR
173 001356 032635 DH1 ;PASS PC EXPCTD ACTUAL ACR
174 001360 033712 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
175 001362 033760 DAF1
176
177 ;ERROR 2
178 001364 030033 EM2 ;ACR BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
179 001366 032701 DH2 ;PASS PC ACR
180 001370 033726 DT2 ;$PASS,$ERRPC,DISPLY
181 001372 033760 DAF1
182
183 ;ERROR 3
184 001374 030473 EM3 ;PGTE BIT11-8 LOAD-READ ERROR
185 001376 032752 DH4 ;PASS PC EXPCTD ACUTAL PGTE
186 001400 033712 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
187 001402 033760 DAF1
188
189 ;ERROR 4
190 001404 030530 EM4 ;PGTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
191 001406 032725 DH3 ;PASS PC PGTE
192 001410 033726 DT2 ;$PASS,$ERRPC,DISPLY
193 001412 033760 DAF1
194
195 ;ERROR 5
196 001414 030611 EM5 ;PGTE BIT3-0 LOAD-READ ERROR
197 001416 032752 DH4 ;PASS PC EXPCTD ACUTAL PGTE
198 001420 033712 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
199 001422 033760 DAF1
200
201 ;ERROR 6
202 001424 030645 EM6 ;PGTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
203 001426 032725 DH3 ;PASS PC PGTE
204 001430 033726 DT2 ;$PASS,$ERRPC,DISPLY
205 001432 033760 DAF1
  
```

207			:ERROR	7	
208	001434	030112		EM7	:IE (PGTE BIT 6) CANNOT BE SET
209	001436	032725		DH3	:PASS PC PGTE
210	001440	033726		DT2	:\$PASS,\$ERRPC,DISPLY
211	001442	033760		DAF1	
212					
213			:ERROR	10	
214	001444	030147		EM10	:IE (PGTE BIT_) CANNOT BE WRITTEN TO 0
215	001446	032725		DH3	:PASS PC PGTE
216	001450	033712		DT1	:\$PASS,\$ERRPC,DISPLY
217	001452	033760		DAF1	
218					
219			:ERROR	11	
220	001454	030215		EM11	:IE (PGTE BIT6) CANNOT BE CLEARED BY RESET (ACR CLR)
221	001456	032725		DH3	:PASS PC PGTE
222	001460	033726		DT2	:\$PASS,\$ERRPC,DISPLY
223	001462	033760		DAF1	
224					
225			:ERROR	12	
226	001464	030301		EM12	:PTP (PGTE BIT1) CANNOT BE SET
227	001466	032725		DH3	:PASS PC PGTE
228	001470	033726		DT2	:\$PASS,\$ERRPC,DISPLY
229	001472	033760		DAF1	
230					
231			:ERROR	13	
232	001474	030337		EM13	:PTP (PGTE BIT1) CANNOT BE WRITTEN TO 0
233	001476	032725		DH3	:PASS PC PGTE
234	001500	033726		DT2	:\$PASS,\$ERRPC,DISPLY
235	001502	033760		DAF1	
236					
237			:ERROR	14	
238	001504	030406		EM14	:PTP (PGTE BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)
239	001506	032725		DH3	:PASS PC PGTE
240	001510	033726		DT2	:\$PASS,\$ERRPC,DISPLY
241	001512	033760		DAF1	
242					
243			:ERROR	15	
244	001514	030725		EM15	:STTE BIT11-8 LOAD-READ ERROR
245	001516	033017		DH5	:PASS PC EXPCTD ACUTAL STTE
246	001520	033712		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
247	001522	033760		DAF1	
248					
249			:ERROR	16	
250	001524	030762		EM16	:STTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
251	001526	033064		DH6	:PASS PC STTE
252	001530	033726		DT2	:\$PASS,\$ERRPC,DISPLY
253	001532	033760		DAF1	
254					
255			:ERROR	17	
256	001534	031043		EM17	:STTE BIT3-0 LOAD-READ ERROR
257	001536	033017		DH5	:PASS PC EXPCTD ACUTAL STTE
258	001540	033712		DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
259	001542	033760		DAF1	

261			:ERROR 20	
262	001544	031077	EM20	:STTE BIT3-0 WERE NOT CLEARED BY RESET (ACR·CLR)
263	001546	033064	DH6	:PASS PC STTE
264	001550	033726	DT2	:\$PASS,\$ERRPC,DISPLY
265	001552	033760	DAF1	
266				
267			:ERROR 21	
268	001554	031157	EM21	:LKE (STCS BIT2) CANNOT BE SET
269	001556	033111	DH7	:PASS PC STCS
270	001560	033726	DT2	:\$PASS,\$ERRPC,DISPLY
271	001562	033760	DAF1	
272				
273			:ERROR 22	
274	001564	031215	EM22	:LKE (STCS BIT2) CANNOT BE WRITTEN TO 0
275	001566	033111	DH7	:PASS PC STCS
276	001570	033726	DT2	:\$PASS,\$ERRPC,DISPLY
277	001572	033760	DAF1	
278				
279			:ERROR 23	
280	001574	031264	EM23	:LKE (STCS BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)
281	001576	033111	DH7	:PASS PC STCS
282	001600	033726	DT2	:\$PASS,\$ERRPC,DISPLY
283	001602	033760	DAF1	
284				
285			:ERROR 24	
286	001604	031351	EM24	:STP (STCS BIT1) CANNOT BE SET
287	001606	033111	DH7	:PASS PC STCS
288	001610	033726	DT2	:\$PASS,\$ERRPC,DISPLY
289	001612	033760	DAF1	
290				
291			:ERROR 25	
292	001614	031407	EM25	:STP (STCS BIT1) CANNOT BE WRITTEN TO 0
293	001616	033111	DH7	:PASS PC STCS
294	001620	033726	DT2	:\$PASS,\$ERRPC,DISPLY
295	001622	033760	DAF1	
296				
297			:ERROR 26	
298	001624	031456	EM26	:STP (STCS BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)
299	001626	033111	DH7	:PASS PC STCS
300	001630	033726	DT2	:\$PASS,\$ERRPC,DISPLY
301	001632	033760	DAF1	
302				
303			:ERROR 27	
304	001634	031543	EM27	:ENB (STCS BIT0) CANNOT BE SET
305	001636	033111	DH7	:PASS PC STCS
306	001640	033726	DT2	:\$PASS,\$ERRPC,DISPLY
307	001642	033760	DAF1	
308				
309			:ERROR 30	
310	001644	031601	EM30	:ENR (STCS BIT0) CANNOT BE WRITTEN TO 0
311	001646	033111	DH7	:PASS PC STCS
312	001650	033726	DT2	:\$PASS,\$ERRPC,DISPLY
313	001652	033760	DAF1	

315			:ERROR 31	
316	001654	031650	EM31	:ENB (STCS BIT0) CANNOT BE CLEARED BY RESET (ACR CLR)
317	001656	033111	DH7	:PASS PC STCS
318	001660	033726	DT2	:\$PASS,\$ERRPC,DISPLY
319	001662	033760	DAF1	
320				
321			:ERROR 32	
322	001664	031735	EM32	:STCS BIT15-8 LOAD-READ ERROR
323	001666	033136	DH8	:PASS PC EXPCTD ACUTAL STCS
324	001670	033712	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
325	001672	033760	DAF1	
326				
327			:ERROR 33	
328	001674	031772	EM33	:STCS BIT15-8 WERE NOT CLEARED BY RESET (ACR CLR)
329	001676	033111	DH7	:PASS PC STCS
330	001700	033726	DT2	:\$PASS,\$ERRPC,DISPLY
331	001702	033760	DAF1	
332				
333			:ERROR 34	
334	001704	032053	EM34	:IMSK BIT11-8 LOAD-READ ERROR
335	001706	033203	DH9	:PASS PC EXPCTD ACUTAL IMSK
336	001710	033712	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
337	001712	033760	DAF1	
338				
339			:ERROR 35	
340	001714	032110	EM35	:IMSK BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
341	001716	033250	DH10	:PASS PC IMSK
342	001720	033726	DT2	:\$PASS,\$ERRPC,DISPLY
343	001722	033760	DAF1	
344				
345			:ERROR 36	
346	001724	032171	EM36	:IMSK BIT3-0 LOAD-READ ERROR
347	001726	033203	DH9	:PASS PC EXPCTD ACUTAL IMSK
348	001730	033712	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
349	001732	033760	DAF1	
350				
351			:ERROR 37	
352	001734	032225	EM37	:IMSK BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
353	001736	033250	DH10	:PASS PC IMSK
354	001740	033726	DT2	:\$PASS,\$ERRPC,DISPLY
355	001742	033760	DAF1	
356				
357			:ERROR 40	
358	001744	032305	EM40	:MTCE BIT11-8 LOAD-READ ERROR
359	001746	033275	DH11	:PASS PC EXPCTD ACUTAL MTCE
360	001750	033712	DT1	:\$PASS,\$ERRPC,\$GDDAT,\$BDDAT,DISPLY
361	001752	033760	DAF1	
362				
363			:ERROR 41	
364	001754	032341	EM41	:MTCE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)
365	001756	033342	DH12	:PASS PC MTCE
366	001760	033726	DT2	:\$PASS,\$ERRPC,DISPLY
367	001762	033760	DAF1	

```

369 ;ERROR 42
370 001764 032421 EM42 ;MTCE BIT3-0 LOAD-READ ERROR
371 001766 033275 DH11 ;PASS PC EXPCTD ACUTAL MTCE
372 001770 033712 DT1 ;$PASS,$ERRPC,$GDDAT,$BDDAT,DISPLY
373 001772 033760 DAF1
374
375 ;ERROR 43
376 001774 032454 EM43 ;MTCE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)
377 001776 033342 DH12 ;PASS PC MTCE
378 002000 033726 DT2 ;$PASS,$ERRPC,DISPLY
379 002002 033760 DAF1
380 ;ERROR 44
381 002004 032533 EM44 ;IIST ERROR - REFER TO THE LISTING AT 'PC'
382 ;TO DETERMINE THE ERROR.
383 002006 033367 DH13 ;PASS PC LINE # EXPCTD ACTUAL DISPLY $TMP0 $TMP1
384 002010 033736 D*3 ;$PASS,$ERRPC,LINMBR,$GDDAT,$BDDAT,DISPLY,$TMP0,$TMP1
385 002012 033760 DAF1
  
```

.SBTTL REGISTER DEFINITIONS

:ALR BIT DEFINITIONS

```

389
390
391 100000 CLR=BIT15 ;MASTER CLEAR
392 000000 PGTEE=0
393 000001 PGCSE=1
394 000002 STTEE=2
395 000003 STCSE=3
396 000004 IMSKE=4
397 000005 PGFE=5
398 000006 STFE=6
399 000007 DCFE=7
400 000010 EXCE=10
401 000015 MTCE=15
  
```

:PGTE BIT DEFINITION

```

402
403
404
405 004000 PB3=BIT11 ;PROGRAM BOOT TRANSMIT ENABLE 3
406 002000 PB2=BIT10 ;PROGRAM BOOT TRANSMIT ENABLE 2
407 001000 PB1=BIT9 ;PROGRAM BOOT TRANSMIT ENABLE 1
408 000400 PB0=BIT8 ;PROGRAM BOOT TRANSMIT ENABLE 0
409 000010 PI3=BIT3 ;PROGRAM INTERRUPT TRANSMIT ENABLE 3
410 000004 PI2=BIT2 ;PROGRAM INTERRUPT TRANSMIT ENABLE 2
411 000002 PI1=BIT1 ;PROGRAM INTERRUPT TRANSMIT ENABLE 1
412 000001 PI0=BIT0 ;PROGRAM INTERRUPT TRANSMIT ENABLE 0
413
  
```

:PGCS BIT DEFINITIONS

```

414
415
416 100000 ERR=BIT15 ;ERROR
417 040000 GRJ=BIT14 ;GO REJECT
418 020000 PGMR=BIT13 ;PGTE MODIFICATION REFUSED
419 010000 STMR=BIT12 ;SITE MODIFICATION REFUSED
420 004000 PRDY=BIT11 ;PG READY
421 000000 SID0=0 ;SELF ID 0
422 000400 SID1=BIT8 ;SELF ID 1
423 001000 SID2=BIT9 ;SELF ID 2
424 001400 SID3=BIT9+BIT8 ;SELF ID 3
  
```

```

425      000010      IP=BIT3      :INTERRUPT PENDING IP=ERR+FLAGS
426      000004      IE=BIT2      :IIST INTERRUPT ENABLE
427      000002      PTP=BIT1     :PGTE PARITY BIT
428      000001      GO=BIT0      :GENERATE BOOT/INTERRUPT
429
430      ;STTE BIT DEFINITIONS
431
432      004000      SB3=BIT11     :SANITY TIMER BOOT TRANSMIT ENABLE 3
433      002000      SB2=BIT10     :SANITY TIMER BOOT TRANSMIT ENABLE 2
434      001000      SB1=BIT9      :SANITY TIMER BOOT TRANSMIT ENABLE 1
435      000400      SB0=BIT8      :SANITY TIMER BOOT TRANSMIT ENABLE 0
436      000010      SI3=BIT3      :SANITY TIMER INTERRUPT TRANSMIT ENABLE 3
437      000004      SI2=BIT2      :SANITY TIMER INTERRUPT TRANSMIT ENABLE 2
438      000002      SI1=BIT1      :SANITY TIMER INTERRUPT TRANSMIT ENABLE 1
439      000001      SI0=BIT0      :SANITY TIMER INTERRUPT TRANSMIT ENABLE 0
440
441      ;STCS BIT DEFINITIONS
442
443      000010      TMO=BIT3      :TIMER EXPIRED
444      000004      LKE=BIT2      :LOCKUP ENABLE
445      000002      STP=BIT1      :STTE PARITY BIT
446      000001      ENB=BIT0      :SANITY TIMER ENABLE
447
448      ;IMSK BIT DEFINITIONS
449
450      004000      BM3=BIT11     :BOOT INHIBIT 3
451      002000      BM2=BIT10     :BOOT INHIBIT 2
452      001000      BM1=BIT9      :BOOT INHIBIT 1
453      000400      BM0=BIT8      :BOOT INHIBIT 0
454      000010      IM3=BIT3      :INTERRUPT INHIBIT 3
455      000004      IM2=BIT2      :INTERRUPT INHIBIT 2
456      000002      IM1=BIT1      :INTERRUPT INHIBIT 1
457      000001      IM0=BIT0      :INTERRUPT INHIBIT 0
458
459      ;PGF BIT DEFINITIONS
460
461      004000      PBF3=BIT11     :PROGRAM GENERATED BOOT FLAG 3
462      002000      PBF2=BIT10     :PROGRAM GENERATED BOOT FLAG 2
463      001000      PBF1=BIT9      :PROGRAM GENERATED BOOT FLAG 1
464      000400      PBF0=BIT8      :PROGRAM GENERATED BOOT FLAG 0
465      000010      PIF3=BIT3      :PROGRAM GENERATED INTERRUPT FLAG 3
466      000004      PIF2=BIT2      :PROGRAM GENERATED INTERRUPT FLAG 2
467      000002      PIF1=BIT1      :PROGRAM GENERATED INTERRUPT FLAG 1
468      000001      PIF0=BIT0      :PROGRAM GENERATED INTERRUPT FLAG 0
469
470      ;STF BIT DEFINITIONS
471
472      000010      SBF3=BIT3      :SANITY TIMER GENERATED BOOT FLAG 3
473      000004      SBF2=BIT2      :SANITY TIMER GENERATED BOOT FLAG 2
474      000002      SBF1=BIT1      :SANITY TIMER GENERATED BOOT FLAG 1
475      000001      SBF0=BIT0      :SANITY TIMER GENERATED BOOT FLAG 0
476      000010      SIF3=BIT3      :SANITY TIMER GENERATED INTERRUPT FLAG 3
477      000004      SIF2=BIT2      :SANITY TIMER GENERATED INTERRUPT FLAG 2
478      000002      SIF1=BIT1      :SANITY TIMER GENERATED INTERRUPT FLAG 1
479      000001      SIF0=BIT0      :SANITY TIMER GENERATED INTERRUPT FLAG 0
480
  
```

```
481 ;DCF BIT DEFINITIONS
482
483 004000 BRK3=BIT11 ;DISCONNECT OR POWER LOSS STATE OF IIST 3
484 002000 BRK2=BIT10 ;DISCONNECT OR POWER LOSS STATE OF IIST 2
485 001000 BRK1=BIT9 ;DISCONNECT OR POWER LOSS STATE OF IIST 1
486 000400 BRK0=BIT8 ;DISCONNECT OR POWER LOSS STATE OF IIST 0
487 000010 DCF3=BIT3 ;DCLO/DISCONNECT INTERRUPT FLAG 3
488 000004 DCF2=BIT2 ;DCLO/DISCONNECT INTERRUPT FLAG 2
489 000002 DCF1=BIT1 ;DCLO/DISCONNECT INTERRUPT FLAG 1
490 000001 DCF0=BIT0 ;DCLO/DISCONNECT INTERRUPT FLAG 0
491
492 ;EXC BIT DEFINITIONS
493
494 004000 UI3=BIT11 ;UNEXPECTED VALID INPUT FLAG 3
495 002000 UI2=BIT10 ;UNEXPECTED VALID INPUT FLAG 2
496 001000 UI1=BIT9 ;UNEXPECTED VALID INPUT FLAG 1
497 000400 UI0=BIT8 ;UNEXPECTED VALID INPUT FLAG 0
498 000010 RTE3=BIT3 ;UNEXPECTED INVALID INPUT FLAG 3
499 000004 RTE2=BIT2 ;UNEXPECTED INVALID INPUT FLAG 2
500 000002 RTE1=BIT1 ;UNEXPECTED INVALID INPUT FLAG 1
501 000001 RTE0=BIT0 ;UNEXPECTED INVALID INPUT FLAG 0
502
503 ;MTC BIT DEFINITIONS
504
505 004000 MTYP=BIT11 ;MNT TYPE BIT.0=PG; 1=ST
506 002000 MFRM=BIT10 ;MNT FRAME BIT.0=NORMAL; 1=FRAMING ERROR
507 000000 MID0=0 ;MNT ID 0
508 000400 MID1=BIT8 ;MNT ID 1
509 001000 MID2=BIT9 ;MNT ID 2
510 001400 MID3=BIT9+BIT8 ;MNT ID 3
511 000010 DSBT=BIT3 ;DISABLE BOOT
512 000004 MENB=BIT2 ;ENABLE BIT-8 OF MNT
513 000002 MLEN=BIT1 ;ENABLE MAINT. LOOP (IDLE XMIT DRIVERS)
514 000001 MDSD=BIT0 ;DISABLE XMIT DRIVER (CAUSE A BREAK)
515 000016 TMT=DSBT+MENB+MLEN
516
517 ;OTHER DEFINITIONS
518
519 005726 POP 5726
520 022626 POPPOP 22626
521 000200 CRLF 200
```

523
524
570

.SBTTL TEST MACRO DEFINITIONS

```

611 002014
(1)
(1)
(1) 002014 012706 001100
(1) 002020 005026
(1) 002022 022706 001140
(1) 002026 001374
(1) 002030 012706 001100
(1)
(1) 002034 012737 024012 000020
(1) 002042 012737 000340 000022
(1) 002050 012737 024264 000030
(1) 002056 012737 000340 000032
(1) 002064 012737 027272 000034
(1) 002072 012737 000340 000036
(1) 002100 012737 027354 000024
(1) 002106 012737 000340 000026
(1) 002114 013737 023724 023716
(1) 002122 005037 001214
(1) 002126 005037 001216
(1) 002132 112737 000001 001115
(1) 002140 012737 002140 001106
(1) 002146 012737 002146 001110
(2)
(2)
(2) 002154 013746 000004
(2) 002160 012737 002214 000004
(2) 002166 012737 177570 001140
(2) 002174 012737 177570 001142
(2) 002202 022777 177777 176730
(2) 002210 001012
(2)
(2) 002212 000403
(2) 002214 012716 002222 64$:
(2) 002220 000002
(2) 002222 012737 000176 001140 65$:
(2) 002230 012737 000174 001142
(2) 002236 012637 000004 66$:
(1)
(2) 002242 005037 001236
(2) 002246 132737 000200 001251
(2) 002254 001403
(2) 002256 012737 001252 001140
(2) 002264
612
(1)
(1) 002264 005227 177777
(1) 002270 001062
(1) 002272 022737 023756 000042
(1) 002300 001456
(1) 002302 104401 002350
(2)
(2) 002306 005737 000042
(2) 002312 001012
(2) 002314 123727 001250 000001
(2) 002322 001406

```

```

START:
.SBTTL INITIALIZE THE COMMON TAGS
::CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV # $CMTAG,R6 ::FIRST LOCATION TO BE CLEARED
CLR (R6)+ ::CLEAR MEMORY LOCATION
CMP #SWR,R6 ::DONE?
BNE -6 ::LOOP BACK IF NO
MOV #STACK,SP ::SETUP THE STACK POINTER
::INITIALIZE A FEW VECTORS
MOV # $SCOPE,@#IOTVEC ::IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@#IOTVEC+2 ::LEVEL 7
MOV # $ERROR,@#EMTVEC ::EMT VECTOR FOR ERROR ROUTINE
MOV #340,@#EMTVEC+2 ::LEVEL 7
MOV # $TRAP,@#TRAPVEC ::TRAP VECTOR FOR TRAP CALLS
MOV #340,@#TRAPVEC+2 ::LEVEL 7
MOV # $PWRDN,@#PWRVEC ::POWER FAILURE VECTOR
MOV #340,@#PWRVEC+2 ::LEVEL 7
MOV $ENDCT,$EOPCT ::SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ::INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ::CLEAR THE ESCAPE ON ERROR ADDRESS
MOV# #1,$ERMAX ::ALLOW ONE ERROR PER TEST
MOV #.,,$LPADR ::INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #.,,$LPERR ::SETUP THE ERROR LOOP ADDRESS
::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
::EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @#ERRVEC,-(SP) ::SAVE ERROR VECTOR
MOV #64,$@#ERRVEC ::SET UP ERROR VECTOR
MOV #DSWR,$SWR ::SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,$DISPLAY ::AND A HARDWARE DISPLAY REGISTER
CMP #-1,@#SWR ::TRY TO REFERENCE HARDWARE SWR
BNE 66$ ::BRANCH IF NO TIMEOUT TRAP OCCURRED
::AND THE HARDWARE SWR IS NOT -1
BR 65$ ::BRANCH IF NO TIMEOUT
MOV #65,$(SP) ::SET UP FOR TRAP RETURN
RTI
MOV #SWREG,$SWR ::POINT TO SOFTWARE SWR
MOV #DISPREG,$DISPLAY
MOV (SP)+,@#ERRVEC ::RESTORE ERROR VECTOR
CLR $PASS ::CLEAR PASS COUNT
BITB #APTSIZE,$ENVM ::TEST USER SIZE UNDER APT
BEQ 67$ ::YES,USE NON-APT SWITCH
MOV # $SWREG,$SWR ::NO,USE APT SWITCH REGISTER
67$:
.SBTTL TYPE PROGRAM NAME
::TYPE THE NAME OF THE PROGRAM IF FIRST PASS
INC #-1 ::FIRST TIME?
BNE 68$ ::BRANCH IF NO
CMP # $ENDAD,@#42 ::ACT-11?
BEQ 68$ ::BRANCH IF YES
TYPE ,69$ ::TYPE ASCIZ STRING
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42 ::ARE WE RUNNING UNDER XXDP/ACT?
BNE 70$ ::BRANCH IF YES
CMP# $L.V,#1 ::ARE WE RUNNING UNDER APT?
BEQ 70$ ::BRANCH IF YES

```

```

(2) 002324 023727 001140 000176      CMP      SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
(2) 002332 001005                      BNE      71$            ;;BRANCH IF NO
(2) 002334 104406                      GTSWR                      ;;GET SOFT-SWR SETTINGS
(2) 002336 000403                      BR       71$
(2) 002340 112737 000001 001134 70$:  MOVB     #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
(2) 002346 000433                      BR       68$
(1) 002346 000433                      BR       68$
(1) 002436                      ;;69$: .ASCIZ <CRLF>#DIP11-A (IIST) MONOPORT DIAGNOSTIC MAINDEC-11-CRIIA#<CRLF>
(1) 002436                      68$:
613 002436 012737 000006 000004      MOV      #6,4          ;SET TO TRAP
614 002444 012737 000004 000006      MOV      #4,6          ;ON AN NXM
615                      ; JSR     PC,$SIZE      ;SIZE MEMORY
616 002452 005737 023656                      TST     PARFLG ;SEE IF PARAMETERS TO BE INPUT
617 002456 001402                      BEQ     1$            ;SKIP IF NOT
618 002460 004737 022336                      JSR     PC,GETPAR     ;GET PARAMETERS
619 002464 012737 000340 177776 1$:  MOV      #340,PSW      ;RISE PRIORITY
620 002472 005037 023662                      CLR     STCNT         ;INITIALIZE S.T. COUNT VALUE.
621 002476 000137 002524                      JMP     BEGIN         ;GO START TESTING
622
623
624                      ;COME HERE ON START AT 200
625 002502 005037 023656      NOPAR: CLR     PARFLG  ;CLEAR PARAMETER FLAG
626 002506 000137 002014                      JMP     START        ;GO TO STARTUP
627
628                      ;COME HERE ON START AT 204
629 002512 012737 000001 023656  SELPAR: MOV    #1,PARFLG ;SET PARAMETER FLAG
630 002520 000137 002014                      JMP     START
631
632 002524                      BEGIN:
  
```

```

634
635
636
637
638
639
(3)
(3)
(2) 002524 000004
640 002526 012777 100000 021036
641 002534 012737 002550 001110
642 002542 012737 000017 001124
643 002550 013777 001124 021014
644 002556 017737 021010 023610
645 002564 013737 023610 001126
646 002572 042737 177760 001126
647 002600 023737 001124 001126
648 002606 001403
649 002610 104001
650 002612 004737 022234
651 002616 005237 001124
652 002622 042737 177760 001124
653 002630 023727 001124 000017
654 002636 001344
655 002640 012737 002646 001110
656 002646 012777 000017 020710
657 002654 012777 100000 020710
658 002662 017737 020704 023610
659 002670 032737 000017 023610
660 002676 001403
661 002700 104002
662 002702 004737 022234
    
```

```

.SBTTL
.SBTTL
.SBTTL BASIC REGISTER TESTS
.SBTTL
    
```

```

:*****
:*TEST 1 ACR (BIT3-0) WRITE-READ TEST USING A COUNT PATTERN
:*****
TST1: SCOPE
MOV #CLR,@ACR ;CLEAR THE WORLD
MOV #1$, $LPERR ;SET SCOPE LOOP POINTER
MOV #17,$GDDAT ;INIT EXPECTED TO 17
1$: MOV $GDDAT,@ACR ;LOAD ACR WITH $GDDAT
MOV @ACR,DISPLY ;READ ACR
MOV DISPLY,$BDDAT ;COPY ACR 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+ 1 ;ACR BIT3-0 LOAD-READ ERROR
JSR PC,DMPREG
2$: INC $GDDAT ;BUMP EXPECTED
BIC #-<17+1>,$GDDAT ;MASK TO BITS UNDER TEST
CMP $GDDAT,#17 ;DONE ALL?
BNE 1$ ;TEST AGAIN IF NO
MOV #3$, $LPERR ;CHANGE LOOP CONTROL
3$: MOV #17,@ACR ;SET ALL BITS
MOV #CLR,@ACR ;CLEAR THE WORLD
MOV @ACR,DISPLY ;READ ACR
BIT #17,DISPLY ;CHECK BIT3-0 IN ACR FOR 0
BEQ TST2 ;SKIP IF RESET CLEARED BIT3-0 IN ACR
ERROR+ 2 ;BIT3-0 IN ACR WERE NOT CLEARED BY RESET
.SR PC,DMPREG
    
```



```

667
(4)
(4)
(3) 002706 000004
(1) 002710 012777 100000 020654
(1) 002716 012737 002732 001110
(1) 002724 012737 007400 001124
(1) 002732 012777 000000 020632
(1) 002740 013777 001124 020630
(1) 002746 012777 000000 020616
(1) 002754 017737 020616 023610
(1) 002762 013737 023610 001126
(1) 002770 042737 170377 001126
(1) 002776 023737 001124 001126
(1) 003004 001403
(1) 003006 104003
(1) 003010 004737 022234
(1) 003014 062737 000400 001124
(1) 003022 042737 170377 001124
(1) 003030 023727 001124 007400
(1) 003036 001335
(1) 003040 012737 003046 001110
(1) 003046 012777 000000 020516
(1) 003054 012777 007400 020514
(1) 003062 012777 100000 020502
(1) 003070 012777 000000 020474
(1) 003076 017737 020474 023610
(1) 003104 032737 007400 023610
(1) 003112 001403
(1) 003114 104004
(1) 003116 004737 022234
(1) 003122

*****
*TEST 2 PGTE (BIT11-8) WRITE-READ TEST USING PGTE COUNT PATTERN
*****
TST2: SCOPE
MOV #CLR,@ACR ;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
3$: MOV #3$, $LPERR ;CHANGE LOOP CONTROL
MOV #PGTEE,@ACR ;RESET ADR TO PGTE
MOV #7400,@PGTE ;SET ALL BITS
MOV #CLR,@ACR ;CLEAR THE WORLD
MOV #PGTEE,@ACR ;SET ADDR TO PGTE
MOV @PGTE,DISPLY ;READ PGTE
BIT #7400,DISPLY ;CHECK BIT11-8 IN PGTE FOR 0
BEQ 4$ ;SKIP IF RESET CLEARED BIT11-8 IN PGTE
ERROR+ N ;BIT11-8 IN PGTE WERE NOT CLEARED BY RESET
4$: JSR PC,DMPREG
  
```

```

669
(4)
(4)
(3) 003122 000004
(1) 003124 012777 100000 020440
(1) 003132 012737 003146 001110
(1) 003140 012737 000017 001124
(1) 003146 012777 000000 020416
(1) 003154 013777 001124 020414
(1) 003162 012777 000000 020402
(1) 003170 017737 020402 023610
(1) 003176 013737 023610 001126
(1) 003204 042737 177760 001126
(1) 003212 023737 001124 001126
(1) 003220 001403
(1) 003222 104005
(1) 003224 004737 022234
(1) 003230 062737 000001 001124
(1) 003236 042737 177760 001124
(1) 003244 023727 001124 000017
(1) 003252 001335
(1) 003254 012737 003262 001110
(1) 003262 012777 000000 020302
(1) 003270 012777 000017 020300
(1) 003276 012777 100000 020266
(1) 003304 012777 000000 020260
(1) 003312 017737 020260 023610
(1) 003320 032737 000017 023610
(1) 003326 001403
(1) 003330 104006
(1) 003332 004737 022234
(1) 003336
  
```

```

*****
: *TEST 3 PGTE (BIT3-0) WRITE-READ TEST USING PGTE COUNT PATTERN
*****
TST3: SCOPE
      MOV #CLR,@ACR ;CLEAR THE WORLD
      MOV #1$, $LPERR ;SET SCOPE LOOP POINTER
      MOV #17,$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
      MOV #3$, $LPERR ;CHANGE LOOP CONTROL
3$:   MOV #PGTEE,@ACR ;RESET ADR TO PGTE
      MOV #17,@PGTE ;SET ALL BITS
      MOV #CLR,@ACR ;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$:
  
```

671

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

: *TEST 4 TEST THAT IE (PGCS BIT2) CAN BE SET
:*****

TST4: SCOPE
MOV #CLR,@ACR ;CLEAR THE SLATE
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 TST5 ;SKIP IF IE IS SET IN PGCS
ERROR+ N ;IE (PGCS BIT2) CANNOT BE SET
JSR PC,DMPREG

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

TST5: 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 TST6 ;SKIP IF IE IS CLEAR IN PGCS
ERROR+ N ;IE (PGCS BIT2) CANNOT BE CLEARED
JSR PC,DMPREG
;BY WRITING IT TO 0

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

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

673

(4)

(4)

(3) 003556 000004
(1) 003560 012777 100000 020004
(1) 003566 012777 000001 017776
(1) 003574 012777 000002 017774
(1) 003602 012777 000001 017762
(1) 003610 017737 017762 023610
(1) 003616 032737 000002 023610
(3) 003624 001003
(1) 003626 104012
(1) 003630 004737 022234

```
*****  
*TEST 7 TEST THAT PTP (PGCS BIT1) CAN BE SET  
*****  
TST7: SCOPE  
MOV #CLR,@ACR ;CLEAR THE SLATE  
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 TST10 ;SKIP IF PTP IS SET IN PGCS  
ERROR+ N ;PTP (PGCS BIT1) CANNOT BE SET  
JSR PC,DMPREG
```

(1)

(5)

(4)

(4)

(3) 003634 000004
(1) 003636 012777 000001 017726
(1) 003644 012777 000002 017724
(1) 003652 012777 000001 017712
(1) 003660 042777 000002 017710
(1) 003666 012777 000001 017676
(1) 003674 017737 017676 023610
(1) 003702 032737 000002 023610
(3) 003710 001403
(1) 003712 104013
(1) 003714 004737 022234

```
*****  
*TEST 10 TEST THAT PTP (PGCS BIT1) CAN BE CLEARED  
*****  
TST10: 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 TST11 ;SKIP IF PTP IS CLEAR IN PGCS  
ERROR+ N ;PTP (PGCS BIT1) CANNOT BE CLEARED  
JSR PC,DMPREG
```

(1)

(1)

(5)

(4)

(4)

(3) 003720 000004
(1) 003722 012777 000001 017642
(1) 003730 012777 000002 017640
(1) 003736 012777 100000 017626
(1) 003744 012777 000001 017620
(1) 003752 017737 017620 023610
(1) 003760 032737 000002 023610
(3) 003766 001403
(1) 003770 104014
(1) 003772 004737 022234

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

;BY WRITING IT TO 0


```

677
(4)
(4)
(3) 004212 000004
(1) 004214 012777 100000 017350
(1) 004222 012737 004236 001110
(1) 004230 012737 000017 001124
(1) 004236 012777 000002 017326
(1) 004244 013777 001124 017324
(1) 004252 012777 000002 017312
(1) 004260 017737 017312 023610
(1) 004266 013737 023610 001126
(1) 004274 042737 177760 001126
(1) 004302 023737 001124 001126
(1) 004310 001403
(1) 004312 104017
(1) 004314 004737 022234
(1) 004320 062737 000001 001124
(1) 004326 042737 177760 001124
(1) 004334 023727 001124 000017
(1) 004342 001335
(1) 004344 012737 004352 001110
(1) 004352 012777 000002 017212
(1) 004360 012777 000017 017210
(1) 004366 012777 100000 017176
(1) 004374 012777 000002 017170
(1) 004402 017737 017170 023610
(1) 004410 032737 000017 023610
(1) 004416 001403
(1) 004420 104020
(1) 004422 004737 022234
(1) 004426

```

```

*****
*TEST 13 STTE (BIT3-0) WRITE-READ TEST USING STTE COUNT PATTERN
*****
TST13: SCOPE
MOV #CLR,@ACR ;CLEAR THE WORLD
MOV #1$, $LPERR ;SET SCOPE LOOP POINTER
MOV #17,$GDDAT ;INIT EXPECTED TO 17
1$: MOV #STTEE,@ACR ;SET ADR TO STTE
MOV $GDDAT,@STTE ;LOAD STTE WITH $GDDAT
MOV #STTEE,@ACR ;RESET ADR TO STTE
MOV @STTE,DISPLY ;READ STTE
MOV DISPLY,$BDDAT ;COPY STTE 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 ;STTE 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$, $LPERR ;CHANGE LOOP CONTROL
3$: MOV #STTEE,@ACR ;RESET ADR TO STTE
MOV #17,@STTE ;SET ALL BITS
MOV #CLR,@ACR ;CLEAR THE WORLD
MOV #STTEE,@ACR ;SET ADDR TO STTE
MOV @STTE,DISPLY ;READ STTE
BIT #17,DISPLY ;CHECK BIT3-0 IN STTE FOR 0
BEQ 4$ ;SKIP IF RESET CLEARED BIT3-0 IN STTE
ERROR+ N ;BIT3-0 IN STTE WERE NOT CLEARED BY RESET
JSR PC,DMPREG
4$:

```

679

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(1)

*TEST 14 TEST THAT LKE (STCS BIT2) CAN BE SET

TST14: SCOPE

```
MOV #CLR,@ACR ;CLEAR THE SLATE
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 TST15 ;SKIP IF LKE IS SET IN STCS
ERROR+ N ;LKE (STCS BIT2) CANNOT BE SET
JSR PC,DMPREG
```

*TEST 15 TEST THAT LKE (STCS BIT2) CAN BE CLEARED

TST15: 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 TST16 ;SKIP IF LKE IS CLEAR IN STCS
ERROR+ N ;LKE (STCS BIT2) CANNOT BE CLEARED
JSR PC,DMPREG
```

;BY WRITING IT TO 0

*TEST 16 TEST THAT LKE (STCS BIT2) CAN BE CLEARED BY RESET

TST16: SCOPE

```
MOV #STCSE,@ACR ;SET ADDRESS TO STCS
MOV #LKE,@STCS ;SET LKE IN STCS
MOV #CLR,@ACR ;CLEAR THE INTERFACE
MOV #STCSE,@ACR ;SET ADR TO STCS
MOV @STCS,DISPLY ;READ STCS
BIT #LKE,DISPLY ;CHECK FOR LKE=0
BEQ TST17 ;SKIP IS LKE IS CLEARED BY RESET IN STCS
ERROR+ N ;LKE (STCS BIT2) CANNOT BE CLEARED BY RESET
JSR PC,DMPREG
```

```
681
(4)
(4)
(3) 004646 000004
(1) 004650 012777 100000 016714
(1) 004656 012777 000003 016706
(1) 004664 012777 000002 016704
(1) 004672 012777 000003 016672
(1) 004700 017737 016672 023610
(1) 004706 032737 000002 023610
(3) 004714 001003
(1) 004716 104024
(*) 004720 004737 022234
(1)
(5)
(4)
(4)
(3) 004724 000004
(1) 004726 012777 000003 016636
(1) 004734 012777 000002 016634
(1) 004742 012777 000003 016622
(1) 004750 042777 000002 016620
(1) 004756 012777 000003 016606
(1) 004764 017737 016606 023610
(1) 004772 032737 000002 023610
(3) 005000 001403
(1) 005002 104025
(1) 005004 004737 022234
(1)
(1)
(5)
(4)
(4)
(3) 005010 000004
(1) 005012 012777 000003 016552
(1) 005020 012777 000002 016550
(1) 005026 012777 100000 016536
(1) 005034 012777 000003 016530
(1) 005042 017737 016530 023610
(1) 005050 032737 000002 023610
(3) 005056 001403
(*) 005060 104026
(1) 005062 004737 022234

::*****
:*TEST 17 TEST THAT STP (STCS BIT1) CAN BE SET
::*****
TST17: SCOPE
MOV #CLR,@ACR ;CLEAR THE SLATE
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 TST20 ;SKIP IF STP IS SET IN STCS
ERROR+ N ;STP (STCS BIT1) CANNOT BE SET
JSR PC,DMPREG

::*****
:*TEST 20 TEST THAT STP (STCS BIT1) CAN BE CLEARED
::*****
TST20: 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 TST21 ;SKIP IF STP IS CLEAR IN STCS
ERROR+ N ;STP (STCS BIT1) CANNOT BE CLEARED
JSR PC,DMPREG
;BY WRITING IT TO 0

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


683

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(5)

(4)

(4)

(3)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(3)

(1)

(1)

(1)

*TEST 22 TEST THAT ENB (STCS BIT0) CAN BE SET

TST22: SCOPE

MOV #CLR,@ACR ;CLEAR THE SLATE
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 TST23 ;SKIP IF ENB IS SET IN STCS
ERROR+ N ;ENB (STCS BIT0) CANNOT BE SET
JSR PC,DMPREG

*TEST 23 TEST THAT ENB (STCS BIT0) CAN BE CLEARED

TST23: 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 ;READ STCS
BIT #ENB,DISPLY ;CHECK FOR ENB=0
BEQ TST24 ;SKIP IF ENB IS CLEAR IN STCS
ERROR+ N ;ENB (STCS BIT0) CANNOT BE CLEARED
JSR PC,DMPREG

;BY WRITING IT TO 0

*TEST 24 TEST THAT ENB (STCS BIT0) CAN BE CLEARED BY RESET

TST24: SCOPE

MOV #STCSE,@ACR ;SET ADDRESS TO STCS
MOV #ENB,@STCS ;SET ENB IN STCS
MOV #CLR,@ACR ;CLEAR THE INTERFACE
MOV #STCSE,@ACR ;SET ADR TO STCS
MOV @STCS,DISPLY ;READ STCS
BIT #ENB,DISPLY ;CHECK FOR ENB 0
BEQ TST25 ;SKIP IS ENB IS CLEARED BY RESET IN STCS
ERROR+ N ;ENB (STCS BIT0) CANNOT BE CLEARED BY RESET
JSR PC,DMPREG

```

685
(4)
(4)
(3) 005306 000004
(1) 005310 012777 100000 016254
(1) 005316 012737 005332 001110
(1) 005324 012737 177400 001124
(1) 005332 012777 000003 016232
(1) 005340 013777 001124 016230
(1) 005346 012777 000003 016216
(1) 005354 017737 016216 023610
(1) 005362 013737 023610 001126
(1) 005370 042737 000377 001126
(1) 005376 023737 001124 001126
(1) 005404 001403
(1) 005406 104032
(1) 005410 004737 022234
(1) 005414 062737 000400 001124
(1) 005422 042737 000377 001124
(1) 005430 023727 001124 177400
(1) 005436 001335
(1) 005440 012737 005446 001110
(1) 005446 012777 000003 016116
(1) 005454 012777 177400 016114
(1) 005462 012777 100000 016102
(1) 005470 012777 000003 016074
(1) 005476 017737 016074 023610
(1) 005504 032737 177400 023610
(1) 005512 001403
(1) 005514 104033
(1) 005516 004737 022234
(1) 005522

```

```

:*****
:*TEST 25 STCS (BIT15-8) WRITE-READ TEST USING STCS COUNT PATTERN
:*****
T25: SCOPE
      MOV #CLR,@ACR ;CLEAR THE WORLD
      MOV #1$,SLPERR ;SET SCOPE LOOP POINTER
      MOV #177400,$GDDAT ;INIT EXPECTED TO 177400
1$:   MOV #STCSE,@ACR ;SET ADR TO STCS
      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
      MOV #CLR,@ACR ;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$:

```


691
(4)
(4)
(3) 006152 000004
(1) 006154 012777 100000 015410
(1) 006162 012737 006176 001110
(1) 006170 012737 007400 001124
(1) 006176 012777 000015 015366
(1) 006204 013777 001124 015364
(1) 006212 012777 000015 015352
(1) 006220 017737 015352 023610
(1) 006226 013737 023610 001126
(1) 006234 042737 170377 001126
(1) 006242 023737 001124 001'26
(1) 006250 001403
(1) 006252 104040
(1) 006254 004737 022234
(1) 006260 062737 000400 001124
(1) 006266 042737 170377 001124
(1) 006274 023727 001124 007400
(1) 006302 001335
(1) 006304 012737 006312 001110
(1) 006312 012777 000015 015252
(1) 006320 012777 007400 015250
(1) 006326 012777 100000 015236
(1) 006334 012777 000015 015230
(1) 006342 017737 015230 023610
(1) 006350 032737 007400 023610
(1) 006356 001403
(1) 006360 104041
(1) 006362 004737 022234
(1) 006366

```
::*****  
:*TEST 30 MTC (BIT11-8) WRITE-READ TEST USING MTC COUNT PATTERN  
:*****  
TST30: SCOPE  
MOV #CLR,@ACR ;CLEAR THE WORLD  
MOV #1$,SLPERR ;SET SCOPE LOOP POINTER  
MOV #7400,$GDDAT ;INIT EXPECTED TO 7400  
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 $BDDAT  
BIC #-<7400+1>,$BDDAT ;CLEAR ALL BUT BITS UNDER TEST  
CMP $GDDAT,$BDDAT ;CHECK BITS LOADED AGAINST BITS READ  
BEQ 2$ ;SKIP IF OK  
ERROR+ N ;MTC 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 #MTC,@ACR ;RESET ADR TO MTC  
MOV #7400,@MTC ;SET ALL BITS  
MOV #CLR,@ACR ;CLEAR THE WORLD  
MOV #MTC,@ACR ;SET ADDR TO MTC  
MOV @MTC,DISPLY ;READ MTC  
BIT #7400,DISPLY ;CHECK BIT11-8 IN MTC FOR 0  
BEQ 4$ ;SKIP IF RESET CLEARED BIT11-8 IN MTC  
ERROR+ N ;BIT11-8 IN MTC WERE NOT CLEARED BY RESET  
JSR PC,DMPREG  
4$:
```


750 007050 110437 001176
751 007054 005000

MOV8 R4,\$TMP0 ;\$TMP0=REGISTER BEING WRITTEN
CLR R0 ;START AT 0


```

753 007056 113777 001176 014506 7$:  MOVB  $TMP0,@ACR      ;SELECT REGISTER
754 007064 010001                MOV   R0,R1          ;COPY R0
755 007066 120427 000001        CMPB  R4,#PGCSE      ;PGCS?
756 007072 001411                BEQ   10$            ;SKIP IF YES
757 007074 120427 000003        CMPB  R4,#STCSE      ;STCS?
758 007100 001406                BEQ   10$            ;SKIP IF YES
759 007102 120427 000015        CMPB  R4,#MTCE      ;MTC?
760 007106 001005                BNE   20$            ;SKIP IF NO
761 007110 042701 000007        BIC   #7,R1          ;CLEAR MAINT BITS
762 007114 000402                BR    20$            ;SKIP
763 007116 042701 000001        10$: BIC   #1,R1          ;DO NOT SET GO
764 007122 010177 014450        20$: MOV   R1,@ADR      ;LOAD DATA INTO REGISTER
765 007126 005200                INC   R0              ;BUMP DATA
766 007130 001352                BNE   7$              ;LOAD 0-7777 INTO REGISTER
767 007132 012702 000000        MOV   #0,R2          ;SET POINTER FOR REGISTER ADDRESS TABLE
768 007136 012701 000020        MOV   #16,R1         ;SET COUNTER FOR 16 REGISTERS
769 007142 012703 023616        MOV   #SNAPO,R3      ;SET POINTER FOR INITIAL CONTENTS TABLE
770 007146 110277 014420        8$:  MOVB  R2,@ACR      ;SELECT A REGISTER
771 007152 012337 001124        MOV   (3)+,$GDDAT    ;FETCH INITIAL CONTENTS OF REGISTER
772 007156 127737 014410 001176  CMPB  @ACR,$TMP0      ;IS SELECTED REGISTER THE ONE WRITTEN?
773 007164 001415                BEQ   9$              ;SKIP TESTING IF YES
774 007166 017737 014404 001126  MOV   @ADR,$BDDAT    ;FETCH REGISTER DATA
775 007174 117737 014372 001200  MOVB  @ACR,$TMP1      ;SAVE ADDRESS OF REGISTER CHECKED IN $TMP1
776 007202 023737 001124 001126  CMP   $GDDAT,$BDDAT  ;CHECK REGISTER FOR ANY CHANGES
777 007210 001403                BEQ   9$              ;SKIP IF NO CHANGE
778 007212 104044                ERROR+ 44             ;ADDRESSING REGISTER IN $TMP0 ALSO
779 007214 004737 022234        JSR   PC,DMPREG      ;ADDRESS REGISTER IN $TMP1
780
781 007220 005202                9$:  INC   R2              ;BUMP ADR CODE
782 007222 005301                DEC   R1              ;CHECKED ALL REGISTERS?
783 007224 001350                BNE   8$              ;CHECK ALL REGISTERS
784 007226 005204                INC   R4              ;BUMP POINTER FOR REGISTER ADDRESS
785 007230 005305                DEC   R5              ;DONE ALL?
786 007232 001266                BNE   5$              ;DO ALL
787
99$:

```

```
789 .SBTTL
790 .SBTTL
791 .SBTTL INDIVIDUAL LINE TESTS
792 .SBTTL
793
794 007234 012704 000000 LINE0: MOV #0,R4 ;SET INDEX TO 0
795 007240 004737 007324 JSR PC,LINTST ;TEST LINE
796 007244 012704 000001 MOV #1,R4 ;SET INDEX TO 1
797 007250 004737 007324 JSR PC,LINTST ;TEST LINE
798 007254 012704 000002 MOV #2,R4 ;SET INDEX TO 2
799 007260 004737 007324 JSR PC,LINTST ;TEST LINE
800 007264 012704 000003 MOV #3,R4 ;SET INDEX TO 3
801 007270 004737 007324 JSR PC,LINTST ;TEST LINE
802 007274 013704 023570 LINE1: MOV SIDNUM,R4 ;SET INDEX TO ID #
803 007300 006304 ASL R4 ;SHIFT
804 007302 004737 016072 JSR PC,PGONL ;TEST PGF ONLINE
805 007306 013704 023570 LINE2: MOV SIDNUM,R4 ;SET INDEX TO ID #
806 007312 006304 ASL R4 ;SHIFT
807 007314 004737 017674 JSR PC,STONL ;TEST STF ONLINE
808 007320 000137 020546 JMP ENDTST ;SKIP OVER SUBROUTINE
```

```
810 007324 010437 023614 LINTST: MOV R4,LINMBR ;SAVE LIN UNDER TEST
811 007330 006304 ASL R4 ;MAKE INDEX INTO A WORD POINTER
812
813 ;ISSUE A PROGRAM INTERRUPT REQUEST AND CHECK FOR:
814 ;RDY TO RESET
815 ;RDY TO SET
816 ;PIF BIT TO SET
817 ;DCF REGISTER TO STAY RESET
818 ;EXC REGISTER TO STAY RESET
819
820 ;*****
(3) ;*TEST 33 CHECK A LEGAL PGM PI REQUEST FOR PROPER OPERATION
(3) ;*****
(2) 007332 000004 TST33: SCOPE
821 007334 005037 177776 CLR PSW ;LOWER PRIORITY TO ALLOW INTERUPTS
822 007340 004737 022042 JSR PC,SETMNT ;ENABLE MAINT LOOPBACK
823 007344 012777 000000 014220 MOV #PGTEE,@ACR ;SELECT PGTE
824 007352 016477 020526 014216 MOV PGITAB(4),@PGTE ;LOAD PIE
825 007360 012777 000001 014210 MOV #GO,@PGCS ;TRANSMIT
826 007366 012777 000001 014176 MOV #PGCSE,@ACR ;SELECT PGCS
827 007374 017737 014176 001126 MOV @PGCS,$BDDAT ;READ PGCS
828 007402 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
829 007410 001406 BEQ 1$ ;=0
830 007412 016437 020516 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED PGCS
831 007420 104044 ERROR+ 44 ;SETTING PGCS GO DID NOT CLEAR RDY
832 007422 004737 022234 JSR PC,DMPREG
833 007426 005000 1$: CLR R0 ;INITIALIZE TIMER
834 007430 012777 000001 014134 2$: MOV #PGCSE,@ACR ;SELECT PGCS
835 007436 017737 014134 001126 MOV @PGCS,$BDDAT ;READ PGCS
836 007444 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
837 007452 001013 BNE 3$ ;SKIP IF SET
838 007454 005300 DEC R0 ;TIMER =0?
839 007456 001364 BNE 2$ ;TRY AGAIN IF NO
840 007460 016437 020516 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
841 007466 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY
842 007474 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM PI
843 007476 004737 022234 JSR PC,DMPREG
844 007502 012777 000005 014062 3$: MOV #PGFE,@ACR ;SELECT PGF
845 007510 017737 014062 001126 MOV @PGF,$BDDAT ;READ PGF
846 007516 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAG BITS
847 007524 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
848 007532 001403 BEQ 4$ ;SKIP IF OK
849 007534 104044 ERROR+ 44 ;PGM PI DID NOT SET CORRECT FLAG
850 007536 004737 022234 JSR PC,DMPREG
851 007542 012777 000007 014022 4$: MOV #DCFE,@ACR ;SELECT DCF
852 007550 017737 014022 023610 MOV @DCF,DISPLY ;READ DCF
853 007556 001403 BEQ 5$ ;SKIP IF 0
854 007560 104044 ERROR+ 44 ;PGM PI SET A DCF CONDITION
855 007562 004737 022234 JSR PC,DMPREG
856 007566 012777 000010 013776 5$: MOV #EXCE,@ACR ;SELECT EXC
857 007574 017737 013776 023610 MOV @EXC,DISPLY ;READ EXC
858 007602 001403 BEQ 6$ ;SKIP IF 0
859 007604 104044 ERROR+ 44 ;PGM PI SET AN EXC CONDITION
860 007606 004737 022234 JSR PC,DMPREG
861 007612 012777 000006 013752 6$: MOV #STFE,@ACR ;SELECT STF
862 007620 017737 013752 023610 MOV @STF,DISPLY ;READ STF
```

863 007626 001403
864 007630 04044
865 007632 004737 022234

BEQ 7\$
ERROR+ 44
JSR PC,DMPREG

:SKIP IF CLEAR
:PGM PI SET AN STF CONDITION

867	007636	012700	000017		7\$:	MOV	#17,R0	:SET ALL 4 LINES
868	007642	046400	020526			BIC	PGITAB(4),R0	:CLEAR LINE UNDER TEST
869	007646	012777	000005	013716		MOV	#PGFE,@ACR	:SELECT PGF
870	007654	010077	013716			MOV	R0,@PGF	:CLEAR ALL PIFS EXCEPT LINE UNDER TEST
871	007660	012777	000005	013704		MOV	#PGFE,@ACR	:SELECT PGF
872	007666	017737	013704	001202		MOV	@PGF,\$TMP2	:READ PGF
873	007674	012777	000001	013670		MOV	#PGCSE,@ACR	:SELECT PGCS
874	007702	017737	013670	023610		MOV	@PGCS,DISPLY	:READ PGCS
875	007710	032737	000010	023610		BIT	#IP,DISPLY	:CHECK FOR IP
876	007716	001003				BNE	8\$:SKIP IF SET
877	007720	104044				ERROR+	44	:IP DID NOT SET ON PGF
878	007722	004737	022234			JSR	PC,DMPREG	
879	007726	017746	013652		8\$:	MOV	@IIIP,-(SP)	:SAVE OLD
880	007732	017746	013644			MOV	@IIIV,-(SP)	:PI INFORMATION
881	007736	012777	007770	013636		MOV	#9\$,@IIIV	:LOAD NEW PI VECTOR
882	007744	012777	000340	013632		MOV	#340,@IIIP	:LOAD NEW INT. LEVEL
883	007752	005037	177776			CLR	PS	:ALLOW PI
884	007756	000240				NOP		:STALL
885	007760	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
886	007766	000404				BR	10\$:SKIP OVER ERROR
887	007770	022626			9\$:	POPPOP		:FIX STACK
888	007772	104044				ERROR+	44	:PGF PI WITH IE=0
889	007774	004737	022234			JSR	PC,DMPREG	
890	010000	012777	010074	013574	10\$:	MOV	#12\$,@IIIV	:LOAD NEW PI VECTOR
891	010006	012777	000001	013556		MOV	#PGCSE,@ACR	:SELECT PGCS
892	010014	012777	000004	013554		MOV	#IE,@PGCS	:SET IE
893	010022	012737	000340	001212		MOV	#340,\$TMP6	:INIT PI LEVEL
894	010030	162737	000040	001212	11\$:	SUB	#40,\$TMP6	:DROP PI LEVEL BY 1
895	010036	013737	001212	177776		MOV	\$TMP6,PS	:LOWER PSW
896	010044	000240				NOP		:STALL
897	010046	023727	001212	000140		CMP	\$TMP6,#140	:TRIED ALL LEVELS?
898	010054	001365				BNE	11\$:BR IF NO
899	010056	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
900	010064	104044				ERROR+	44	:NO PGF PI WITH IE=1
901	010066	004737	022234			JSR	PC,DMPREG	
902	010072	000422				BR	13\$:SKIP NFXT TEST
903	010074	022626			12\$:	POPPOP		:FIX STACK
904	010076	006337	001212			ASL	\$TMP6	:RIGHT JUSTIFY
905	010102	006337	001212			ASL	\$TMP6	
906	010106	006337	001212			ASL	\$TMP6	
907	010112	000337	001212			SWAB	\$TMP6	:THE PI LEVEL
908	010116	005237	001212			INC	\$TMP6	:ADD 1 TO MAKE CORECT LEVEL
909	010122	023737	001212	023606		CMP	\$TMP6,IIIL	:CHECK AGAINST KNOWN LEVEL
910	010130	001403				BEQ	13\$:SKIP IF OK
911	010132	104044				ERROR+	44	:PGF PI TO THE WRONG LEVEL
912	010134	004737	022234			JSR	PC,DMPREG	
913	010140	012677	013436		13\$:	MOV	(SP)+,@IIIV	:RESTORE
914	010144	012677	013434			MOV	(SP)+,@IIIP	:OLD PI INFO
915	010150	012777	000005	013414	14\$:	MOV	#PGFE,@ACR	:SELECT PGF
916	010156	016437	020526	001124		MOV	PIFTAB(4),\$GDDAT	:FETCH FLAG BIT
917	010164	013777	001124	013404		MOV	\$GDDAT,@PGF	:CLEAR FLAG
918	010172	005037	001124			CLR	\$GDDAT	:CLEAR EXPECTED
919	010176	012777	000005	013366		MOV	#PGFE,@ACR	:SELECT PGF
920	010204	017737	013366	001126		MOV	@PGF,\$BDDAT	:READ PGF
921	010212	001403				BEQ	15\$:SKIP IF OK
922	010214	104044				ERROR+	44	:WRITING A 1 TO PGF FLAG DID NOT

MAINDEC-11-CRIIA MACY11 30A(1052) 30-NOV-78 11:06 PAGE 6 4
CRIIAA.P11 30-NOV-78 10:58 T33 CHECK A LEGAL PGM PI REQUEST FOR PROPER OPERATION SEQ 0045
923 010216 004737 022234 JSR PC,DMPREG ;CLEAR IT
924
925 010220 58:

```

927
928
929
930
931
932
933
(3)
(3)
(2) 010222 000004
934 010224 004737 022042
935 010230 012777 000000 013334
936 010236 016477 020536 013332
937 010244 012777 000001 013324
938 010252 005000
939 010254 012777 000001 013310
940 010262 017737 013310 001126
941 010270 032737 004000 001126
942 010276 001013
943 010300 005300
944 010302 001364
945 010304 016437 020516 001124
946 010312 062737 004000 001124
947 010320 104044
948 010322 004737 022234
949 010326 012777 000005 013236
950 010334 017737 013236 001126
951 010342 012737 007400 001124
952 010350 023737 001124 001126
953 010356 001403
954 010360 104044
955 010362 004737 022234
956 010366 012777 000007 013176
957 010374 017737 013176 023610
958 010402 001403
959 010404 104044
960 010406 004737 022234
961 010412 012777 000010 013152
962 010420 017737 013152 023610
963 010426 001403
964 010430 104044
965 010432 004737 022234
966 010436 012777 000006 013126
967 010444 017737 013126 023610
968 010452 001403
969 010454 104044
970 010456 004737 022234
971 010462 012777 000005 013102
972 010470 016437 020536 001124
973 010476 013777 001124 013072
974 010504 005137 001124
975 010510 042737 170377 001124
976 010516 012777 000005 013046
977 010524 017737 013046 001126
978 010532 023737 001124 001126
979 010540 001403

```

```

:ISSUE A PROGRAM BOOT REQUEST AND CHECK FOR:
:RDY TO SET
:PGF BIT TO SET
:DCF REGISTER TO STAY RESET
:EXC REGISTER TO STAY RESET

:*****
:*TEST 34 CHECK A LEGAL PGM BOOT REQUEST FOR PROPER OPERATION
:*****
TST34: SCOPE
JSR PC,SETMNT ;ENABLE MAINT LOOPBACK
MOV #PGTEE,@ACR ;SELECT PGTE
MOV PGFTAB(4),@PGTE ;LOAD PGF
MOV #GO,@PGCS ;TRANSMIT
1$: CLR RO ;INITIALIZE TIMER
2$: MOV #PGCSE,@ACR ;SELECT PGCS
MOV @PGCS,$BDDAT ;READ PGCS
BIT #PRDY,$BDDAT ;CHECK FOR RDY
BNE 3$ ;SKIP IF SET
DEC RO ;TIMER -0?
BNE 2$ ;TRY AGAIN IF NO
MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
ADD #PRDY,$GDDAT ;SID + RDY
ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM BOOT
JSR PC,DMPREG
3$: MOV #PGFE,@ACR ;SELECT PGF
MOV @PGF,$BDDAT ;READ PGF
MOV #7400,$GDDAT ;FETCH EXPECTED FLAG BITS
CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
BEQ 4$ ;SKIP IF OK
ERROR+ 44 ;PGM BOOT DID NOT SET CORRECT FLAG
JSR PC,DMPREG
4$: MOV #DCFE,@ACR ;SELECT DCF
MOV @DCF,DISPLY ;READ DCF
BEQ 5$ ;SKIP IF 0
ERROR+ 44 ;PGM BOOT SET A DCF CONDITION
JSR PC,DMPREG
5$: MOV #EXCE,@ACR ;SELECT EXC
MOV @EXC,DISPLY ;READ EXC
BEQ 6$ ;SKIP IF 0
ERROR+ 44 ;PGM BOOT SET AN EXC CONDITION
JSR PC,DMPREG
6$: MOV #STFE,@ACR ;SELECT STF
MOV @STF,DISPLY ;READ STF
BEQ 7$ ;SKIP IF 0
ERROR+ 44 ;PGM BOOT SET AN STF CONDITION
JSR PC,DMPREG
7$: MOV #PGFE,@ACR ;SELECT PGF
MOV PGFTAB(4),$GDDAT ;FETCH FLAG BIT
MOV $GDDAT,@PGF ;CLEAR FLAG
COM $GDDAT ;FLIP GDDAT
BIC #170377,$GDDAT ;MASK OUT JUNK
MOV #PGFE,@ACR ;SELECT PGF
MOV @PGF,$BDDAT ;READ PGF
CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
BEQ 8$ ;SKIP IF OK

```

980 010542 104044 ERROR+ 44 ;WRITING A 1 TO PGF FLAG DID NOT
981 010544 004737 022234 JSR PC,DMPREG ;CLEAR IT
982
983 01055. 8\$:


```

985 ;ISSUE A SANITY TIMER INTERRUPT REQUEST AND CHECK FOR:
986 ;RDY TO RESET
987 ;RDY TO SET
988 ;SIF BIT TO SET
989 ;DCF REGISTER TO STAY RESET
990 ;EXC REGISTER TO STAY RESET
991 ;PGF REGISTER TO STAY RESET
992 ;SIF BIT TO BE CLEARED BY WRITING A 1 TO IT
993
994
(3) ;*****
(3) ;*TEST 35 CHECK A LEGAL ST INTR REQUEST FOR PROPER OPERATION
(2) ;*****
995 010550 000004 TST35: SCOPE
996 010552 004737 022204 JSR PC,SSTMNT ;ENABLE MAINT LOOPBACK
997 010556 012777 000000 013006 MOV #PGTEE,@ACR ;SELECT PGTE
998 010564 016477 020526 013004 MOV PGITAB(4),@PGTE ;LOAD PIE
999 010572 012777 000003 012776 MOV #PTP+GO,@PGCS ;TRANSMIT
1000 010600 012777 000001 012764 MOV #PGCSE,@ACR ;SELECT PGCS
1001 010606 017737 012764 001126 MOV @PGCS,$BDDAT ;READ PGCS
1002 010622 001406 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1003 010624 016437 020516 001124 BEQ 1$ ;=0
1004 010632 104044 ERROR+ 44 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED PGCS
1005 010634 004737 022234 JSR PC,DMPREG ;SETTING PGCS GO DID NOT CLEAR RDY
1006 010640 005000 1$: CLR R0 ;INITIALIZE TIMER
1007 010642 012777 000001 012722 2$: MOV #PGCSE,@ACR ;SELECT PGCS
1008 010650 017737 012722 001126 MOV @PGCS,$BDDAT ;READ PGCS
1009 010656 032737 004000 001126 BIT #PRDY,$BDDAT ;CHECK FOR RDY
1010 010664 001013 BNE 3$ ;SKIP IF SET
1011 010666 005300 DEC R0 ;TIMER =0?
1012 010670 001364 BNE 2$ ;TRY AGAIN IF NO
1013 010672 016437 020516 001124 MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
1014 010700 062737 004000 001124 ADD #PRDY,$GDDAT ;SID + RDY
1015 010706 104044 ERROR+ 44 ;PGCS RDY DID NOT SET ON A PGM PI
1016 010710 004737 022234 JSR PC,DMPREG
1017 010714 012777 000006 012650 3$: MOV #STFE,@ACR ;SELECT STF
1018 010722 017737 012650 001126 MOV @STF,$BDDAT ;READ STF
1019 010730 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAG BITS
1020 010736 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
1021 010744 001403 BEQ 4$ ;SKIP IF OK
1022 010746 104044 ERROR+ 44 ;ST INTR DID NOT SET CORRECT FLAG
1023 010750 004737 022234 JSR PC,DMPREG
1024 010754 012777 000007 012610 4$: MOV #DCFE,@ACR ;SELECT DCF
1025 010762 017737 012610 023610 MOV @DCF,DISPLY ;READ DCF
1026 010770 001403 BEQ 5$ ;SKIP IF 0
1027 010772 104044 ERROR+ 44 ;ST INTR SET A DCF CONDITION
1028 010774 004737 022234 JSR PC,DMPREG
1029 011000 012777 000010 012564 5$: MOV #EXCE,@ACR ;SELECT EXC
1030 011006 017737 012564 023610 MOV @EXC,DISPLY ;READ EXC
1031 011014 001403 BEQ 6$ ;SKIP IF 0
1032 011016 104044 ERROR+ 44 ;ST INTR SET AN EXC CONDITION
1033 011020 004737 022234 JSR PC,DMPREG
1034 011024 012777 000005 012540 6$: MOV #PGFE,@ACR ;SELECT PGF
1035 011032 017737 012540 023610 MOV @PGF,DISPLY ;READ PGF
1036 011040 001403 BEQ 7$ ;SKIP IF CLEAR
1037 011042 104044 ERROR+ 44 ;ST INTR SET A PGF CONDITION

```

MAINDEC-11-CRIIA MACY11 30A(1052) 30-NOV-78 11:06 PAGE 35-1
CRIIAA.P11 30-NOV-78 10:58 T35 CHECK A LEGAL ST INTR REQUEST FOR PROPER OPERATION

SEQ 0049

038 011044 004737 022234

JSR PC,DMPREC

1040	011050	012700	000017		7\$:	MOV	#17,R0	:SET ALL 4 LINES
1041	011054	046400	020526			BIC	STIAB(4),R0	:CLEAR LINE UNDER TEST
1042	011060	012777	000006	012504		MOV	#STFE,@ACR	:SELECT STF
1043	011066	010077	012504			MOV	R0,@STF	:CLEAR ALL SIFS EXCEPT LINE UNDER TEST
1044	011072	012777	000006	012472		MOV	#STFE,@ACR	:SELECT STF
1045	011100	017737	012472	001202		MOV	@STF,\$TMP2	:READ STF
1046	011106	012777	000001	012456		MOV	#PGCSE,@ACR	:SELECT PGCS
1047	011114	017737	012456	023610		MOV	@PGCS,DISPLY	:READ PGCS
1048	011122	032737	000010	023610		BIT	#IP,DISPLY	:CHECK FOR IP
1049	011130	001003				BNE	8\$:SKIP IF SET
1050	011132	104044				ERROR+	44	:IP DID NOT SET ON STF
1051	011134	004737	022234			JSR	PC,DMPREG	
1052	011140	017746	012440		8\$:	MOV	@IIIP,-(SP)	:SAVE OLD
1053	011144	017746	012432			MOV	@IIIV,-(SP)	:PI INFORMATION
1054	011150	012777	011202	012424		MOV	#9\$,@IIIV	:LOAD NEW PI VECTOR
1055	011156	012777	000340	012420		MOV	#340,@IIIP	:LOAD NEW INT. LEVEL
1056	011164	005037	177776			CLR	PS	:ALLOW PI
1057	011170	000240				NOP		:STALL
1058	011172	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
1059	011200	000404				BR	10\$:SKIP OVER ERROR
1060	011202	022626			9\$:	POPPOP		:FIX STACK
1061	011204	104044				ERROR+	44	:STF PI WITH IE=0
1062	011206	004737	022234			JSR	PC,DMPREG	
1063	011212	012777	011260	012362	10\$:	MOV	#11\$,@IIIV	:LOAD NEW PI VECTOR
1064	011220	012777	000001	012344		MOV	#PGCSE,@ACR	:SELECT PGCS
1065	011226	012777	000004	012342		MOV	#IE,@PGCS	:SET IE
1066	011234	005037	177776			CLR	PS	:ALLOW PI
1067	011240	000240				NOP		:STALL
1068	011242	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
1069	011250	104044				ERROR+	44	:NO STF PI WITH IE=1
1070	011252	004737	022234			JSR	PC,DMPREG	
1071	011256	000401				BR	12\$:SKIP NEXT TEST
1072	011260	022626			11\$:	POPPOP		:FIX STACK
1073	011262	012677	012314		12\$:	MOV	(SP)+,@IIIV	:RESTORE
1074	011266	012677	012312			MOV	(SP)+,@IIIP	:OLD PI INFO
1075	011272	012777	000006	012272	13\$:	MOV	#STFE,@ACR	:SELECT STF
1076	011300	016437	020526	001124		MOV	SIFTAB(4),\$GDDAT	:FETCH FLAG BIT
1077	011306	013777	001124	012262		MOV	\$GDDAT,@STF	:CLEAR FLAG
1078	011314	005037	001124			CLR	\$GDDAT	:CLEAR EXPECTED
1079	011320	012777	000006	012244		MOV	#STFE,@ACR	:SELECT STF
1080	011326	017737	012244	001126		MOV	@STF,\$BDDAT	:READ STF
1081	011334	001403				BEQ	14\$:SKIP IF OK
1082	011336	104044				ERROR+	44	:WRITING A 1 TO STF FLAG DID NOT
1083	011340	004737	022234			JSR	PC,DMPREG	
1084								:CLEAR IT
1085	011344				14\$:			

```
1087 :ISSUE A SANITY TIMER BOOT REQUEST AND CHECK FOR:
1088 :RDY TO SET
1089 :SBF BIT TO SET
1090 :DCF REGISTER TO STAY RESET
1091 :EXC REGISTER TO STAY RESET
1092
1093 :*****
(3) :*TEST 36 CHECK A LEGAL ST BOOT REQUEST FOR PROPER OPERATION
(3) :*****
(2) 011344 000004
1094 011346 004737 022204
1095 011352 012777 000000 012212
1096 011360 016477 020536 012210
1097 011366 012777 000003 012202
1098 011374 005000
1099 011376 012777 000001 012166
1100 011404 017737 012166 001126
1101 011412 032737 004000 001126
1102 011420 001013
1103 011422 005300
1104 011424 001364
1105 011426 016437 020516 001124
1106 011434 062737 004000 001124
1107 011442 104044
1108 011444 004737 022234
1109 011450 012777 000006 012114
1110 011456 017737 012114 001126
1111 011464 012737 007400 001124
1112 011472 023737 001124 001126
1113 011500 001403
1114 011502 104044
1115 011504 004737 022234
1116 011510 012777 000007 012054
1117 011516 017737 012054 023610
1118 011524 001403
1119 011526 104044
1120 011530 004737 022234
1121 011534 012777 000010 012030
1122 011542 017737 012030 023610
1123 011550 001403
1124 011552 104044
1125 011554 004737 022234
1126 011560 012777 000007 012004
1127 011566 017737 012004 023610
1128 011574 001403
1129 011576 104044
1130 011600 004737 022234
1131 011604 012777 000006 011760
1132 011612 016437 020536 001124
1133 011620 013777 001124 011750
1134 011626 005137 001124
1135 011632 042737 170377 001124
1136 011640 012777 000006 011724
1137 011646 017737 011724 001126
1138 011654 023737 001124 001126
1139 011662 001403

TST36: SCOPE
JSR PC,SSTMT ;ENABLE MAINT LOOPBACK
MOV #PGTEE,@ACR ;SELECT PGTE
MOV PGBTAB(4),@PGTE ;LOAD PIB
MOV #PTP+G,@PGCS ;TRANSMIT
1$: CLR R0 ;INITIALIZE TIMER
2$: MOV #PGCSE,@ACR ;SELECT PGCS
MOV @PGCS,$BDDAT ;READ PGCS
BIT #PRDY,$BDDAT ;CHECK FOR RDY
BNE 3$ ;SKIP IF SET
DEC R0 ;TIMER -0?
BNE 2$ ;TRY AGAIN IF NO
MOV SIDTAB(4),$GDDAT ;LOAD EXPECTED WITH
ADD #PRDY,$GDDAT ;SID + RDY
ERROR+ 44 ;PGCS RDY DID NOT SET ON A ST BOOT
JSR PC,DMPREG
MOV #STFE,@ACR ;SELECT STF
MOV @STF,$BDDAT ;READ STF
MOV #7400,$GDDAT ;FETCH EXPECTED FLAG BITS
CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT FLAG
BEQ 4$ ;SKIP IF OK
ERROR+ 44 ;ST BOOT DID NOT SET CORRECT FLAG
JSR PC,DMPREG
MOV #DCFE,@ACR ;SELECT DCF
MOV @DCF,DISPLY ;READ DCF
BEQ 5$ ;SKIP IF 0
ERROR+ 44 ;ST BOOT SET A DCF CONDITION
JSR PC,DMPREG
MOV #EXCE,@ACR ;SELECT EXC
MOV @EXC,DISPLY ;READ EXC
BEQ 6$ ;SKIP IF 0
ERROR+ 44 ;ST BOOT SET AN EXC CONDITION
JSR PC,DMPREG
MOV #PGFE,@ACR ;SELECT PGF
MOV @PGF,DISPLY ;READ PGF
BEQ 7$ ;SKIP IF 0
ERROR+ 44 ;ST BOOT SET A PGF CONDITION
JSR PC,DMPREG
MOV #STFE,@ACR ;SELECT STF
MOV SBTAB(4),$GDDAT ;FETCH FLAG BIT
MOV $GDDAT,@STF ;CLEAR FLAG
COM $GDDAT ;FLIP GDDAT
BIC #170377,$GDDAT ;MASK OUT JUNK
MOV #STFE,@ACR ;SELECT STF
MOV @STF,$BDDAT ;READ STF
CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
BEQ 8$ ;SKIP IF OK
```

1140 011654 104044
1141 011656 004737 022234
1142
1143 01167.

ERROR+ 44 ;WRITING A 1 TO STF FLAG DID NOT
JSR PC,DMPREG ;CLEAR IT
88:

```
1145 ;ISSUE A PROGRAM INTERRUPT REQUEST WITH IMSK BIT =1 AND CHECK:
1146 ;PIF BIT TO STAY 0
1147 ;UI BIT TO SET
1148 ;UI BIT TO CLEAR WHEN WRITTEN TO A 1
1149
1150 ;*****
(3) ;*TEST 37 CHECK THE IMSK BIT FOR PROPER OPERATION
(3) ;*****
(2) 011672 000004
1151 011674 004737 022042 JSR PC,SETMNT ;ENABLE MAINT. LOOP
1152 011700 012777 000000 011664 MOV #PGTEF,@ACR ;SELECT PGTE
1153 011706 016477 020526 011662 MOV PGITAB(4),@PGTE ;LOAD PIB
1154 011714 012777 000004 011650 MOV #IMSKE,@ACR ;SELECT IMSK
1155 011722 016477 020526 011646 MOV IMITAB(4),@IMSK;INHIBIT LINE
1156 011730 012777 000001 011634 MOV #PGCSE,@ACR ;SELECT PGCS
1157 011736 012777 000001 011632 MOV #GO,@PGCS ;TRANSMIT
1158 011744 012777 000001 011620 3: MOV #PGCSE,@ACR ;SELECT PGCS
1159 011752 032777 004000 011616 BIT #PRDY,@PGCS ;CHECK FOR
1160 011760 001771 BEQ 1$ ;READY
1161 011762 012777 000005 011602 MOV #PGFE,@ACR ;SELECT PGF
1162 011770 017737 011602 001126 MOV @PGF,$BDDAT ;READ PGF
1163 011776 012737 000017 001124 MOV #17,$GDDAT ;FETCH EXPECTED FLAGS
1164 012004 046437 020526 001124 BIC PGITAB(4),$GDDAT;MINUS EXPECTED FLAG
1165 012012 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1166 012020 001403 BEQ 2$ ;SKIP IF OK
1167 012022 104044 ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS PGF BIT
1168 012024 004737 022234 JSR PC,DMPREG
1169 012030 016437 020536 001124 2$: MOV UVITAB(4),$GDDAT;LOAD EXPECTED UI BIT
1170 012036 012777 000010 011526 MOV #EXCE,@ACR ;SELECT EXC
1171 012044 017737 011526 001126 MOV @EXC,$BDDAT ;READ EXC
1172 012052 023737 001124 001126 CMP $GDDAT,$BDDAT ;CHECK THE UI BIT
1173 012060 001403 BEQ 3$ ;SKIP IF SET
1174 012062 104044 ERROR+ 44 ;UI DID NOT SET ON A PIF WITH IMSK 1
1175 012064 004737 022234 JSR PC,DMPREG
1176 012070 005037 001124 3$: CLR $GDDAT ;EXPECT 0
1177 012074 012777 000010 011470 MOV #EXCE,@ACR ;SELECT EXC
1178 012102 016477 020536 011466 MOV UVITAB(4),@EXC ;CLEAR UI BIT
1179 012110 012777 000010 011454 MOV #EXCE,@ACR ;SELECT EXC
1180 012116 017737 011454 023610 MOV @EXC,DISPLY ;READ EXC
1181 012124 013737 023610 001126 MOV DISPLY,$BDDAT ;TEST EXC
1182 012132 001403 BEQ 4$ ;SKIP IF 0
1183 012134 104044 ERROR+ 44 ;WRITING A 1 TO UI BIT DID NOT CLEAR IT
1184 012136 004737 022234 JSR PC,DMPREG
1185 012142 012777 000006 011422 4$: MOV #STFE,@ACR ;SELECT STF REGISTER
1186 012150 017737 011422 001126 MOV @STF,$BDDAT ;GET AND TEST STF REGISTER
1187 012156 001403 BEQ 5$ ;STF SHOULD BE 0
1188 012160 104044 ERROR+ 44 ;STF NOT ZERO ON A PGI WITH IMSK-1
1189 012162 004737 022234 JSR PC,DMPREG
1190 012166 5$:
```

```
1192 ;ISSUE A PROGRAM BOOT REQUEST WITH BIMSK BIT -1 AND CHECK:
1193 ;PBF BIT TO STAY 0
1194 ;UI BIT TO SET
1195
1196 ;*****
(3) ;*TEST 40 CHECK THE IMSK BIT FOR PROPER OPERATION ON PGM BOOT
(3) ;*****
(2) TST40: SCOPE
1197 012166 000004 JSR PC,SETMNT ;ENABLE MAINT. LOOP
1198 012170 004737 022042 MOV #PGTEE,@ACR ;SELECT PGTE
1199 012174 012777 000000 011370 MOV PGBTAB(4),@PGTE ;LOAD PGB
1200 012202 016477 020536 011366 MOV #IMSKE,@ACR ;SELECT IMSK
1201 012210 012777 000004 011354 MOV #PGCSE,@ACR ;SELECT PGCS
1202 012216 016477 020536 011352 MOV #IMSK,@ACR ;INHIBIT LINE
1203 012224 012777 000001 011340 MCV #GO,@PGCS ;TRANSMIT
1204 012232 012777 000001 011336 1$: MOV #PGCSE,@ACR ;SELECT PGCS
1205 012240 012777 000001 011324 BIT #PRDY,@PGCS ;CHECK FOR
1206 012246 032777 004000 BEQ 1$ ;READY
1207 012254 001771 MOV #PGFE,@ACR ;SELECT PGF
1208 012256 012777 000005 011306 MOV @PGF,$BDDAT ;READ PGF
1209 012264 017737 011306 001126 MOV #7400,$GDDAT ;FEICH EXPECTED FLAGS
1210 012272 012737 007400 001124 BIC PGBTAB(4),$GDDAT ;MINUS EXPECTED FLAG
1211 012300 046437 020536 001124 CMP $GDDAT,$BDDAT ;CHECK FLAG FOR 0
1212 012306 023737 001124 BEQ 2$ ;SKIP IF OK
1213 012314 001403 ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS PGF BIT
1214 012316 104044 JSR PC,DMPREG
1215 012320 004737 022234 2$: MOV UVITAB(4),$GDDAT ;LOAD EXPECTED UI BIT
1216 012324 016437 020536 001124 MOV #EXCE,@ACR ;SELECT EXC
1217 012332 012777 000010 011232 MOV @EXC,$BDDAT ;READ EXC
1218 012340 017737 011232 001126 CMP $GDDAT,$BDDAT ;CHECK THE UI BIT
1219 012346 023737 001124 001126 BEQ 3$ ;SKIP IF SET
1220 012354 001403 ERROR+ 44 ;UI DID NOT SET ON A PIF WITH IMSK=1
1221 012356 104044 JSR PC,DMPREG
1222 012360 004737 022234 3$: CLR $GDDAT ;EXPECT STF TO BE 0
1223 012364 005037 001124 MOV #STFE,@ACR ;SELECT STF
1224 012370 012777 000006 011174 MOV @STF,$BDDAT ;GET AND TEST STF
1225 012376 017737 011174 001126 BEQ 4$ ;IT SHOULD BE 0
1226 012404 001403 ERROR+ 44
1227 012406 104044 JSR PC,DMPREG
1228 012410 004737 022234 4$:
1228 012414
```

1230
1231
1232
1233
1234
1235
(3)
(3)
(2) 012414 000004
1236 012416 004737 022204
1237 012422 012777 000000 011142
1238 012430 016477 020526 011140
1239 012436 012777 000004 011126
1240 012444 016477 020526 011124
1241 012452 012777 000001 011112
1242 012460 012777 000003 011110
1243 012466 012777 000001 011076
1244 012474 032777 004000 011074
1245 012502 001771
1246 012504 012777 000006 011060
1247 012512 017737 011060 001126
1248 012520 012737 000017 001124
1249 012526 046437 020526 001124
1250 012534 023737 001124 001126
1251 012542 001403
1252 012544 104044
1253 012546 004737 022234
1254 012552 016437 020536 001124 2\$:
1255 012560 012777 000010 011004
1256 012566 017737 011004 001126
1257 012574 023737 001124 001126
1258 012602 001403
1259 012604 104044
1260 012606 004737 022234
1261 012612 005037 001124 3\$:
1262 012616 012777 000010 010746
1263 012624 016477 020536 010744
1264 012632 012777 000010 010732
1265 012640 017737 010732 023610
1266 012646 013737 023610 001126
1267 012654 001403
1268 012656 104044
1269 012660 004737 022234
1270 012664 012777 000005 010700 4\$:
1271 012672 017737 010700 001126
1272 012700 001403
1273 012702 104044
1274 012704 004737 022234
1275 012700

:ISSUE A SANITY TIMER INTERRUPT REQUEST WITH IMSK BIT =1 AND CHECK:
:SIF BIT TO STAY 0
:UI BIT TO SET
:UI BIT TO CLEAR WHEN WRITTEN TO A 1
:*****
:*TEST 41 CHECK THE IMSK BIT FOR PROPER OPERATION ON ST INTR
:*****
TST41: SCOPE
JSR PC,SSTMT ;ENABLE MAINT. LOOP
MOV #PGTEE,@ACR ;SELECT PGTE
MOV PGITAB(4),@PGTE ;LOAD PIB
MOV #IMSKE,@ACR ;SELECT IMSK
MOV IMITAB(4),@IMSK;INHIBIT LINE
MOV #PGCSE,@ACR ;SELECT PGCS
MOV #PTP+GO,@PGCS ;TRANSMIT
\$: MOV #PGCSE,@ACR ;SELECT PGCS
BIT #PRDY,@PGCS ;CHECK FOR
BEQ 1\$;READY
MOV #STFE,@ACR ;SELECT STF
MOV @STF,\$BDDAT ;READ STF
MOV #17,\$GDDAT ;FETCH EXPECTED FLAGS
BIC PGITAB(4),\$GDDAT;MINUS EXPECTED FLAG
CMP \$GDDAT,\$BDDAT ;CHECK FLAG FOR 0
BEQ 2\$;SKIP IF OK
ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS SIF BIT
JSR PC,DMPREG
2\$: MOV UVITAB(4),\$GDDAT;LOAD EXPECTED UI BIT
MOV #EXCE,@ACR ;SELECT EXC
MOV @EXC,\$BDDAT ;READ EXC
CMP \$GDDAT,\$BDDAT ;CHECK THE UI BIT
BEQ 3\$;SKIP IF SET
ERROR+ 44 ;UI DID NOT SET ON A SIF WITH IMSK=1
3\$: JSR PC,DMPREG
CLR \$GDDAT ;EXPECT 0
MOV #EXCE,@ACR ;SELECT EXC
MOV UVITAB(4),@EXC ;CLEAR UI BIT
MOV #EXCE,@ACR ;SELECT EXC
MOV @EXC,DISPLY ;READ EXC
MOV DISPLY,\$BDDAT ;TEST EXC
BEQ 4\$;SKIP IF 0
ERROR+ 44 ;WRITING A 1 TO UI BIT DID NOT CLEAR IT
4\$: JSR PC,DMPREG
MOV #PGFE,@ACR ;SELECT PGF REGIPGER
MOV @PGF,\$BDDAT ;GET AND TEST STF REGISTER
BEQ 5\$;PGF SHOULD BE 0
ERROR+ 44 ;PGF NOT ZERO ON A STI WITH IMSK=1
5\$: JSR PC,DMPREG

1277
1278
1279
1280
1281
(3)
(3)
(2)
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313

012710 000004
012712 004737 022204
012716 012777 000000 010646
012724 016477 020536 010644
012732 012777 000004 010632
012740 016477 020536 010630
012746 012777 000001 010616
012754 012777 000003 010614
012762 012777 000001 010602
012770 032777 004000 010600
012776 001771
013000 012777 000006 010564
013006 017737 010564 001126
013014 012737 007400 001124
013022 046437 020536 001124
013030 023737 001124 001126
013036 001403
013040 104044
013042 004737 022234
013046 016437 020536 001124
013054 012777 000010 010510
013062 017737 010510 001126
013070 023737 001124 001126
013076 001403
013100 104044
013102 004737 022234
013106 005037 001124
013112 012777 000005 010452
013120 017737 010452 001126
013126 001403
013130 104044
013132 004737 022234
013136

;ISSUE A SANITY TIMER BOOT REQUEST WITH BIMSK BIT 1 AND CHECK:
;SBF BIT TO STAY 0
;UI BIT TO SET
;*****
;*TEST 42 CHECK THE IMSK BIT FOR PROPER OPERATION ON ST BOOT
;*****
TST42: SCOPE
JSR PC,SSTMNT ;ENABLE MAINT. LOOP
MOV #PGTEE,@ACR ;SELECT PGTE
MOV PGBTAB(4),@PGTE ;LOAD PGB
MOV #IMSKE,@ACR ;SELECT IMSK
MOV BMITAB(4),@IMSK ;INHIBIT LINE
MOV #PGCSE,@ACR ;SELECT PGCS
MOV #PTP+GO,@PGCS ;TRANSMIT
1\$: MOV #PGCSE,@ACR ;SELECT PGCS
BIT #PRDY,@PGCS ;CHECK FOR
BEQ 1\$;READY
MOV #STFE,@ACR ;SELECT STF
MOV @STF,\$BDDAT ;READ STF
MOV #7400,\$GDDAT ;FETCH EXPECTED FLAGS
BIC PGBTAB(4),\$GDDAT ;MINUS EXPECTED FLAG
CMP \$GDDAT,\$BDDAT ;CHECK FLAG FOR 0
BEQ 2\$;SKIP IF OK
ERROR+ 44 ;IMSK BIT DID NOT INHIBIT ITS SBF BIT
JSR PC,DMPREG
2\$: MOV UVITAB(4),\$GDDAT ;LOAD EXPECTED UI BIT
MOV #EXCE,@ACR ;SELECT EXC
MOV @EXC,\$BDDAT ;READ EXC
CMP \$GDDAT,\$BDDAT ;CHECK THE UI BIT
BEQ 3\$;SKIP IF SET
ERROR+ 44 ;UI DID NOT SET ON A SBF WITH IMSK 1
JSR PC,DMPREG
3\$: CLR \$GDDAT ;EXPECT PGF TO BE 0
MOV #PGFE,@ACR ;SELECT PGF
MOV @PGF,\$BDDAT ;GET AND TEST PGF
BEQ 4\$;IT SHOULD BE 0
ERROR+ 44
4\$: JSR PC,DMPREG

1315
1316
1317
(3)
(3)
(2) 013136 000004
1318 013140 004737 022204
1319 013144 005000
1320 013146 012777 000002 010416
1321 013154 016477 020526 010414
1322 013162 012777 000001 010406
1323 013170 012777 000003 010374
1324 013176 017737 010374 023610
1325 013204 032737 177410 023610
1326 013212 001007
1327 013214 005200
1328 013216 001364
1329 013220 104044
1330 013222 004737 022234
1331 013226 012700 177777
1332 013232 010037 023612
1333 013236 006200
1334 013240 006200
1335 013242 006200
1336 013244 060037 023612
1337 013250 012777 000003 010314
1338 013256 017737 010314 023610
1339 013264 005037 001126
1340 013270 113737 023611 001126
1341 013276 012737 000377 001124
1342 013304 023737 001124 001126
1343 013312 001403
1344 013314 104044
1345 013316 004737 022234
1346 013322 012737 000011 001124
1347 013330 113737 023610 001126
1348 013336 023737 001124 001126
1349 013344 001403
1350 013346 104044
1351 013350 004737 022234
1352 013354 012777 000001 010210
1353 013362 017737 010210 023610
1354 013370 032737 000010 023610
1355 013376 001003
1356 013400 104044
1357 013402 004737 022234
1358 013406

;TEST OPERATION OF THE TIMER LOGIC

; *TEST 43 CHECK THE TIMER INTERRUPT LOGIC AND COUNT REGISTER

TST43: SCOPE

JSR PC,SSTMNT ;SET MAINT. LOOPBACK
CLR RO ;INIT. TIMER COUNTER
MOV #STEE,@ACR ;SELECT STE
MOV SIFTAB(4),@STTE ;ENABLE SI LINE
MOV #ENB,@STCS ;ENABLE SANITY TIMER
1\$: 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 #-1,RO ;SET TIME TO MAX
2\$: MOV RO,TIMPER ;SAVE TIME PERIOD CONSTANT
ASR RO ;DIVIDE BY
ASR RO ;8
ADD RO,TIMPER ;ADD 12%
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
BNE 5\$;SKIP IF SET
ERROR+ 44 ;SIF DID NOT SET IP
3\$: JSR PC,DMPREG
5\$: JSR PC,DMPREG

```

1360          :*****
(3)          :*TEST 44      CHECK THE TIMER BOOT LOGIC AND COUNT REGISTER
(3)          :*****
(2) 013406 000004 TST44: SCOPE
1361 013410 004737 022204      JSR    PC,SSTMNT      ;SET MAINT. LOOPBACK
1362 013414 012777 000002 010150  MOV    #STEE,@ACR     ;SELECT STTE
1363 013422 016477 020536 010146  MOV    SBFTAB(4),@STTE ;ENABLE SB LINE
1364 013430 012777 000000 010140  MOV    #ENB,@STCS     ;ENABLE SANITY TIMER
1365 013436 005000          CLR    R0             ;CLR COUNTER
1366 013440 012777 000003 010124  1$:  MOV    #STCSE,@ACR    ;SELECT STCS
1367 013446 017737 010124 023610  MOV    @STCS,DISPLY   ;READ STCS
1368 013454 032737 177410 023610  BIT    #177410,DISPLY ;CHECK FOR TMO OR COUNT -1
1369 013462 001005          BNE    2$            ;SKIP IF ANY BITS SET
1370 013464 005200          INC    R0             ;COUNT BACKGROUND TIMER
1371 013466 001364          BNE    1$            ;KEEP COUNTING UNTIL OVERFLOW
1372 013470 104044          ERROR+ 44           ;TIMER DID NOT COUNTDOWN WHEN ENABLED
1373 013472 004737 022234      JSR    PC,DMPREG
1374 013476 012777 000003 010066  2$:  MOV    #STCSE,@ACR     ;***
1375 013504 017737 010066 023610  MOV    @STCS,DISPLY   ;***
1376 013512 005037 001126          CLR    $BDDAT        ;INIT. DISPLAY
1377 013516 113737 023611 001126  MOV    DISPLY+1,$BDDAT ;FETCH COUNT
1378 013524 012737 000377 001124  MOV    #377,$GDDAT    ;LOAD EXPECTED
1379 013532 023737 001124 001126  CMP    $GDDAT,$BDDAT  ;CHECK FOR TIMER -1
1380 013540 001403          BEQ    3$            ;SKIP IF OK
1381 013542 104044          ERROR+ 44           ;TIMER DID NOT COUNT FROM 0 TO -1
1382 013544 004737 022234      JSR    PC,DMPREG
1383 013550 012737 000011 001124  3$:  MOV    #TMO+ENB,$GDDAT ;LOAD EXPECTED
1384 013556 113737 023610 001126  MOV    DISPLY,$BDDAT  ;LOAD ACTUAL
1385 013564 023737 001124 001126  CMP    $GDDAT,$BDDAT ;CHECK FOR TMO
1386 013572 001403          BEQ    4$            ;SKIP IF SET
1387 013574 104044          ERROR+ 44           ;TMO DID NOT SET ON TIMER UNDERFLOW
1388 013576 004737 022234      JSR    PC,DMPREG
1389 013602 012777 000001 007762  4$:  MOV    #PGCSE,@ACR    ;SELECT PGCS
1390 013610 017737 007762 023610  MOV    @PGCS,DISPLY   ;READ PGCS
1391 013616 032737 000010 023610  BIT    #IP,DISPLY     ;TEST FOR IP
1392 013624 001403          BEQ    5$            ;SKIP IF CLR
1393 013626 104044          ERROR+ 44           ;SBF SET IP
1394 013630 004737 022234      JSR    PC,DMPREG
1395 013634          $:

```

```

1397
(3)
(3)
(2) 013634 000004
1398 013636 012737 000001 001204
1399 013644 005037 001126
1400 013650 012737 013656 001110
1401 013656 004737 022204
1402 013662 012777 000002 007702
1403 013670 016477 020526 007700
1404 013676 112777 000001 007672
1405 013704 005377 007662
1406 013710 113777 001204 007662
1407 013716 013737 001204 001124
1408 013724 005337 001124
1409 013730 013700 023612
1410 013734 006300
1411 013736 005377 007630
1412 013742 117737 007632 001126
1413 013750 123737 001204 001126
1414 013756 001006
1415 013760 005300
1416 013762 001365
1417 013764 004044
1418 013766 004737 022234
1419 013772 000414
1420 013774 005377 007572
1421 014000 117737 007574 001126
1422 014006 023737 001124 001126
1423 014014 001403
1424 014016 104044
1425 014020 004737 022234
1426 014024 105237 001204
1427 014030 001312

```

```

:*****
:*TEST 45 CHECK THE COUNTER FOR PROPER DECREMENTATION
:*****
TST45: SCOPE
MOV #1,$TMP3 ;INIT THE COUNT VALUE
CLR $BDDAT ;ZERO THE HIGH BYTE OF $BDDAT
MOV #1,$SLPERR ;SET SCOPE POINT
1$: JSR PC,SSTMT ;ENTER MAINT. LOOP MODE
MOV #STEE,@ACR ;SELECT SITE
MOV SIFTAB(4),@STTE ;SET LINE ENABLE
MOVB #ENB,@STCS ;ENABLE COUNTING
DEC @ACR ;SELECT STCS
MOVB $TMP3,@STCSH ;LOAD COUNT AND START
MOV $TMP3,$GDDAT ;GET STARTING COUNT VALUE
DEC $GDDAT ;MINUS 1
MOV TIMPER,R0 ;GET BASIC TIME CONSTANT
ASL R0 ;*2
2$: DEC @ACR ;SELECT STCS
MOVB @STCSH,$BDDAT ;READ COUNT VALUE
CMPB $TMP3,$BDDAT ;CHECK FOR CHANGE
BNE 3$ ;SKIP IF CHANGED
DEC R0 ;DEFLATE COUNTER
BNE 2$ ;CHECK AGAIN
ERROR+ 44 ;TIMER DID NOT DECREMENT
JSR PC,DMPREG
BR 4$ ;SKIP NEXT CHECK
3$: DEC @ACR ;SELECT STCS
MOVB @STCSH,$BDDAT ;REREAD TIMER
CMP $GDDAT,$BDDAT ;CHECK FOR CORRECT DECREMENTATION
BEQ 4$ ;SKIP IF OK
ERROR+ 44 ;COUNT REGISTER DID NOT DECREMENT CORRECTLY
JSR PC,DMPREG
4$: INCB $TMP3 ;BUMP TEST VALUE
BNE 1$ ;DO 1 - 377

```

```

1429
1430
1431
1432
1433
1434
1435
1436
1437          000005
1438
1439
(3)
(3)
(2) 014032 000004
1440 014034 004737 022042
1441 014040 012737 014064 001110
1442 014046 012777 000007 007516
1443 014054 017737 007516 023616
1444 014062 005000
1445
1446 014064 010037 001200
1447 014070 004737 022214
1448 014074 113737 001200 001201
1449 014102 106037 001201
1450 014106 106037 001201
1451 014112 106037 001201
1452 014116 106037 001201
1453 014122 012777 000000 007442
1454 014130 013777 001200 007440
1455 014136 006301
1456 014140 042701 177775
1457 014144 016437 020516 001124
1458 014152 050137 001124
1459 014156 052701 000001
1460 014162 012777 000001 007402
1461 014170 010177 007402
1462 014174 005001
1463 014176 052737 004010 001124
1464 014204 105377 007362
1465 014210 017737 007362 001126
1466 014216 032737 004000 001126
1467 014224 001005
1468 014226 005301
1469 014230 001365
1470 014232 104044
1471 014234 004737 022234
1472 014240 023737 001126 001124
1473 014246 001403
1474 014250 104044
1475 014252 004737 022234

```

```

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

```

W=5

```

*****
;TEST 46 CHECK OPERATION OF PARITY & FRAMING ERRORS, IM(3:0) 0
*****

```

```

1ST46:  SCOPE
        JSR    PC,SETMNT      ;ENABLE MAINTENANCE LOOPBACK
        MOV    #1$, $LPERR   ;FIX START OF LOOP
        MOV    #DCFE,@ACR
        MOV    @DCF,$NAPO    ;SAVE IMAGE OF DCF REGISTER
        CLR    R0            ;INIT FRAME, TYPE, PGTE DATA SETUP

1$:     MOV    R0,$TMP1
        JSR    PC,PARCAL     ;CALCULATE PARITY OF DATA
        MOVB  $TMP1,$TMP1+1 ;POSITION DATA
        RORB  $TMP1+1
        RORB  $TMP1+1
        RORB  $TMP1+1
        RORB  $TMP1+1
        MOV    #PGTEE,@ACR
        MOV    $TMP1,@PGTE   ;LOAD TRANSMIT ENABLES
        ASL    R1            ;GET PARITY

2$:     BIC    #177775,R1
        MOV    SIDTAB(4),$GDDAT ;GET SELF-ID
        BIS    R1,$GDDAT
        BIS    #1,R1
        MOV    #PGCSE,@ACR
        MOV    R1,@PGCS     ;LOAD PARITY, SET GO
        CLR    R1
        BIS    #PRDY+IP,$GDDAT

3$:     DECB  @ACR
        MOV    @PGCS,$BDDAT
        BIT   #PRDY,$BDDAT  ;WAIT FOR READY
        BNE   4$
        DEC   R1            ;BUT DON'T WAIT TOO LONG
        BNE   3$
        ERROR+ 44
        JSR    PC,DMPREG
        CMP   $BDDAT,$GDDAT ;IS PCC OK?
        BEQ   5$
        ERROR+ 44
        JSR    PC,DMPREG

```

```

1477 014256 012737 000017 001124 5$:  MOV #17,$GDDAT ;GET EXPECTED RTE FLAGS
1478 014264 012777 000010 007300  MOV #EXCE,@ACR
1479 014272 017737 007300 001126  MOV @EXC,$BDDAT ;GET EXC REGISTER
1480 014300 023737 001124 001126  CMP $GDDAT,$BDDAT
1481 014306 001403  BEQ 6$
1482 014310 104044  ERROR+ 44
1483 014312 004737 022234  JSR PC,DMPREG
1484 014316 105377 007250 6$:  DECB @ACR
1485 014322 016437 020526 001124  MOV PIFTAB(4),$GDDAT
1486 014330 013777 001124 007240  MOV $GDDAT,@EXC ;CLEAR FLAG
1487 014336 005137 001124  COM $GDDAT ;MAKE FLAG DATA
1488 014342 042737 177760 001124  BIC #177760,$GDDAT
1489 014350 105377 007216  DECB @ACR
1490 014354 017737 007216 001126  MOV @EXC,$BDDAT
1491 014362 023737 001126 001124  CMP $BDDAT,$GDDAT
1492 014370 001403  BEQ 7$
1493 014372 104044  ERROR+ 44
1494 014374 004737 022234  JSR PC,DMPREG
1495 014400 005037 001124 7$:  CLR $GDDAT
1496 014404 012777 000005 007160  MOV #PGFE,@ACR
1497 014412 017737 007160 001126  MOV @PGF,$BDDAT ;PGF SHOULD BE 0
1498 014420 001403  BEQ 8$
1499 014422 104044  ERROR+ 44
1500 014424 004737 022234  JSR PC,DMPREG
1501 014430 017737 007142 001126 8$:  MOV @STF,$BDDAT ;GET STF
1502 014436 001403  BEQ 9$ ;STF SHOULD BE 0
1503 014440 104044  ERROR+ 44
1504 014442 004737 022234  JSR PC,DMPREG
1505 014446 013737 023616 001124 9$:  MOV SNAPO,$GDDAT ;GET EXPECTED DATA FOR DCF
1506 014454 017737 007116 001126  MOV @DCF,$BDDAT ;GET DCF REGISTER
1507 014462 023737 001126 001124  CMP $BDDAT,$GDDAT ;CHECK THAT THERE IS NO CHANGE
1508 014470 001403  BEQ 10$
1509 014472 104044  ERROR+ 44
1510 014474 004737 022234  JSR PC,DMPREG
1511 014500 005200 10$:  INC R0 ;INCREMENT TEST DATA
1512 014502 012777 000015 007062  MOV #MTC,@ACR ;POINT TO MTC
1513 014510 042777 006000 007060  BIC #MTC+MFRM,@MTC ;CLEAR TYPE AND FRAME ERROR
1514 014516 032700 000400  BIT #400,R0
1515 014522 001403  BEQ 11$ ;IF R0(8)=1, SET MTYPE
1516 014524 052777 004000 007044  BIS #MTC,@MTC
1517 014532 032700 001000 11$:  BIT #1000,R0
1518 014536 001403  BEQ 12$
1519 014540 052777 002000 007030  BIS #MFRM,@MTC ;IF R0(9)=1, SET FRAMING ERROR
1520 014546 020027 002000 12$:  CMP R0,#2000
1521 014552 103002  BHIS 13$ ;LOOP UNTIL ALL COMBOS OF DATA, TYPE &
1522 014554 000137 014064 ;FRAMING HAVE BEEN TESTED.
1523 014560 13$:  JMP 1$
1524 014560 13$:

```

1526 :ISSUE A PROGRAM-GENERATED REQUEST WITH INTERRUPT MASKS SET WITH ALL COMBINATIONS OF
1527 :TRANSMIT ENABLES, MAINTENANCE TYPE, AND MAINTENANCE FRAMING
1528 :AND CHECK FOR:
1529 : CORRECT CONTENTS OF PGCS (RDY, SELF-ID)
1530 : RTE BITS IN THE EXC REGISTER TO BE CLEAR
1531 : ALL OTHER FLAGS TO BE CLEAR (PGF, STF, DCF, UIF)
1532 :
1533 :*****

(3) :*TEST 47 CHECK OPERATION OF PARITY & FRAMING ERRORS, IM(3:0) 1
(3) :*****

(2) 014560 000004
1534 014562 004737 022042 1\$: JSR PC,SETMNT ;ENABLE MAINTENANCE LOOPBACK
1535 014566 012737 014626 001110 MOV #1\$,SLPERR ;FIX START OF LOOP
1536 014574 012777 000007 006770 MOV #DCF,@ACR
1537 014602 017737 006770 023616 MOV @DCF,SNAPO ;SAVE IMAGE OF DCF REGISTER
1538 014610 012777 000004 006754 MOV #IMSKE,@ACR ;POINT TO IMASK
1539 014616 012777 000017 006752 MOV #17,@IMSK ;SET ALL INTERRUPT MASKS
1540 014624 005000 CLR R0 ;INIT FRAME, TYPE, PGTE DATA SETUP
1541
1542 014626 010037 001200 1\$: MOV R0,\$TMP1
1543 014632 004737 022214 JSR PC,PARCAL ;CALCULATE PARITY OF DATA
1544 014636 113737 001200 001201 MOVB \$TMP1,\$TMP1+1
1545 014644 106037 001201 RORB \$TMP1+1 ;POSITION DATA
1546 014650 106037 001201 RORB \$TMP1+1
1547 014654 106037 001201 RORB \$TMP1+1
1548 014660 106037 001201 RORB \$TMP1+1
1549 014664 012777 000000 006700 MOV #PGTEE,@ACR
1550 014672 013777 001200 006676 MOV \$TMP1,@PGTE ;LOAD TRANSMIT ENABLES
1551 014700 006301 ASL R1 ;GET PARITY
1552 014702 042701 177775 2\$: BIC #177775,R1
1553 014706 016437 020516 001124 MOV SIDTAB(4),\$GDDAT ;GET SELF-ID
1554 014714 050137 001124 BIS R1,\$GDDAT
1555 014720 052701 000001 BIS #1,R1
1556 014724 012777 000001 006640 MOV #PGCSE,@ACR
1557 014732 010177 006640 MOV R1,@PGCS ;LOAD PARITY, SET GO
1558 014736 005001 CLR R1
1559 014740 052737 004000 001124 BIS #PRDY,\$GDDAT
1560 014746 105377 006620 3\$: DECB @ACR
1561 014752 017737 006620 001126 MOV @PGCS,\$BDDAT
1562 014760 032737 004000 001126 BIT #PRDY,\$BDDAT ;WAIT FOR READY
1563 014766 001005 BNE 4\$
1564 014770 005301 DEC R1 ;BUT DON'T WAIT TOO LONG
1565 014772 001365 BNE 3\$
1566 014774 104044 ERROR+ 44
1567 014776 004737 022234 JSR PC,DMPREG
1568 015002 023737 001126 001124 4\$: CMP \$BDDAT,\$GDDAT ;IS PGCS OK?
1569 015010 001403 BEQ 5\$
1570 015012 104044 ERROR+ 44
1571 015014 004737 022234 JSR PC,DMPREG
1572 015020 012737 000000 001124 5\$: MOV #0,\$GDDAT ;GET EXPECTED RTE FLAGS
1573 015026 012777 000010 006536 MOV #EXCE,@ACR
1574 015034 017737 006536 001126 MOV @EXC,\$BDDAT ;GET EXC REGISTER
1575 015042 001403 BEQ 6\$
1576 015044 104044 ERROR+ 44
1577 015046 004737 022234 JSR PC,DMPREG
1578 015052 6\$:

1608
1609
1610
(3)
(3)
(2) 015226 000004
1611 015230 004737 022042
1612 015234 013701 023572
1613 015240 013702 023576
1614 015244 012711 000000
1615 015250 016412 020526
1616 015254 012712 000001
1617 015260 005311
1618 015262 005212
1619 015264 016437 020516 001124
1620 015272 062737 144010 001124
1621 015300 012711 000001 \$:
1622 015304 011237 001126
1623 015310 032737 004000 001126
1624 015316 001770
1625 015320 023737 001124 001126
1626 015326 001403
1627 015330 104044
1628 015332 004737 022234
1629
1630 015336 012711 000005 2\$:
1631 015342 012712 000017
1632 015346 017746 006232
1633 015352 017746 006224
1634 015356 012777 015426 006216
1635 015364 012777 000340 006212
1636 015372 012711 000001
1637 015376 012712 000004
1638 015402 005037 177776
1639 015406 000240
1640 015410 012737 000340 177776
1641 015416 104044
1642 015420 004737 022234
1643 015424 000401
1644 015426 022626 3\$:
1645 015430 012677 006146 4\$:
1646 015434 012677 006144
1647 015440 005311
1648 015442 012712 100000
1649 015446 042737 140010 001124
1650 015454 005311
1651 015456 011237 001126
1652 015462 023737 001124 001126
1653 015470 001403
1654 015472 104044
1655 015474 004737 022234
1656
1657 015500 \$:

;TEST ALL THE PGCS ERROR BITS FOR PROPER OPERATION

;TEST 50 TEST GRJ FOR PROPER OPERATION

TST50: 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
\$: 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 @I1IP,-(SP) ;CLEAR PGF
MOV @I1IV,-(SP) ;SAVE OLD
MOV #3\$,@I1IV ;PI DATA
MOV #340,@I1IP ;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\$;SKIP
3\$: POPPOP ;FIX STACK
4\$: MOV (SP)+,@I1IV ;RESTORE OLD
MOV (SP)+,@I1IP ;PI INFO
DEC @R1 ;SELECT PGCS
MOV #ERR,@R2 ;CLEAR ERR BY WRITING 1 TO IT
BIC #ERR+GRJ+IP,\$GDDAT
DEC @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,GRJ, AND IP
5\$:

```

1659          :*****
(3)          :*TEST 51      TEST PG RMR FOR PROPER OPERATION
(3)          :*****
(2) 015500 000004 TST51: SCOPE
1660 015502 004737 022042 JSR PC,SETMNT ;CLR IIST AND ENTER MAINT. MODE
1661 015506 013701 023572 MOV ACR,R1 ;R1 ACR
1662 015512 013702 023576 MOV ADR,R2 ;R2=ADR
1663 015516 012711 000000 MOV #PGTEE,@R1 ;SELECT PGTE
1664 015522 016412 020526 MOV PGITAB(4),@R2 ;LOAD PGTE WITH ENABLE
1665 015526 012712 000001 MOV #GO,@R2 ;TRANSMIT
1666 015532 012711 000000 MOV #PGTEE,@R1 ;SELECT PGTE
1667 015536 005212 INC @R2 ;ACCESS PGTE AGAIN
1668 015540 016437 020516 001124 MOV SIDTAB(4),%GDDAT;LOAD EXPECTED
1669 015546 062737 124010 001124 ADD #ERR+PGMR+PRDY-IP,%GDDAT
1670 015554 012711 000001 1$: MOV #PGCSE,@R1 ;SELECT PGCS
1671 015560 011237 001126 MOV @R2,%BDDAT ;READ PGCS
1672 015564 032737 004000 001126 BIT #PRDY,%BDDAT ;TEST FOR RDY
1673 015572 001770 BEQ 1$ ;LOOP UNTIL SET
1674 015574 023737 001124 001126 (MP %GDDAT,%BDDAT ;CHECK STATUS
1675 015602 001403 BEQ 2$ ;SKIP IF OK
1676 015604 104044 ERROR+ 44 ;ERR,PGMR, OR IP DID NOT SET ON
1677 015606 004737 022234 JSR PC,DMPREG
1678          ;A PGMR ERROR
1679 015612 012711 000005 2$: MOV #PGFE,@R1 ;SELECT PGF
1680 015616 012712 000017 MOV #17,@R2 ;CLEAR PGF
1681 015622 012711 000001 MOV #PGCSE,@R1 ;SELECT PGCS
1682 015626 012712 100000 MOV #ERR,@R2 ;CLEAR ERR BY WRITING 1 TO IT
1683 015632 042737 120010 001124 BIC #ERR+PGMR+IP,%GDDAT
1684 015640 012711 000001 MOV #PGCSE,@R1 ;SELECT PGCS
1685 015644 011237 001126 MOV @R2,%BDDAT ;READ PGCS
1686 015650 023737 001124 001126 (MP %GDDAT,%BDDAT ;CHECK STATUS
1687 015656 001403 BEQ 5$ ;SKIP IF OK
1688 015660 104044 ERROR+ 44 ;WRITING A ONE TO ERR DID NOT
1689 015662 004737 022234 JSR PC,DMPREG
1690          ;CLEAR ERR,PGMR, AND IP
1691 015666          3$:
    
```

1693
(3)
(3)
(2)
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731

015666 000004
015670 004737 022204
015674 013701 023572
015700 013702 023576
015704 012711 000002
015710 016412 020526
015714 012712 000001
015720 016437 020516 001124
015726 062737 114010 001124
015734 012711 000003
015740 032712 000010
015744 001773
015746 012711 000002
015752 005212
015754 012711 000001
015760 011237 001126
015764 023737 001124 001126
015772 001403
015774 104044
015776 004737 022234

016002 012746 100000
016006 005216
016010 001376
016012 005026
016014 012711 000006
016020 012712 000017
016024 012711 000001
016030 012712 100000
016034 042737 110010 001124
016042 012711 000001
016046 011237 001126
016052 023737 001124 001126
016060 001403
016062 104044
016064 004737 022234

016070
016070 000207

```
*****  
*TEST 52 TEST ST RMR FOR PROPER OPERATION  
*****  
TST52: SCOPE  
JSR PC,SSTMNT ;CLR IIST AND ENTER MAINT. MODE  
MOV ACR,R1 ;R1=ACR  
MOV ADR,R2 ;R2=ADR  
MOV #STEE,@R1 ;SELECT STTE  
MOV STITAB(4),@R2 ;LOAD STTE 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 STTE  
INC @R2 ;ACCESS STTE AGAIN  
MOV #PGCSE,@R1 ;SELECT PGCS  
MOV @R2,$BDDAT ;READ PGCS  
1709 CMP $GDDAT,$BDDAT ;CHECK STATUS  
1710 BEQ 2$ ;SKIP IF OK  
1711 ERROR+ 44 ;ERR,STMR, OR IP DID NOT SET ON  
1712 JSR PC,DMPREG  
  
1714 2$: MOV #100000,-(SP) ;A STMR ERROR  
1715 4$: JNC (SP) ;PUSH VALUE  
BNE 4$ ;STALL  
CLR (SP)+  
1718 MOV #STFE,@R1 ;SELECT STF  
1719 MOV #17,@R2 ;CLEAR STF  
1720 MOV #PGCSE,@R1 ;SELECT PGCS  
1721 MOV #ERR,@R2 ;CLEAR ERR BY WRITING 1 TO IT  
1722 BIC #ERR+STMR+IP,$GDDAT  
1723 MOV #PGCSE,@R1 ;SELECT PGCS  
1724 MOV @R2,$BDDAT ;READ PGCS  
1725 CMP $GDDAT,$BDDAT ;CHECK STATUS  
1726 BEQ 5$ ;SKIP IF OK  
1727 ERROR+ 44 ;WRITING A ONE TO ERR DID NOT  
1728 JSR PC,DMPREG  
  
1730 5$: ;CLEAR ERR,STMR, AND IP  
1731 RTS PC ;:EXIT
```

```

1733 ;ISSUE A PROGRAM-GENERATED REQUEST WITH INTERRUPT MASKS CLEAR WITH ALL COMBINATIONS OF
1734 ;TRANSMIT ENABLES AND CHECK FOR:
1735 ;
1736 ;   CORRECT CONTENTS OF PGCS (RDY, IP, SELF-ID)
1737 ;   RTE BITS IN THE EXC REGISTER TO BE CLEAR
1738 ;   ALL OTHER FLAGS TO BE CLEAR ( STF, DCF, UIF)
  
```

```

1739 016072 PGONL:
1740 ;*****
1741 (3) ;*TEST 53 CHECK OPERATION OF PG REQUESTS IN NORMAL MODE
1742 (3) ;*****
1743 (2) 016072 000004 TST53: SCOPE
1741 016074 012777 100000 005470 MOV #CLR,@ACR ;CLEAR THE IIST
1742 016102 005737 023660 TST CONFIG ;ARE WE PREVENTED FROM ONLINE? (CONFIG=0)
1743 016106 001002 BNE .+6 ;SKIP IF NOT
1744 016110 000137 016716 JMP 10$ ;IF YES, GO TO NEXT TEST.
1745 016114 005000 CLR R0 ;STALL
1746 016116 012777 100000 005446 55$: MOV #CLR,@ACR
1747 016124 105200 INCB R0
1748 016126 001373 BNE 55$
1749 016130 012737 016154 001110 MOV #1$, $LPERR ;FIX START OF LOOP
1750 016136 012777 000007 005426 MOV #DCFE,@ACR
1751 016144 017737 005426 023616 MOV @DCF,$NAPO ;SAVE IMAGE OF DCF REGISTER
1752 016152 005000 CLR R0 ;INIT PGTE DATA SETUP
1753 016154 012777 100000 005410 1$: MOV #CLR,@ACR ;CLEAR THE IIST
1754 016162 012777 000005 005402 MOV #PGFE,@ACR ;SELECT PGF
1755 016170 012777 007400 005400 MOV #7400,@PGF ;CLEAR ALL PGF FLAGS
1756 016176 012777 007400 005372 MOV #7400,@STF ;CLEAR ALL STF FLAGS
1757 016204 012777 000017 005364 MOV #17,@DCF ;CLR BRK
1758 016212 012777 000015 005352 MOV #MTCE,@ACR ;SELECT MTC
1759 016220 012777 000010 005350 MOV #DSBT,@MTC ;DISABLE BOOTS
1760 016226 010037 001200 MOV R0,$TMP1
1761 016232 023727 023660 000003 CMP CONFIG,#3 ;SHOULD WE PREVENT XMIT OF ALL CODES?
1762 016240 001013 BNE 11$ ;NO IF NOT CONFIG #3.
1763 016242 016401 020526 MOV PGITAB(4),R1 ;YES -- GET MASK FOR SELF.
1764 016246 010146 MOV R1,-(SP) ;SAVE INTR ENB
1765 016250 006301 ASL R1 ;SHIFT 4X TO BOOT ENB
1766 016252 006301 ASL R1
1767 016254 006301 ASL R1
1768 016256 006301 ASL R1
1769 016260 052601 BIS (SP)+,R1 ;GET INTR ENB MASK
1770 016262 005101 COM R1
1771 016264 040137 001200 BIC R1,$TMP1 ;CLEAR BITS FOR DESTINATION OTHER THAN SELF.
1772 016270 11$:
1773 016270 010046 MOV R0,-(SP) ;SAVE COUNT VALUE
1774 016272 013700 001200 MOV $TMP1,R0 ;GET ENABLE FOR PARITY GENERATOR
1775 016276 004737 022214 JSR PC,PARCAL ;CALCULATE PARITY OF DATA
1776 016302 012600 MOV (SP)+,R0 ;RESTORE COUNT VALUE.
1777 016304 113737 001200 001201 MOVB $TMP1,$TMP1+1
1778 016312 106037 001201 RORB $TMP1+1 ;POSITION DATA
1779 016316 106037 001201 RORB $TMP1+1
1780 016322 106037 001201 RORB $TMP1+1
1781 016326 106037 001201 RORB $TMP1+1
1782 016332 012777 000000 005232 MOV #PGTEE,@ACR ;LOAD TRANSMIT ENABLES
1783 016340 013777 001200 005230 MOV $TMP1,@PGTE ;GET PARITY
1784 016346 006301 ASL R1 ;FIX PAR BIT
1785 016350 005101 COM R1
  
```

D 6

MAINDEC-11-CRIIA MACY11 30A(1052) 30-NOV-78 11:06 PAGE 51-1
 CRIIAA.P11 30-NOV-78 10:58 T53 CHECK OPERATION OF PG REQUESTS IN NORMAL MODE SEQ 0068

1786	016352	042701	177775		2\$:	BIC	#177775,R1	
1787	016356	016437	020516	001124		MOV	SIDTAB(4), \$GDDAT	;GET SELF-ID
1788	016364	050137	001124			BIS	R1, \$GDDAT	
1789	016370	052701	000001			BIS	#1, R1	
1790	016374	012777	000001	005170		MOV	#PGCSE, @ACR	
1791	016402	010177	005170			MOV	R1, @PGCS	;LOAD PARITY, SET GO
1792	016406	005001				CLR	R1	
1793	016410	052737	004000	001124		BIS	#PRDY, \$GDDAT	
1794	016416	036437	020526	001200		BIT	PGITAB(4), \$TMP1	;TEST FOR PGI ENABLE
1795	016424	001403				BEQ	3\$;SKIP IF CLEAR
1796	016426	052737	000010	001124		BIS	#IP, \$GDDAT	;SET IP IF RELEVANT
1797	016434	105377	005132		3\$:	DECB	@ACR	
1798	016440	017737	005132	001126		MOV	@PGCS, \$BDDAT	
1799	016446	032737	004000	001126		BIT	#PRDY, \$BDDAT	;WAIT FOR READY
1800	016454	001005				BNE	4\$	
1801	016456	005301				DEC	R1	;BUT DON'T WAIT TOO LONG
1802	016460	001365				BNE	3\$	
1803	016462	104044				ERROR+	44	;PRDY DID NOT SET ON A PG XFER
1804	016464	004737	022234			JSR	PC, DMPREG	
1805	016470	023737	001126	001124	4\$:	CMP	\$BDDAT, \$GDDAT	;IS PGCS OK?
1806	016476	001403				BEQ	5\$	
1807	016500	104044				ERROR+	44	;PGCS STATUS ERROR ON PG XFER
1808	016502	004737	022234			JSR	PC, DMPREG	
1809	016506	005037	001124		5\$:	CLR	\$GDDAT	;SET EXPECTED RTE FLAGS
1810	016512	012777	000010	005052		MOV	#EXCE, @ACR	
1811	016520	017737	005052	001126		MOV	@EXC, \$BDDAT	;GET EXC REGISTER
1812	016526	023737	001124	001126		CMP	\$GDDAT, \$BDDAT	
1813	016534	001403				BEQ	6\$	
1814	016536	104044				ERROR+	44	;EXC NOT 0 ON A PG XFER
1815	016540	004737	022234			JSR	PC, DMPREG	
1816	016544	036437	020526	001200	6\$:	BIT	PGITAB(4), \$TMP1	;CHECK FOR FGAL PGF
1817	016552	001403				BEQ	61\$;SKIP IF NO
1818	016554	056437	020526	001124		BIS	PGITAB(4), \$GDDAT	;LOAD PGF
1819	016562	036437	020536	001200	61\$:	BIT	PGBTAB(4), \$TMP1	;CHECK FOR LEGAL PGB
1820	016570	001403				BEQ	62\$;SKIP IF NO
1821	016572	056437	020536	001124		BIS	PGBTAB(4), \$GDDAT	;LOAD PGB
1822	016600	012777	000005	004764	62\$:	MOV	#PGFE, @ACR	
1823	016606	017737	004764	001126		MOV	@PGF, \$BDDAT	;READ PGF
1824	016614	023737	001124	001126		CMP	\$GDDAT, \$BDDAT	;CHECK PGF
1825	016622	001403				BEQ	7\$	
1826	016624	104044				ERROR+	44	;PGF ERROR ON PG XFER
1827	016626	004737	022234			JSR	PC, DMPREG	
1828	016632	005037	001124		7\$:	CLR	\$GDDAT	;EXPECTED = 0
1829	016636	017737	004734	001126		MOV	@STF, \$BDDAT	;GET STF
1830	016644	001403				BEQ	8\$;STF SHOULD BE 0
1831	016646	104044				ERROR+	44	;STF NO 0 ON A PG XFER
1832	016650	004737	022234			JSR	PC, DMPREG	
1833	016654	013737	023616	001124	8\$:	MOV	SNAPO, \$GDDAT	;GET EXPECTED DATA FOR DCF
1834	016662	017737	004710	001126		MOV	@DCF, \$BDDAT	;GET DCF REGISTER
1835	016670	023737	001126	001124		CMP	\$BDDAT, \$GDDAT	;CHECK THAT THERE IS NO CHANGE
1836	016676	001403				BEQ	9\$	
1837	016700	104044				ERROR+	44	;DCF ERROR ON A PG XFER
1838	016702	004737	022234			JSR	PC, DMPREG	
1839	016706	105200			9\$:	INCB	R0	;INCREMENT TEST DATA
1840	016710	001402				BEQ	10\$;SKIP WHEN DONE
1841	016712	000137	016154			JMP	1\$	

MAINDEC-11-CRIIA MACY11 30A(1052) 30-NOV-78 11:06 PAGE 51-2
CRIIAA.P11 30-NOV-78 10:58 T53 CHECK OPERATION OF PG REQUESTS IN NORMAL MODE

SEQ 0069

842 016716

10\$:

1844
1845
1846
1847
(3)
(3)
(2) 016716 000004
1848 016720 012777 100000 004644
1849 016726 012737 000010 001200
1850 016734 012737 000000 001202
1851 016742 005737 023660
1852 016746 001005
1853 016750 004737 022042
1854 016754 012737 000012 001200
1855 016762 016401 020536
1856 016766 056401 020526
1857 016772 005101
1858 016774 042701 170360
1859 017000 010137 001202
1860 017004 012777 000007 004560
1861 017012 017737 004560 023616
1862 017020 013737 023616 001124
1863 017026 023727 023660 000001
1864 017034 001003
1865 017036 012737 007400 001124
1866 017044 005737 023660
1867 017050 001002
1868 017052 005037 001124
1869 017056 046437 020536 001124
1870 017064 013737 023616 001126
1871 017072 023737 001124 001126
1872 017100 001403
1873 017102 104044
1874 017104 004737 022234
1875 017110 012777 000015 004454
1876 017116 012777 000011 004452
1877 017124 012777 000007 004440
1878 017132 017737 004440 001126
1879 017140 043737 001202 001126
1880 017146 016437 020536 001124
1881 017154 056437 020526 001124
1882 017162 023737 001124 001126
1883 017170 001403
1884 017172 104044
1885 017174 004737 022234
1886 017200 012777 000015 004364
1887 017206 013777 001200 004362
1888 017214 012777 000007 004350
1889 017222 017737 004350 001126
1890 017230 043737 001202 001126
1891 017236 046437 020536 001124
1892 017244 023737 001124 001126
1893 017252 001403
1894 017254 104044
1895 017256 004737 022234
1896 017262 016437 020516 001124

:FORCE A BREAK USING THE MAINTENANCE REGISTER AND TEST
:THE DCF REGISTER
:*****
:TEST 54 TEST THE DCF REGISTER
:*****
TST54: SCOPE
MOV #CLR,@ACR ;INIT THE IIST
MOV #DSBT,\$TMP1 ;SET VALUE FOR MTC TO CLEAR BREAK.
MOV #0,\$TMP2 ;SET VALUE FOR CLEARING UNWANTED BITS.
TST CONFIG ;SEE WHAT CONFIGURATION.
BNE 11\$;IF NOT MAINT., SKIP ON.
JSR PC,SETMNT ;IF MAINT. ONLY, SET MAINT. LOOPBACK.
MOV #MLEN+DSBT,\$TMP1 ;NEW VALUE FOR CLEARING BRK.
11\$: MOV BRKTAB(4),R1 ;GET BRK BIT FOR OWN LINE.
BIS DCFTAB(4),R1 ;AND DCF BIT.
COM R1 ;MAKE 'BIC' MASK.
BIC #170360,R1
MOV R1,\$TMP2 ;STORE THE VALUE.
MOV #DCFE,@ACR ;SELECT DCF
MOV @DCF,\$SAPD ;SAVE THE DCF CONTENTS
MOV \$SAPD,\$GDDAT ;COPY DCF
CMP CONFIG,#1 ;ANY OTHER INTERFACES?
BNE 12\$
MOV #7400,\$GDDAT ;SAY EXPECTED HAS OTHER BRK BITS SET.
12\$: TST CONFIG ;IN MAINT. MODE?
BNE 13\$
CLR \$GDDAT ;YES -- ALL DCF BITS CLEAR.
13\$: BIC BRKTAB(4),\$GDDAT ;REMOVE SID BRK
MOV \$SAPD,\$BDDAT ;READ ACTUAL
CMP \$GDDAT,\$BDDAT ;CHECK FOR LEGAL DCF
BEQ 1\$;SKIP IF OK
ERROR+ 44 ;INCORRECT DCF FOR SELECTED IIST
JSR PC,DMPREG
1\$: MOV #MTC,@ACR ;SELECT MTC
MOV #DSBT+MDS,@MTC ;FORCE A BREAK
MOV #DCFE,@ACR ;SELECT DCF
MOV @DCF,\$BDDAT ;READ DCF
BIC \$TMP2,\$BDDAT ;CLEAR BITS NOT OUR OWN.
MOV BRKTAB(4),\$GDDAT ;SET RELEVANT BRK BIT
BIS DCFTAB(4),\$GDDAT ;SET RELEVANT DCF BIT
CMP \$GDDAT,\$BDDAT ;CHECK DCF
BEQ 2\$;SKIP IF OK
ERROR+ 44 ;INCORRECT DCF AFTER FORCING A BREAK
JSR PC,DMPREG
2\$: MOV #MTC,@ACR ;SELECT MTC
MOV \$TMP1,@MTC ;CLEAR BRK BIT
MOV #DCFE,@ACR ;SELECT DCF
MOV @DCF,\$BDDAT ;READ DCF
BIC \$TMP2,\$BDDAT
BIC BRKTAB(4),\$GDDAT ;CLEAR OUT BRK BIT
CMP \$GDDAT,\$BDDAT ;CHECK DCF
BEQ 3\$;SKIP IF BRK=0
ERROR+ 44 ;INCORRECT DCF AFTER REMOVING THE BRK CONDITION
JSR PC,DMPREG
3\$: MOV SIDTAB(4),\$GDDAT ;LOAD SELF ID #

1897	017270	052737	004010	001124		BIS	#IP+PRDY,\$GDDAT	:ADD OTHER EXPECTED DATA
1898	017276	012777	000007	004266		MOV	#DCFE,@ACR	:SELECT DCF
1899	017304	013777	001202	004264		MOV	\$TMP2,@DCF	:CLEAR UNWANTED FLAGS.
1900	017312	012777	000001	004252		MOV	#PGCSE,@ACR	:SELECT PGCS
1901	017320	017737	004252	001126		MOV	@PGCS,\$BDDAT	:READ PGCS
1902	017326	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	:CHECK PGCS FOR IP
1903	017334	001403				BEQ	4\$:SKIP IF OK
1904	017336	104044				ERROR+	44	:DCF FLAG DID NOT SET IP
1905	017340	004737	022234			JSR	PC,DMPREG	
1906	017344	017746	004232		4\$:	MOV	@IIIV,-(SP)	:SAVE OLD PI
1907	017350	017746	004230			MOV	@IIIP,-(SP)	:DATA
1908	017354	012777	017430	004220		MOV	#5\$,@IIIV	:LOAD NEW
1909	017362	012777	000340	004214		MOV	#340,@IIIP	:PI DATA
1910	017370	012777	000001	004174		MOV	#PGCSE,@ACR	:SELECT PGCS
1911	017376	012777	000004	004172		MOV	#IE,@PGCS	:ENABLE PI
1912	017404	005037	177776			CLR	PS	:ALLOW PI
1913	017410	000240				NOP		:STALL
1914	017412	012737	000340	177776		MOV	#340,PS	:LOCKOUT PI
1915	017420	104044				ERROR+	44	:NO PI ON A DCF FLAG
1916	017422	004737	022234			JSR	PC,DMPREG	
1917	017426	000401				BR	6\$:SKIP
1918	017430	022626			5\$:	POPPOP		:FIX STACK
1919	017432	012677	004146		6\$:	MOV	(SP)+,@IIIP	:RESTORE OLD
1920	017436	012677	004140			MOV	(SP)+,@IIIV	:PI DATA
1921	017442	012737	000000	001124		MOV	#0,\$GDDAT	:RELOAD EXPECTED DCF VALUE
1922	017450	012777	000007	004114		MOV	#DCFE,@ACR	:SELECT DCF
1923	017456	016477	020526	004112		MOV	DCFTAB(4),@DCF	:CLEAR DCF FLAG
1924	017464	005377	004102			DEC	@ACR	:RESELECT DCF
1925	017470	017737	004102	001126		MOV	@DCF,\$BDDAT	:READ DCF
1926	017476	043737	001202	001126		BIC	\$TMP2,\$BDDAT	:MASK OUT JUNK.
1927	017504	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	:CHECK DCF
1928	017512	001403				BEQ	7\$:SKIP IF DCF=0
1929	017514	104044				ERROR+	44	:WRITING A 1 TO DCF FLAG DID NOT CLEAR IT
1930	017516	004737	022234			JSR	PC,DMPREG	
1931	017522	012777	100000	004042	7\$:	MOV	#CLR,@ACR	:INIT. THE INTERFACE.
1932	017530	012777	000004	004034		MOV	#IMSK,@ACR	:SELECT THE IMSK REGISTER.
1933	017536	012777	000017	004032		MOV	#17,@IMSK	:SET ALL INTERRUPT MASKS.
1934	017544	012777	000015	004020		MOV	#MTC,@ACR	:SELECT MTC.
1935	017552	012777	000012	004016		MOV	#DSBT+MLEN,@MTC	:DO MAINT. LOOP TO CLEAR BREAKS.
1936	017560	027777	004006	004004		CMP	@ACR,@ACR	:DELAY
1937	017566	027777	004000	003776		CMP	@ACR,@ACR	:
1938	017574	027777	003772	003770		CMP	@ACR,@ACR	:
1939	017602	012777	000011	003766		MOV	#DSBT+MDSO,@MTC	:CAUSE BREAK
1940	017610	027777	003756	003754		CMP	@ACR,@ACR	
1941	017616	027777	003750	003746		CMP	@ACR,@ACR	
1942	017624	012737	000000	001124		MOV	#0,\$GDDAT	:SAY GOOD IS 0
1943	017632	012777	000007	003732		MOV	#DCFE,@ACR	:SELECT DCF
1944	017640	017737	003732	001126		MOV	@DCF,\$BDDAT	:GET DCF
1945	017646	042737	007400	001126		BIC	#7400,\$BDDAT	:CLEAR BREAKS
1946	017654	023737	001126	001124		CMP	\$BDDAT,\$GDDAT	:VERIFY
1947	017662	001403				BEQ	8\$	
1948	017664	104044				ERROR+	44	
1949	017666	004737	022234			JSR	PC,DMPREG	
1950	017672	000207			8\$:	RTS	PC	:EXIT


```

1952 ;ISSUE A SANITY TIMER REQUEST WITH INTERRUPT MASKS CLEAR WITH ALL COMBINATIONS OF
1953 ;TRANSMIT ENABLES AND CHECK FOR:
1954 ; CORRECT CONTENTS OF PGCS (RDY, IP, SELF-ID)
1955 ; RTE BITS IN THE EXC REGISTER TO BE CLEAR
1956 ; ALL OTHER FLAGS TO BE CLEAR (PGF, DCF, UIF)
1957 ;

```

1958 017674

STONL:

```

1959 ;*****
(3) ;*TEST 55 CHECK OPERATION OF ST REQUESTS IN NORMAL MODE
(3) ;*****

```

```

(2) 017674 000004
1960 017676 012777 100000 003666
1961 017704 005737 023660
1962 017710 001001
1963 017712 000207
1964 017714 012737 017740 001110
1965 017722 012777 000007 003642
1966 017730 017737 003642 023616
1967 017736 005000
1968 017740 012777 100000 003624 1$:
1969 017746 012777 000005 003616
1970 017754 012777 007400 003614
1971 017762 012777 007400 003606
1972 017770 012777 000017 003600
1973 017776 012777 000015 003566
1974 020004 012777 000010 003564
1975 020012 010037 001200
1976 020016 023727 023660 000003
1977 020024 001013
1978 020026 016401 020526
1979 020032 010146
1980 020034 006301
1981 020036 006301
1982 020040 006301
1983 020042 006301
1984 020044 052601
1985 020046 005101
1986 020050 040137 001200
1987 020054
1988 020054 010046
1989 020056 013700 001200
1990 020062 004737 022214
1991 020066 012600
1992 020070 113737 001200 001201
1993 020076 106037 001201
1994 020102 106037 001201
1995 020106 106037 001201
1996 020112 106037 001201
1997 020116 012777 000002 003446
1998 020124 013777 001200 003444
1999 020132 006301
2000 020134 005101
2001 020136 042701 177775 2$:
2002 020142 010137 001124
2003 020146 052701 000001
2004 020152 012777 000003 003444

```

```

SCOPE
MOV #CLR,@ACR ;CLEAR THE IIST
TST CONFIG
BNE .+4
RTS PC ;EXIT IF WE CAN'T XMIT
MOV #1$,SLPERR ;FIX START OF LOOP
MOV #DCF,@ACR
MOV @DCF,SNAPO ;SAVE IMAGE OF DCF REGISTER
CLR R0 ;INIT SITE DATA SETUP
1$: MOV #CLR,@ACR ;CLEAR THE IIST
MOV #PGFE,@ACR ;SELECT PGF
MOV #7400,@PGF ;CLEAR ALL
MOV #7400,@STF ;FLAGS
MOV #17,@DCF ;CLEAR BRK
MOV #MTCF,@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 STITAB(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.
RORB $TMP1+1 ;POSITION DATA
RORB $TMP1+1
RORB $TMP1+1
RORB $TMP1+1
MOV #STEE,@ACR
MOV $TMP1,@STTE ;LOAD TRANSMIT ENABLES
ASL R1 ;GET PARITY
COM R1 ;FIX PAR BIT
2$: BIC #177775,R1 ;LOAD PARITY
MOV R1,$GDDAT
BIS #,R1
MOV #STCSE,@ACR

```

2005	020160	010177	003412			MOV	R1,@STCS	;LOAD PARITY, ENB, AND COUNT
2006	020164	005001				CLR	R1	
2007	020166	052737	177411	001124		BIS	#TMO+177401,\$GDDAT	
2008	020174	012777	000004	003370		MOV	#IMSKE,@ACR	;SELECT IMSK
2009	020202	105377	003364		3\$:	DECB	@ACR	
2010	020206	017737	003364	001126		MOV	@STCS,\$BDDAT	
2011	020214	032737	000010	001126		BIT	#TMO,\$BDDAT	;WAIT FOR TMO
2012	020222	001005				BNE	4\$	
2013	020224	005301				DEC	R1	;BUT DON'T WAIT TOO LONG
2014	020226	001365				BNE	3\$	
2015	020230	104044				ERROR+	44	;TMO DID NOT SET ON A ST XFER
2016	020232	004737	022234			JSR	PC,DMPREG	
2017	020236	012777	000003	003326	4\$:	MOV	#STCSE,@ACR	;SELECT STCS
2018	020244	017737	003326	001126		MOV	@STCS,\$BDDAT	;READ STCS
2019	020252	023737	001126	001124		CMP	\$BDDAT,\$GDDAT	;IS STCS OK?
2020	020260	001403				BEQ	5\$	
2021	020262	104044				ERROR+	44	;STCS STATUS ERROR ON ST XFER
2022	020264	004737	022234			JSR	PC,DMPREG	
2023	020270	005037	001124		5\$:	CLR	\$GDDAT	;SET EXPECTED RTE FLAGS
2024	020274	012777	000010	003270		MOV	#EXCE,@ACR	
2025	020302	017737	003270	001126		MOV	@EXC,\$BDDAT	;GET EXC REGISTER
2026	020310	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	
2027	020316	001403				BEQ	6\$	
2028	020320	104044				ERROR+	44	;EXC NOT 0 ON A ST XFER
2029	020322	004737	022234			JSR	PC,DMPREG	
2030	020326	036437	020526	001200	6\$:	BIT	STITAB(4),\$TMP1	;CHECK FOR STF
2031	020334	001403				BEQ	61\$;SKIP IF 0
2032	020336	056437	020526	001124		BIS	STITAB(4),\$GDDAT	;SET STF
2033	020344	036437	020536	001200	61\$:	BIT	STBTAB(4),\$TMP1	;CHECK FOR STB
2034	020352	001403				BEQ	62\$;SKIP IF 0
2035	020354	056437	020536	001124		BIS	STBTAB(4),\$GDDAT	;SET STB
2036	020362	012777	000006	003202	62\$:	MOV	#STFE,@ACR	
2037	020370	017737	003202	001126		MOV	@STF,\$BDDAT	;READ STF
2038	020376	023737	001124	001126		CMP	\$GDDAT,\$BDDAT	;CHECK DATA
2039	020404	001403				BEQ	7\$	
2040	020406	104044				ERROR+	44	;STF ERROR ON ST XFER
2041	020410	004737	022234			JSR	PC,DMPREG	
2042	020414	005037	001124		7\$:	CLR	\$GDDAT	;EXPECTED = 0
2043	020420	012777	000005	003144		MOV	#PGFE,@ACR	;SELECT PGF
2044	020426	017737	003144	001126		MOV	@PGF,\$BDDAT	;GET PGF
2045	020434	001403				BEQ	8\$;PGF SHOULD BE 0
2046	020436	104044				ERROR+	44	;PGF NO 0 ON A ST XFER
2047	020440	004737	022234			JSR	PC,DMPREG	
2048	020444	013737	023616	001124	8\$:	MOV	SNAPO,\$GDDAT	;GET EXPECTED DATA FOR DCF
2049	020452	012777	000007	003112		MOV	#DCFE,@ACR	;SELECT DCF
2050	020460	017737	003112	001126		MOV	@DCF,\$BDDAT	;GET DCF REGISTER
2051	020466	023737	001126	001124		CMP	\$BDDAT,\$GDDAT	;CHECK THAT THERE IS NO CHANGE
2052	020474	001403				BEQ	9\$	
2053	020476	104044				ERROR+	44	;DCF ERROR ON A ST XFER
2054	020500	004737	022234			JSR	PC,DMPREG	
2055	020504	105200			9\$:	INCB	R0	;INCREMENT TEST DATA
2056	020506	001402				BEQ	10\$;SKIP WHEN DONE
2057	020510	000137	017740			JMP	1\$	
2058	020514	000207			10\$:	RTS	PC	;EXIT

```
2060                                     .SBTTL LINE CONTROL BIT TABLES
2061
2062 020516 000000          SIDTAB: SID0          ;SID TABLE
2063 020520 000400          SID1
2064 020522 001000          SID2
2065 020524 001400          SID3
2066
2067 020526          PGITAB:          ;PROGRAM INTERRUPT ENABLE TABLE
2068 020526          SITAB:          ;SANITY INTERRUPT ENABLE TABLE
2069 020526          IMITAB:         ;INTERRUPT MASK INHIBIT TABLE
2070 020526          PIFTAB:        ;PGM INTERRUPT FLAG TABLE
2071 020526          SIFTAB:        ;SANITY INTERRUPT FLAG TABLE
2072 020526          DCFTAB:        ;DCLO INTERRUPT FLAG TABLE
2073 020526 000001          RTETAB: RTE0          ;INVALID INPUT FLAG TABLE
2074 020530 000002          RTE1
2075 020532 000004          RTE2
2076 020534 000010          RTE3
2077
2078 020536          PGBTAB:          ;PROGRAM BOOT FNABLE TABLE
2079 020536          STBTAB:         ;SANITY BOOT ENABLE TABLE
2080 020536          BMITAB:        ;BOOT MASK INHIBIT TABLE
2081 020536          PSFTAB:        ;PGM BOOT FLAG TABLE
2082 020536          SBF TAB:        ;SANITY BOOT FLAG TABLE
2083 020536          BRKTAB:        ;BREAK STATE TABLE
2084 020536 000400          UVITAB: UI0          ;UNEXPECTED INPUT FLAG TABLE
2085 020540 001000          UI1
2086 020542 002000          UI2
2087 020544 004000          UI3
```

2089 020546

ENDTST:

2090
2091
2092
2093
2094
2095
2096
2097

: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)

:*****
:TEST 56 DETERMINE SANITY TIMER COUNT RATE
:*****

(2)

TST56: SCOPE

2098 020550 012737 000340 177776
2099 020556 004737 022042
2100 020562 012777 000015 003002
2101 020570 012777 000013 003000
2102 020576 117704 002772
2103 020602 010437 023614
2104 020606 006304
2105 020610 012777 000002 002754
2106 020616 016477 020526 002752
2107 020624 017746 002754
2108 020630 017746 002746
2109 020634 012777 020770 002740
2110 020642 012777 000340 002734
2111 020650 005000
2112 020652 013701 023572
2113 020656 013702 023576
2114 020662 012703 000001
2115 020666 012704 004000
2116 020672 012705 000007
2117 020676 012711 000003
2118 020702 012712 000401
2119 020706 005037 177776

MOV #340,PSW ;RAISE PROCESSOR PRIORITY
JSR PC,SETMNT ;INIT THE INTERFACE
MOV #MTC,@ACR ;SELECT MTC REGISTER
MOV #DSBT+MLEN+MDS,@MTC ;SET LOOPBACK, DISABLE BOOT & DRIVERS
MOV @ACRH,R4 ;GET HARDWARE SELF-ID
MOV R4,LINMBR ;PUT IT INTO LINE NUMBER WORD
ASL R4 ;MAKE A WORD INDEX
MOV #STIE,@ACR ;SELECT STIE (S.T. TRANSMIT ENABLE)
MOV STIAB(R4),@STIE ;SET XMIT ENABLE FOR SELF
MOV @IIIP,-(SP) ;SAVE OLD INTR VECTOR INFO
MOV @IIIV,-(SP)
MOV #4,@IIIV ;SETUP NEW INTR VECTOR
MOV #340,@IIIP
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

2120
2121
2122
2123
2124
2125
2126
2127
2128
2129

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

2130

:COME HERE IF NO TIMEOUT INTR SEEN AFTER 64K TRANSMITS.

2131 020732 012737 000340 177776
2132 020740 012737 177410 001124
2133 020746 012711 000003
2134 020752 011237 001126
2135 020756 104044
2136 020760 004737 022234
2137 020764 000137 022014
2138

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

2139

:COME HERE ON INTERRUPT, CHECK THAT IT'S THE CORRECT ONE.

2140 020770 062706 000004
2141 020774 010046

4\$: ADD #4,SP ;POP STACK
MOV R0,-(SP) ;SAVE COUNT

```

2142 020776 010311      MOV      R3,(R1)      ;SELECT PGCS
2143 021000 012705 000062  MOV      #50.,R5     ;WAIT FOR FINAL PG RDY
2144 021004 005305      DEC      R5
2145 021006 001376      BNE      -2
2146 021010 011237 001126  MOV      (R2),$BDDAT ;GET PGCS CONTENTS
2147 021014 011137 001124  MOV      (R1),$GDDAT ;GET SELF-ID
2148 021020 042737 000017 001124  BIC      #17,$GDDAT  ;CLEAR CODE
2149 021026 052737 004016 001124  BIS      #PRDY+IE+IP+PTP,$GDDAT ;SET UP EXPECTED VALUE
2150 021034 023737 001126 001124  CMP      $BDDAT,$GDDAT ;VERIFY PGCS CONTENTS
2151 021042 001403      BEQ      5$
2152 021044 104044      ERROR+  44          ;PGCS INCORRECT AFTER INTERRUPT
2153 021046 004737 022234      JSR      PC,DMPREG
2154 021052 005037 001124 5$:      CLR      $GDDAT     ;SETUP EXPECTED VALUE FOR PGF, DCF, AND EXC
2155 021056 012777 000005 002506  MOV      #PGFE,@ACR  ;SELECT PGF
2156 021064 017737 002506 001126  MOV      @PGF,$BDDAT ;GET PG RECEIVE FLAGS
2157 021072 001403      BEQ      6$
2158 021074 104044      ERROR+  44          ;PGF NOT 0 AFTER INTERRUPT
2159 021076 004737 022234      JSR      PC,DMPREG
2160 021102 012777 000007 002462 6$:      MOV      #DCF,@ACR   ;SELECT DCF
2161 021110 017737 002462 001126  MOV      @DCF,$BDDAT ;GET FLAGS & TEST THEM
2162 021116 001403      BEQ      7$
2163 021120 104044      ERROR+  44          ;DCF NOT 0 AFTER INTERRUPT
2164 021122 004737 022234      JSR      PC,DMPREG
2165 021126 012777 000010 002436 7$:      MOV      #EXCE,@ACR  ;SELECT EXC
2166 021134 017737 002436 001126  MOV      @EXC,$BDDAT ;GET FLAGS & TEST THEM
2167 021142 001403      BEQ      8$
2168 021144 104044      ERROR+  44          ;EXC NOT CLEAR AFTER INTERRUPT
2169 021146 004737 022234      JSR      PC,DMPREG
2170 021152 012737 000017 001124 8$:      MOV      #17,$GDDAT  ;SETUP EXPECTED VALUE FOR STF
2171 021160 012777 000006 002404  MOV      #STFE,@ACR  ;SELECT STF
2172 021166 017737 002404 001126  MOV      @STF,$BDDAT ;GET STF
2173 021174 023737 001126 001124  CMP      $BDDAT,$GDDAT ;CHECK VALUE
2174 021202 001403      BEQ      9$
2175 021204 104044      ERROR+  44          ;STF INCORRECT AFTER INTERRUPT
2176 021206 004737 022234      JSR      PC,DMPREG
2177
2178 021212 012637 001126 9$:      MOV      (SP)+,$BDDAT ;GET COUNT VALUE
2179 021216 005737 023662  TST      STLCNT     ;IS THERE A VALID COUNT?
2180 021222 001402      BEQ      +6
2181 021224 000137 021734  JMP      20$
2182 021230 012737 000014 001124  MOV      #12.,$GDDAT ;IF YES, GO VERIFY CONSISTENCY.
2183 021236 023737 001126 001124  CMP      $BDDAT,$GDDAT ;NO -- SO START CHECKING VALUE.
2184 021244 002002      BGE      +6          ;IS COUNT .LT 14.?
2185 021246 000137 021724  JMP      12$
2186 021252 012737 000024 001124  MOV      #20.,$GDDAT ;GO ON IF NO
2187 021260 023737 001126 001124  CMP      $BDDAT,$GDDAT ;ERROR IF YES
2188 021266 003047      BGT      10$
2189 021270 013737 001126 023662  MOV      $BDDAT,STCNT ;GREATER THAN 20.?
2190 021276 012737 000024 023666  MOV      #20.,MAXCNT ;NO, SO BETWEEN 12 & 21 -- VALID
2191 021304 012737 000014 023664  MOV      #12.,MINCNT ;SAVE MAX LIMIT
2192 021312 005737 000042  TST      @42        ;AND MIN LIMIT, FOR FUTURE TESTING.
2193 021316 001402      BEQ      +6          ;ARE WE IN ACT?
2194 021320 000137 022014  JMP      99$
2195 021324 104401 021332  TYPE     65$        ;BEQ IF NO.
(1) 021330 000424      BR      64$         ;IF YES, DON'T PRINT.
(1)                                ;:TYPE ASCIZ STRING
                                ;:GET OVER THE ASCIZ
                                ;:65$: .ASCIZ <200>/SANITY TIMER SET FOR 256US PER (COUNT.<200>

```

```

(1) 021402
2196 021402 000137 022014 64$: JMP 99$
2197
2198 021406 012737 000620 001124 10$: MOV #400.,$GDDAT
2199 021414 023737 001126 001124 CMP $BDDAT,$GDDAT ;CHECK FOR COUNT BETWEEN 400.
2200 021422 002540 BLT 12$
2201 021424 012737 001200 001124 MOV #640.,$GDDAT
2202 021432 023727 001126 001200 CMP $BDDAT,#640. ;AND 640.
2203 021440 003045 BGT 11$
2204 021442 013737 001126 023662 MOV $BDDAT,STCCNT ;SAVE COUNT
2205 021450 012737 000620 023664 MOV #400.,MINCNT ;AND LIMITS
2206 021456 012737 001200 023666 MOV #640.,MAXCNT
2207 021464 005737 000042 TST @#42 ;ARE WE IN ACT?
2208 021470 001151 BNE 99$ ;IF YES, DON'T PRINT.
2209 021472 104401 021500 TYPE ,67$ ;:TYPE ASCIZ STRING
(1) 021476 000425 BR 66$ ;:GET OVER THE ASCIZ
(1) 66$: .ASCIZ <200>/SANITY TIMER SET FOR 8.192MS PER COUNT./<200>
(1) 021552
2210 021552 000520 BR 99$
2211
2212 021554 012737 001440 001124 11$: MOV #800.,$GDDAT
2213 021562 023737 001126 001124 CMP $BDDAT,$GDDAT ;COUNT .LE 800.?
2214 021570 002455 BLT 12$ ;ERROR IF YES.
2215 021572 012737 002400 001124 MOV #1280.,$GDDAT
2216 021600 023727 001126 002400 CMP $BDDAT,#1280. ;GREATER THAN 1280.?
2217 021606 003046 BGT 12$ ;ERROR IF YES.
2218 021610 013737 001126 023662 MOV $BDDAT,STCCNT ;SAVE COUNT
2219 021616 012737 001440 023664 MOV #800.,MINCNT ;AND LIMITS
2220 021624 012737 002400 023666 MOV #1280.,MAXCNT
2221 021632 005737 000042 TST @#42 ;ARE WE IN ACT?
2222 021636 001066 BNE 99$
2223 021640 104401 021646 TYPE ,69$ ;:TYPE ASCIZ STRING
(1) 021644 000426 BR 68$ ;:GET OVER THE ASCIZ
(1) 68$: .ASCIZ <200>/SANITY TIMER SET FOR 16.384MS PER COUNT./<200>
(1) 021722
2224 021722 000434 BR 99$
2225
2226 ;HERE ON ERROR
2227 021724 104044 12$: ERROR+ 44 ;CAN'T RESOLVE SANITY TIMER VALUE
2228 021726 004737 022234 JSR PC,DMPREG ;(COUNTS NOT WITHIN LIMITS)
2229 021732 000430 BR 99$
2230
2231 ;HERE ON SUBSEQUENT PASSES TO CHECK CONSISTENCY
2232 021734 013737 023666 001124 20$: MOV MAXCNT,$GDDAT ;GET EXPECTED MAX LIMIT FOR TIMEOUT.
2233 021742 013737 023662 023610 MOV STCCNT,DISPLY ;DISPLAY COUNT AS FOUND ON FIRST PASS.
2234 021750 023737 001126 001124 CMP $BDDAT,$GDDAT ;SEE THAT COUNT NOT EXCEEDED.
2235 021756 003404 BLE 21$ ;OK IF .LE MAX LIMIT.
2236 021760 104044 ERROR+ 44 ;COUNT GREATER THAN MAX OF RANGE.
2237 021762 004737 022234 JSR PC,DMPREG
2238 021766 000412 BR 99$
2239 021770 013737 023664 001124 21$: MOV MINCNT,$GDDAT ;GET EXPECTED MIN LIMIT.
2240 021776 023737 001126 001124 CMP $BDDAT,$GDDAT ;CHECK
2241 022004 003003 BGT 99$ ;OK IF ACTUAL .GT MIN LIMIT.
2242 022006 104044 ERROR+ 44 ;COUNT LESS THAN MIN OF RANGE.
2243 022010 004737 022234 JSR PC,DMPREG
2244
  
```

MAINDEC-11-CRIIA	MACY11	30A(1052)	30-NOV-78	11:06	PAGE 55-3	N 6	
CRIIAA.P11	30-NOV-78	10:58	T56	DETERMINE SANITY TIMER COUNT RATE			SEQ 0078
2245	022014	012677	001562	99\$:	MOV	(SP)+,@IIIV	;RESTORE VECTOR
2246	022020	012677	001560		MOV	(SP)+,@IIIP	:
2247	022024	012777	100000	001540	MOV	#CLR,@ACR	;INIT THE INTERFACE
2248							
2249							
2250	022032	000005			RESET		
2251	022034	000240			NOP		
2252	022036	000137	023670		JMP	\$EOP	

```

2254          .SBTTI  TEST AND UTILITY SUBROUTINES
2255
2256          ;SET THE IIST TO INTERNAL LOOPBACK MAINTENENCE MODE USING
2257          ;THE SID VALUE POINTED TO BY R4
2258
2259 022042 005037 022202          SETMNT: CLR      MNTBTS          ;CLEAR EXTRA MNT BITS
2260 022046 012777 100000 001516 SETMNT1: MOV     #CLR,@ACR      ;RESET THE INTERFACE
2261 022054 016400 020516          MOV     SIDTAB(4),R0    ;GET SID VALUE
2262 022060 062700 000016          ADD     #TMT,R0        ;ADD LOOPBACK + BOOT DISABLE TO MNT. SID
2263 022064 063700 022202          ADD     MNTBTS,R0     ;ADD ANY EXTRA MNT BITS
2264 022070 027777 001476 001474  CMP     @ACR,@ACR      ;SHORT DELAY TO ALLOW RESET TO WORK
2265 022076 027777 001470 001466  CMP     @ACR,@ACR
2266 022104 012777 000015 001460  MOV     #MTC,@ACR     ;CHOOSE MTC
2267 022112 010077 001460          MOV     R0,@MTC      ;LOAD MTC
2268 022116 012777 000001 001446  MOV     #PGCSE,@ACR   ;SELECT PGCS
2269 022124 017737 001446 023610  MOV     @PGCS,DISPLY  ;READ PGCS
2270 022132 032737 004000 023610  BIT     #PRDY,DISPLY  ;CHECK FOR RDY
2271 022140 001003          BNE     1$           ;SKIP IF SET
2272 022142 104044          ERROR+ 4            ;DEVICE CLR DID NOT SET PGCS RDY
2273 022144 004737 022234          JSR     PC,DMPREG
2274 022150 012777 000005 001414 1$:  MOV     #PGFE,@ACR   ;SELECT PGF
2275 022156 012777 007400 001412  MOV     #7400,@PGF   ;CLEAR BOOT FLAGS
2276 022164 012777 007400 001404  MOV     #7400,@STF   ;CLEAR BOOT FLAGS
2277 022172 012777 000017 001376  MOV     #17,@DCF    ;CLEAR DCF FLAGS
2278 022200 000207          RTS     PC           ;EXIT
2279
2280 022202 000000          MNTBTS: 0
2281
2282          ;SET MAINTENENCE MODE SO THAT SANITY TIMER OPERATIONS ACT
2283          ;LIKE PROGRAM GENERATED OPERATIONS
2284
2285 022204 012737 004000 022202  SSTMNT: MOV     #MTYP,MNTBTS ;SET TYP TO A 1
2286 022212 000715          BR     SETMNT1     ;EXIT THROUGH SETMNT
2287
2288          ;PARITY CALCULATION SUBROUTINE
2289          ;COUNTS THE NUMBER OF 1 BITS IN R0 IN R1
2290
2291 022214 010046          PARCAL: MOV     R0,-(SP)
2292 022216 005001          CLR     R1
2293 022220 006300 1$:  ASL     R0
2294 022222 005501          ADC     R1
2295 022224 005700          TST     R0
2296 022226 001374          BNE     1$
2297 022230 012600          MOV     (SP)+,R0
2298 022232 000207          RTS     PC
    
```



```

2300          :DUMP ALL REGISTERS ROUTINE
2301
2302 022234 032777 000001 156676 DMPREG: BIT    #BIT0,@SWR    ;TEST BIT0
2303 022242 001001          BNE    1$          ;SKIP IF SET
2304 022244 000207          RTS    PC            ;EXIT IF 0
2305 022246 104401 033561          TYPE   ,HDRMES
2306 022252 012737 000011 022334  MOV   #9,TCR    ;DO 9 TIMES
2307 022260 017746 001306          MOV   @ACR,-(SP) ;:PUSH @ACR ON STACK
2308 022264 005077 001302          CLR   @ACR      ;:INIT ADDRESS
2309 022270
    (1) 022270 017746 001302          MOV   @ADR,-(SP) ;:SAVE @ADR FOR TYPEOUT
    (1) 022274 104402          TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2310 022276 104401 033671          TYPE   ,TWOSP
2311 022302 005337 022334          DEC   TCR
2312 022306 001370          BNE   2$          ;DONE 9?
2313 022310 104401 033710          TYPE   ,CARET
2314 022314 012677 001252          MOV   (SP)+,@ACR ;:POP STACK INTO @ACR
2315 022320 005777 156614          TST   @SWR      ;CHECK FOR ERR HLT
2316 022324 100002          BPL   3$          ;SKIP IF 0
2317 022326 000000          HALT
2318 022330 104407
2319 022332 000207          3$:   RTS    PC            ;EXIT
2320
2321 022334 000000          TCR:   0
2322
2323          :ROUTINE TO GET DEVICE PARAMETERS
2324
2325 022336          GETPAR:
    (1) 022336 104401 022344          TYPE   ,65$      ;:TYPE ASCIZ STRING
    (1) 022342 000444          BR     64$      ;:GET OVER THE ASCIZ
    (1) 022454          ;:65$: .ASCIZ <200>/IN THE FOLLOWING THREE PRINTOUTS, TYPE THE NEW DATA OR 0 IF NO CHA
    (1) 022454          64$:
2326 022454 104401 033465          10$:  TYPE   ,ASKDVA
2327 022460 013746 023572          MOV   ACR,-(SP) ;:SAVE ACR FOR TYPEOUT
    (1) 022464 104402          TYPOC          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
2328 022466 104401 033667          TYPE   ,SPACE
2329 022472 104412          RDOCT
2330 022474 012600          MOV   (SP)+,R0  ;GET VALUE TYPED IN.
2331 022476 001402          BEQ   11$      ;IF ZERO, DON'T CHANGE.
2332 022500 010037 023572          MOV   R0,ACR   ;STORE NEW VALUE
2333 022504 023727 023572 160000 11$:  CMP   ACR,#160000 ;CHECK FOR LEGALITY
2334 022512 103760          BLO   10$      ;LOOP IF BAD
2335 022514 032737 000003 023572  BIT   #3,ACR    ;CHECK FOR MOD. 4
2336 022522 001354          BNE   10$      ;LOOP IF BAD
2337 022524 013737 023572 023574  MOV   ACR,ACRH  ;LOAD OTHER ADDRESS SLOTS
2338 022532 005237 023574          INC   ACRH
2339 022536 013737 023574 023576  MOV   ACRH,ADR
2340 022544 005237 023576          INC   ADR
2341 022550 013737 023576 023600  MOV   ADR,ADRH
2342 022556 005237 023600          INC   ADRH
2343 022562 104401 033505          1$:   TYPE   ,ASKPIV
2344 022566 013746 023602          MOV   IIV,-(SP) ;:SAVE IIV FOR TYPEOUT
    (1) 022572 104403          ;:1
    (1) 022574 006          ;:GO TYPE--OCTAL ASCII
    (1) 022575 000          .BYTE 6      ;:TYPE 6 DIGITS
    (1) 022575 000          .BYTE 0      ;:SUPPRESS LEADING ZEROS
    
```

```

2345 022576 104401 033667      TYPE      ,SPACE
2346 022602 104412      RDOCT
2347 022604 012600      MOV      (SP)+,R0      ;GET VALUE TYPED IN.
2348 022606 001402      BEQ      12$          ;IF 0, DON'T CHANGE.
2349 022610 010037 023602      MOV      R0,IIIV      ;LOAD NEW VALUE
2350 022614 023727 023602 002000 12$:      CMP      IIIV,#2000   ;CHECK FOR LEGALITY
2351 022622 103357      BHIS     1$          ;LOOP IF BAD
2352 022624 032737 000003 023602      BIT      #3,IIIV     ;CHECK FOR MOD. 4
2353 022632 001353      BNE     1$          ;LOOP IF BAD
2354 022634 013737 023602 023604      MOV      IIIV,IIIP    ;LOAD PIL
2355 022642 062737 000002 023604      ADD      #2,IIIP     ;WITH IIIV+2
2356 022650 104401 033540      TYPE     ,ASKBRL
2357 022654 013746 023606      MOV      IIIL,-(SP)   ;;SAVE IIIL FOR TYPEOUT
(1)                                     ;;1
(1) 022660 104403      TYPOS
(1) 022662 006          ;GO TYPE--OCTAL ASCII
(1) 022663 000          ;TYPE 6 DIGITS
2358 022664 104401 033667      TYPE     ,SPACE
2359 022670 104412      RDOCT
2360 022672 012600      MOV      (SP)+,R0    ;GET VALUE TYPED IN.
2361 022674 001402      BEQ      21$          ;IF ZERO, DON'T CHANGE.
2362 022676 010037 023606      MOV      R0,IIIL     ;GET BR LEVEL
2363 022702 023727 023606 000004 21$:      CMP      IIIL,#4     ;CHECK FOR LEGALITY
2364 022710 103757      BLO     2$          ;LOOP IF BAD
2365 022712 023727 023606 000007      CMP      IIIL,#7     ;CHECK
2366 022720 101353      BHI     2$          ;LOOP IF BAD
2367 022722 000137 023346      JMP      3$          ;JUMP OVER HELP MSG.
2368 022726      30$:
(1) 022726 104401 022734      TYPE     ,67$        ;;TYPE ASCIZ STRING
(1) 022732 000422      BR      66$          ;;GET OVER THE ASCIZ
(1)                                     ;;67$: .ASCIZ <200>/HARDWARE CONFIGURATION CODES ARE:/
(1) 023000      66$:
2369 023000 104401 023006      TYPE     ,69$        ;;TYPE ASCIZ STRING
(1) 023004 000432      BR      68$          ;;GET OVER THE ASCIZ
(1)                                     ;;69$: .ASCIZ <CRLF>/ 0: MAINT. MODE ONLY;EITHER NO TERM, OR INHIBITS/
(1) 023072      68$:
2370 023072 104401 023100      TYPE     ,71$        ;;TYPE ASCIZ STRING
(1) 023076 000436      BR      70$          ;;GET OVER THE ASCIZ
(1)                                     ;;71$: .ASCIZ <CRLF>/ 1: IIST BUS TERMINATED, WITH NO INTERFERING INTERFACES/
(1) 023174      70$:
2371 023174 104401 023202      TYPE     ,73$        ;;TYPE ASCIZ STRING
(1) 023200 000427      BR      72$          ;;GET OVER THE ASCIZ
(1)                                     ;;73$: .ASCIZ <CRLF>/ 2: OTHER INTERFACES ONLINE WITH IDLE XMIT/
(1) 023260      72$:
2372 023260 104401 023266      TYPE     ,75$        ;;TYPE ASCIZ STRING
(1) 023264 000430      BR      74$          ;;GET OVER THE ASCIZ
(1)                                     ;;75$: .ASCIZ <CRLF>/ 3: OTHER ACTIVE NON-INHIBITED UNITS ONLINE/
(1) 023346      74$:
2373 023346      3$:
(1) 023346 104401 023354      TYPE     ,77$        ;;TYPE ASCIZ STRING
(1) 023352 000436      BR      76$          ;;GET OVER THE ASCIZ
(1)                                     ;;77$: .ASCIZ <200>/ENTER HARDWARE CONFIGURATION TYPE (0-3) OR 111 FOR HELP: /
(1) 023450      76$:
2374 023450 104412      RDOCT
2375 023452 012637 023660      MOV      (SP)+,CONFIG ;GET VALUE
2376 023456 023727 023660 000003      CMP      CONFIG,#3   ;SEE IF VALID

```

```

2377 023464 101402          BLOS      .+6
2378 023466 000137 022726  JMP      30$      ;IF NOT, DO HELP MSG.
2379 023472 000207          RTS      PC      ;EXIT
2380          ;ROUTINE TO REPORT THE SELF ID OF THE IIST
2381
2382 023474 005737 001236  GETSID: TST     $PASS      ;PASS 1?
2383 023500 001401          BEQ     1$          ;SKIP IF YES
2384 0235J2 000207          RTS     PC          ;EXIT IF NOT
2385 023504 005737 000042  1$:     TST     @#42      ;ARE WE IN ACT?
2386 023510 001002          BNE     2$          ;IF YES, DON'T PRINT
2387 023512 104401 033674          TYPE   ,DMPSID
2388 023516 012777 100000 000046  2$:     MOV     #CLR,@ACR   ;CLEAR INTERFACE
2389 023524 017700 000042          MOV     @ACR,R0      ;GET SID
2390 023530 000300          SWAB   R0           ;INTO B2-0
2391 023532 042700 177770          BIC     #177770,R0   ;CLEAR B15-3
2392 023536 010037 023570          MOV     R0,SIDNUM   ;SAVE SELF ID NUM.
2393 023542 005737 000042          TST     @#42      ;IN ACT
2394 023546 001007          BNE     3$          ;DON'T PRINT IF YES.
2395 023550 010046          MOV     RC,-(SP)    ;SAVE R0 FOR TYPEOUT
(1)
(1) 023552 104403          TYPOS   ;:1
(1) 023554 006          .BYTE  6           ;:GO TYPE--OCTAL ASCII
(1) 023555 000          .BYTE  0           ;:TYPE 6 DIGITS
2396 023556 104401 033710          TYPE   ,CARET      ;:SUPPRESS LEADING ZEROS
2397 023562 104401 0337'0          TYPE   ,CARET
2398 023566 000207          3$:     RTS     PC      ;EXIT
2399
2400 023570 000000          SIDNUM: 0
  
```

```

2402          .SBTTL  DEVICE ADDRESS PARAMETER BLOCK
2403
2404 023572 177500      ACR: 177500      ;ACCESS CONTROL REGISTER
2405 023574 177501      ACRH: 177501     ;HIGH BYTE OF ACR
2406
2407 023576          ADR:          ;ACCESS DATA REGISTER
2408 023576          PGTE:         ;PGM-GEN. XMIT ENABLES REGISTER
2409 023576          PGCS:         ;PGM-GEN. CSR
2410 023576          SITE:         ;SANITY TIMER XMIT ENABLES REGISTER
2411 023576          STCS:         ;SANITY TIMER CSR
2412 023576          IMSK:         ;INPUT MASKS REGISTER
2413 023576          PGF:          ;PGM-GEN. I/B FALAGS REGISTER
2414 023576          STF:          ;SANITY TIMER FLAGS REGISTER
2415 023576          DCF:          ;DCLO/DISCONNECT FLAGS REGISTER
2416 023576          EXC:          ;EXCEPTIONS REGISTER
2417 023576 177502      MTC: 177502     ;MAINTENANCE REGISTER
2418 023600          ADRH:         ;ADR HIGH BYTE
2419 023600          PGCSH:        ;PGCS HIGH BYTE
2420 023600          STCSH:        ;STCS HIGH BYTE
2421 023600          IMSKH:        ;IMSK HIGH BYTE
2422 023600          PGFH:         ;PGF HIGH BYTE
2423 023600          DCFH:         ;DCF HIGH BYTE
2424 023600          EXCH:         ;EXC HIGH BYTE
2425 023600 177503      MTC: 177503     ;MTC HIGH BYTE
2426 023602 000260      IIIV: 260      ;IIST INTERRUPT VECTOR ADDRESS
2427 023604 000262      IIIP: 262      ;IIST INTERRUPT PRIORITY ADDRESS
2428 023606 000006      IIIL: 6       ;IIST INTERRUPT LEVEL
2429
2430
2431          ;CONSTANTS AND VARIABLES
2432
2433 023610 000000      DISPLY: 0
2434 023612 000000      TIMPER: 0
2435 023614 000000      LINMBR: 0
2436 023616 000020      SNAPO: .BLKW 16.
2437
2438 023656 000000      PARFLG: .WORD 0
2439
2440 023660 000000      CONFIG: .WORD 0
2441
2442 023662 000000      STCCNT: .WORD 0      ;COUNT OF # PG CHARACTERS TRANSMITTED DURING
2443                                     ;TWO COUNTS OF THE SANITY TIMER.
2444 023664 000000      MINCNT: .WORD 0      ;MINIMUM ALLOWED VALUE OF STCCNT.
2445 023666 000000      MAXCNT: .WORD 0      ;MAXIMUM ALLOWED VALUE OF STCCNT.
  
```

2447

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

.SBTTL END OF PASS ROUTINE

*INCREMENT THE PASS NUMBER (\$PASS)
*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO TST1

(1) 023670
(1) 023670
(1) 023672
(1) 023676
(1) 023702
(1) 023706
(1) 023714
(1) 023716
(1) 023720
(1) 023722
(1) 023724
(1) 023726
(1) 023730
(2) 023734
(2) 023740
(1) 023742
(1) 023746
(1) 023752
(1) 023754
(1) 023756
(1) 023760
(1) 023762
(1) 023764
(1) 023766
(1) 023766
(1) 023770
(1) 023772
(1) 023775
(1) 024002
(1) 024010

000004
005037 001102
005037 001214
005237 001236
042737 100000 001236
005327
000001
003022
012737
000001
023716
104401 023775
013746 001236
104405
104401 023772
013700 000042
001405
000005
004710
000240
000240
000240
000137
002524
377 377 000
015 042412 042116
051501 020123
000043

\$EOP:
SCOPE
CLR \$TSTNM ;;ZERO THE TEST NUMBER
CLR \$TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC \$PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,\$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
\$FOPCT: .WORD 1
BGT \$DOAGN ;;YES
MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
\$ENDCT: .WORD 1
\$EOPCT
TYPE \$ENDMG ;;TYPE 'END PASS #'
MOV \$PASS,-(SP) ;;SAVE \$PASS FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE \$ENULL ;;TYPE A NULL CHARACTER
\$GET42: MOV @42,R0 ;;GET MONITOR ADDRESS
BEQ \$DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
\$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
\$DOAGN: JMP @(PC)+ ;;RETURN
\$RTNAD: .WORD TST1
\$FNUL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
\$ENDMG: .ASCIIZ <15><12>/END PASS #/

2448

2449

(1)
(2)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

.SBTTL SCOPE HANDLER ROUTINE

*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
*AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
*AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW14=1 LOOP ON TEST
*SW11=1 INHIBIT ITERATIONS
*SW09=1 LOOP ON ERROR
*CALL
* SCOPE ;;SCOPE-IOT
\$SCOPE:
CKSWR ;;TEST FOR CHANGE IN SOFT-SW
;;GO TO ERROR ROUTINE IF RETURN PC LESS THAN 1002

```

(1)                ;:OTHERWISE CONTINUE
(1) 024014 021627 001002      CMP      (SP),#1002      ;:UNEXPECTED TRAP OR INTERRUPT
(1) 024020 101002              BHI      1$              ;:ARE TRAPPED HERE VIA IOT
(1) 024022 000137 024264      JMP      $ERROR          ;:GO PROCESS UNEXPECTED TRAP
(1) 024026 032777 040000 155104 1$: BIT      #BIT14,@SWR      ;:LOOP ON PRESENT TEST?
(1) 024034 001104              BNE      $OVER          ;:YES IF SW14=1
(1)                ;:*****START OF CODE FOR THE XOR TESTER*****
(1) 024036 000416      $XTSTR: BR      6$      ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
(1)                ;:THIS INSTRUCTION TO A 'NOP' (NOP=240)
(1) 024040 013746 000004      MOV      @WERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
(1) 024044 012737 024064 000004      MOV      #5,$@WERRVEC ;:SET FOR TIMEOUT
(1) 024052 005737 177060      TST      @177060        ;:TIME OUT ON XOR?
(1) 024056 012637 000004      MOV      (SP)+,@WERRVEC ;:RESTORE THE ERROR VECTOR
(1) 024062 000453              BR      $SVLAD          ;:GO TO THE NEXT TEST
(1) 024064 022626      5$: CMP      (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
(1) 024066 012637 000004      MOV      (SP)+,@WERRVEC ;:RESTORE THE ERROR VECTOR
(1) 024072 000413              BR      7$              ;:LOOP ON THE PRESENT TEST
(1) 024074                6$: ;:*****END OF CODE FOR THE XOR TESTER*****
(1) 024074 105737 001103      2$: TSTB     $ERFLG        ;:HAS AN ERROR OCCURRED?
(1) 024100 001421              BEQ      3$              ;:BR IF NO
(1) 024102 123737 001115 001103      CMPSB   $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
(1) 024110 101015              BHI      3$              ;:BR IF NO
(1) 024112 032777 001000 155020      BIT      #BIT09,@SWR    ;:LOOP ON ERROR?
(1) 024120 001404              BEQ      4$              ;:BR IF NO
(1) 024122 013737 001110 001106 7$: MOV      $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
(1) 024130 000446              BR      $OVER          ;:
(1) 024132 105037 001103      4$: CLRB     $ERFLG        ;:ZERO THE ERROR FLAG
(1) 024136 005037 001214      CLR      $TIMES        ;:CLEAR THE NUMBER OF ITERATIONS TO MA
(1) 024142 000415              BR      1$              ;:ESCAPE TO THE NEXT TEST
(1) 024144 032777 004000 154766 3$: BIT      #BIT11,@SWR    ;:INHIBIT ITERATIONS?
(1) 024152 001011              BNE      1$              ;:BR IF YES
(1) 024154 005737 001236      TST      $PASS         ;:IF FIRST PASS OF PROGRAM
(1) 024160 001406              BEQ      1$              ;:INHIBIT ITERATIONS
(1) 024162 005237 001104      INC      $ICNT         ;:INCREMENT ITERATION COUNT
(1) 024166 023737 001214 001104      CMP      $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
(1) 024174 002024              BGE      $OVER          ;:BR IF MORE ITERATION REQUIRED
(1) 024176 012737 000001 001104 1$: MOV      #1,$ICNT      ;:REINITIALIZE THE ITERATION COUNTER
(1) 024204 013737 024262 001214      MOV      $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
(1) 024212 105237 001102      $SVLAD: INCB    $STSTNM ;:COUNT TEST NUMBERS
(1) 024216 113737 001102 001234      MOVB    $STSTNM,$STSTN ;:SET TEST NUMBER IN APT MAILBOX
(1) 024224 011637 001106      MOV      (SP),$LPADR   ;:SAVE SCOPE LOOP ADDRESS
(1) 024230 011637 001110      MOV      (SP),$LPERR   ;:SAVE ERROR LOOP ADDRESS
(1) 024234 005037 001216      CLR      $ESCAPE       ;:CLEAR THE ESCAPE FROM ERROR ADDRESS
(1) 024240 112737 000001 001115      MOVB    #1,$ERMAX      ;:ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1) 024246 013777 001102 154666 $OVER: MOV      $STSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
(1) 024254 013716 001106      MOV      $LPADR,(SP)   ;:FUDGE RETURN ADDRESS
(1) 024260 000002              RTI                    ;:FIXES PS
(1) 024262 000001      $MXCNT: 1              ;:MAX. NUMBER OF ITERATIONS
2450                .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 $ERRIYP ON ERROR
(1)                ;:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(1)                ;:*SW15: 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) $ERROR:
(1) 024264      104407
(1) 024264 104407 105237 001103 7$: CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
(1) 024266 105237 001103 154640 INCB $ERFLG   ;;SET THE ERROR FLAG
(1) 024272 001775 001102 154630 BEQ 7$      ;;DON'T LET THE FLAG GO TO ZERO
(1) 024274 013777 002000 154630 MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 024302 032777 001112 154630 BIT #BIT10,@SWR   ;;BELL ON ERROR?
(1) 024310 001402 001220 154630 BEQ 1$      ;;NO - SKIP
(1) 024312 104401 001112 154630 TYPE $BELL    ;;RING BELL
(1) 024316 005237 001112 154630 $: INC $ERTTL  ;;COUNT THE NUMBER OF ERRORS
(1) 024322 011637 001116 154630 MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
(1) 024326 162737 000002 001116 SUB #2, $ERRPC
(1) 024334 117737 001114 154556 MOVB @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
(1) 024342 032777 020000 154570 BIT #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
(1) 024350 001055 001002 154570 BNE 20$      ;;SKIP TYPEOUTS
(1) 024352 021627 001002 154570 CMP (SP), #1002  ;;IF RETURN PC LESS THAN 1002
(1) 024356 101046 001002 154570 BHI 12$      ;;ERROR IS ILLEGAL TRAP
(1) ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
(1) 024360 016637 000004 001116 MOV 4(SP), $ERRPC ;;GET PC AT TIME OF FALSE TRAP
(1) 024366 162737 000002 001116 SUB #2, $ERRPC   ;;ADJUST PC
(1) 024374 104401 024440 001116 TYPE 10$      ;;TYPE HEADER
(2) 024400 013746 001116 001116 MOV $ERRPC, -(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 024404 104402 001116 001116 TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 024406 104401 024446 001116 TYPE 11$
(1) 024412 162716 000004 001116 SUB #4, (SP)    ;;GET FALSE TRAP VECTOR ADDR
(1) 024416 011637 001116 001116 MOV (SP), $ERRPC
(2) 024422 013746 001116 001116 MOV $ERRPC, -(SP) ;;SAVE $ERRPC FOR TYPEOUT
(2) 024426 104402 001116 001116 TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
(1) 024430 104401 001225 001116 TYPE $CRLF
(1) 024434 022626 001225 001116 CMP (SP)+, (SP)+ ;;POP FALSE TRAP VECTOR PC&ADDR
(1) 024436 000422 001225 001116 BR 20$
(1) 024440 050200 036503 000040 10$: .ASCIZ <200>'PC- '
(1) 024446 020040 047125 054105 11$: .ASCIZ ' UNEXPECTED TRAP TO '
(1) 024454 042520 052103 042105
(1) 024462 052040 040522 020120
(1) 024470 047524 000040
(1) .EVEN
(1) 024474 004737 024606 001225 12$: JSR PC, $ERRTYP ;;GO TO USER ERROR ROUTINE
(1) 024500 104401 001225 001225 TYPE $CRLF
(1) 024504 122737 000001 001250 20$: CMPB #APTENV, $ENV ;;RUNNING IN APT MODE
(1) 024512 001007 000001 001250 BNE 2$      ;;NO, SKIP APT ERROR REPORT
(1) 024514 113737 001114 024526 MOVB $ITEMB, 21$ ;;SET ITEM NUMBER AS ERROR NUMBER
(1) 024522 004737 027550 024526 JSR PC, $ATY4   ;;REPORT FATAL ERROR TO APT
(1) 024526 000 027550 024526 21$: .BYTE 0
(1) 024527 000 027550 024526 .BYTE 0
(1) 024530 000777 027550 024526 22$: BR 22$      ;;APT ERROR LOOP
(1) 024532 005777 154402 027550 2$: TST @SWR    ;;HALT ON ERROR
(1) 024536 100002 027550 024536 BPL 3$      ;;SKIP IF CONTINUE
(1) 024540 000000 027550 024536 HALT      ;;HALT ON ERROR

```



```

(1) 026374 001405      BEQ      62$      ;;NO,GO CHECK FOR CONSOLE
(1) 026376 010037 026406  MOV      RO,61$  ;;SETUP MESSAGE ADDRESS FOR APT
(1) 026402 004737 027540  JSR      PC,$ATY3 ;;SPOOL MESSAGE TO APT
(1) 026406 000000      .WORD    0        ;;MESSAGE ADDRESS
(1) 026410 132737 000040 001251 62$:     BITB     #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 026416 001003      BNE      60$      ;;YES,SKIP TYPE OUT
(1) 026420 112046      2$:     MOVB    (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 026422 001005      BNE      4$       ;;BR IF IT ISN'T THE TERMINATOR
(1) 026424 005726      TST     (SP)+     ;;IF TERMINATOR POP IT OFF THE STACK
(1) 026426 012600      60$:    MOV     (SP)+,RO  ;;RESTORE RO
(1) 026430 062716 000002  3$:     ADD     #2,(SP)  ;;ADJUST RETURN PC
(1) 026434 000002      RTI      ;;RETURN
(1) 026436 122716 000011  4$:     CMPB    #HT,(SP)  ;;BRANCH IF <HT>
(1) 026442 001430      BEQ      8$       ;;BRANCH IF NOT <CRLF>
(1) 026444 122716 000200  CMPB    #CRLF,(SP)
(1) 026450 001006      BNE      5$       ;;POP <CR><LF> EQUIV
(1) 026452 005726      TST     (SP)+     ;;TYPE A CR AND LF
(1) 026454 104401      TYPE
(1) 026456 001225      $CRLF
(1) 026460 105037 026614  CLRB    $CHARCNT  ;;CLEAR CHARACTER COUNT
(1) 026464 000755      BR      2$       ;;GET NEXT CHARACTER
(1) 026466 004737 026550  5$:     JSR     PC,$TYPEC  ;;GO TYPE THIS CHARACTER
(1) 026472 123726 001156  6$:     CMPB    $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 026476 001350      BNE      2$       ;;IF NO GO GET NEXT CHAR.
(1) 026500 013746 001154  MOV     $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(1)                                ;;AND THE NULL CHAR.
(1) 026504 105366 000001  7$:     DECB    1(SP)    ;;DOES A NULL NEED TO BE TYPED?
(1) 026510 002770      BLT     6$       ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 026512 004737 026550      JSR     PC,$TYPEC  ;;GO TYPE A NULL
(1) 026516 105337 026614      DECB    $CHARCNT  ;;DO NOT COUNT AS A COUNT
(1) 026522 000770      BR      7$       ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

(1) 026524 112716 000040  8$:     MOVB    #' ,(SP)  ;;REPLACE TAB WITH SPACE
(1) 026530 004737 026550  9$:     JSR     PC,$TYPEC  ;;TYPE A SPACE
(1) 026534 132737 000007 026614  BITB    #7,$CHARCNT ;;BRANCH IF NOT AT
(1) 026542 001372      BNE      9$       ;;TAB STOP
(1) 026544 005726      TST     (SP)+     ;;POP SPACE OFF STACK
(1) 026546 000724      BR      2$       ;;GET NEXT CHARACTER
(1) 026550 105777 152374  $TYPEC: TSTB    @STPS  ;;WAIT UNTIL PRINTER IS READY
(1) 026554 100375      BPL     $TYPEC
(1) 026556 116677 000002 152366  MOVB    2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 026564 122766 000015 000002  CMPB    #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
(1) 026572 001003      BNE      1$       ;;BRANCH IF NO
(1) 026574 105037 026614  CLRB    $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
(1) 026600 000406      BR      $TYPEX   ;;EXIT
(1) 026602 122766 000012 000002 1$:     CMPB    #LF,2(SP) ;;IS CHARACTER A LINE FEED?
(1) 026610 001402      BEQ     $TYPEX   ;;BRANCH IF YES
(1) 026612 105227      INCB    (PC)+     ;;COUNT THE CHARACTER
(1) 026614 000000      $CHARCNT: .WORD  0  ;;CHARACTER COUNT STORAGE
(1) 026616 000207      $TYPEX: RTS     PC

```

2456
(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) 026620 017646 000000 $TYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
(1) 026624 116637 000001 027043 MOVVB  1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
(1) 026632 112637 027045 MOVVB  (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
(1) 026636 062716 000002 ADD     #2,(SP)          ;;ADJUST RETURN ADDRESS
(1) 026642 000406 BR      $TYPON
(1) 026644 112737 000001 027043 $TYPOC: MOVVB  #1, $OFILL      ;;SET THE ZERO FILL SWITCH
(1) 026652 112737 000006 027045 MOVVB  #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
(1) 026660 112737 000005 027042 $TYPON: MOVVB  #5, $OCNT      ;;SET THE ITERATION COUNT
(1) 026666 010346 MOV     R3,-(SP)          ;;SAVE R3
(1) 026670 010446 MOV     R4,-(SP)          ;;SAVE R4
(1) 026672 010546 MOV     R5,-(SP)          ;;SAVE R5
(1) 026674 113704 027045 MOVVB  $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 026700 005404 NEG     R4
(1) 026702 062704 000006 ADD     #6,R4            ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 026706 110437 027044 MOVVB  R4, $OMODE        ;;SAVE IT FOR USE
(1) 026712 113704 027043 MOVVB  $OFILL,R4        ;;GET THE ZERO FILL SWITCH
(1) 026716 016605 000012 MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1) 026722 005003 CLR     R3              ;;CLEAR THE OUTPUT WORD
(1) 026724 006105 1$: ROL    R5              ;;ROTATE MSB INTO 'C'
(1) 026726 000404 BR      3$            ;;GO DO MSB
(1) 026730 006105 2$: ROL    R5              ;;FORM THIS DIGIT
(1) 026732 006105 ROL    R5
(1) 026734 006105 ROL    R5
(1) 026736 010503 MOV     R5,R3
(1) 026740 006103 3$: ROL    R3              ;;GET LSB OF THIS DIGIT
(1) 026742 105337 027044 DECB   $OMODE          ;;TYPE THIS DIGIT?
(1) 026746 100016 BPL    7$              ;;BR IF NO
(1) 026750 042703 177770 BIC    #177770,R3     ;;GET RID OF JUNK
(1) 026754 001002 BNE    4$              ;;TEST FOR 0
(1) 026756 005704 TST    R4              ;;SUPPRESS THIS 0?
(1) 026760 001403 BFQ    5$              ;;BR IF YES
(1) 026762 005204 4$: INC    R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1) 026764 052703 000060 BIS    #'C,R3         ;;MAKE THIS DIGIT ASCII
(1) 026770 052703 000040 5$: BIS    #' ,R3      ;;MAKE ASCII IF NOT ALREADY
```



```

(1) 027146 103003          BCC      6$          ;;BR IF NO
(1) 027150 116663 000001 177777  MOVB    1(SP),-1(R3) ;;YES--SET THE SIGN
(1) 027156 052702 000060          BIS     #'0,R2      ;;MAKE THE BCD DIGIT ASCII
(1) 027162 052702 000040          BIS     #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1) 027166 110223          MOVB    R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1) 027170 005720          TST     (R0)+       ;;JUST INCREMENTING
(1) 027172 020027 000010          CMP     R0,#10     ;;CHECK THE TABLE INDEX
(1) 027176 002746          BLT     2$          ;;GO DO THE NEXT DIGIT
(1) 027200 003002          BGT     8$          ;;GO TO EXIT
(1) 027202 010502          MOV     R5,R2      ;;GET THE LSD
(1) 027204 000764          BR      6$          ;;GO CHANGE TO ASCII
(1) 027206 105726          8$: TSTB    (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
(1) 027210 100003          BPL     9$          ;;BR IF NO
(1) 027212 116663 177777 177776  MOVB    -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
(1) 027220 100013          9$: CLRB    (R3)     ;;SET THE TERMINATOR
(3) 027222 012605          MOV     (SP)+,R5   ;;POP STACK INTO R5
(3) 027224 012603          MOV     (SP)+,R3   ;;POP STACK INTO R3
(3) 027226 012602          MOV     (SP)+,R2   ;;POP STACK INTO R2
(3) 027230 012601          MOV     (SP)+,R1   ;;POP STACK INTO R1
(3) 027232 012600          MOV     (SP)+,R0   ;;POP STACK INTO R0
(1) 027234 104401 027262          TYPE   $DBLK      ;;NOW TYPE THE NUMBER
(1) 027240 016666 000002 000004  MOV     2(SP),4(SP) ;;ADJUST THE STACK
(1) 027246 012616          MOV     (SP)+,(SP)
(1) 027250 000002          RTI                    ;;RETURN TO USER
(1) 027252 023420          $DTBL: 10000.
(1) 027254 001750          1000.
(1) 027256 000144          100.
(1) 027260 000012          10.
(1) 027262 000004          $LBLK: .BLKW 4
2458 .SBTTL TRAP DECODER

```

```

(1)
(2)
(1) *****
(1) *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
(1) *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1) *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1) *GO TO THAT ROUTINE.

```

```

(1) 027272 010046          $TRAP: MOV     R0,-(SP) ;;SAVE R0
(1) 027274 016600 000002          MOV     2(SP),R0    ;;GET TRAP ADDRESS
(1) 027300 005740          TST     -(R0)       ;;BACKUP BY 2
(1) 027302 111000          MOVB    (R0),R0     ;;GET RIGHT BYTE OF TRAP
(1) 027304 006300          ASL     R0          ;;POSITION FOR INDEXING
(1) 027306 016000 027326          MOV     $TRPAD(R0),R0 ;;INDEX TO TABLE
(1) 027312 000200          RTS     R0          ;;GO TO ROUTINE

```

```

(1)
(1) ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
(1) 027314 011646          $TRAP2: MOV    (SP),-(SP) ;;MOVE THE PC DOWN
(1) 027316 016666 000004 000002          MOV    4(SP),2(SP) ;;MOVE THE PSW DOWN
(1) 027324 000002          RTI                    ;;RESTORE THE PSW

```

```

(1)
(2) .SBTTL TRAP TABLE
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
(16)
(17)
(18)
(19)
(20)
(21)
(22)
(23)
(24)
(25)
(26)
(27)
(28)
(29)
(30)
(31)
(32)
(33)
(34)
(35)
(36)
(37)
(38)
(39)
(40)
(41)
(42)
(43)
(44)
(45)
(46)
(47)
(48)
(49)
(50)
(51)
(52)
(53)
(54)
(55)
(56)
(57)
(58)
(59)
(60)
(61)
(62)
(63)
(64)
(65)
(66)
(67)
(68)
(69)
(70)
(71)
(72)
(73)
(74)
(75)
(76)
(77)
(78)
(79)
(80)
(81)
(82)
(83)
(84)
(85)
(86)
(87)
(88)
(89)
(90)
(91)
(92)
(93)
(94)
(95)
(96)
(97)
(98)
(99)
(100)

```

```

(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
(14)
(15)
(16)
(17)
(18)
(19)
(20)
(21)
(22)
(23)
(24)
(25)
(26)
(27)
(28)
(29)
(30)
(31)
(32)
(33)
(34)
(35)
(36)
(37)
(38)
(39)
(40)
(41)
(42)
(43)
(44)
(45)
(46)
(47)
(48)
(49)
(50)
(51)
(52)
(53)
(54)
(55)
(56)
(57)
(58)
(59)
(60)
(61)
(62)
(63)
(64)
(65)
(66)
(67)
(68)
(69)
(70)
(71)
(72)
(73)
(74)
(75)
(76)
(77)
(78)
(79)
(80)
(81)
(82)
(83)
(84)
(85)
(86)
(87)
(88)
(89)
(90)
(91)
(92)
(93)
(94)
(95)
(96)
(97)
(98)
(99)
(100)

```

```

(3)
(3)
(3)
(3) 027326 027314
(3) 027330 026336
(3) 027332 026644
(3) 027334 026620
(3) 027336 026660
(3) 027340 027046
(1)
(3) 027342 025500
(1)
(3) 027344 025430
(3) 027346 025712
(3) 027350 026032
(3) 027352 025270
2459
(1)
(2)
(1)
(1) 027354 012737 027514 000024
(1) 027362 012737 000340 000026
(3) 027370 010046
(3) 027372 010146
(3) 027374 010246
(3) 027376 010346
(3) 027400 010446
(3) 027402 010546
(3) 027404 017746 151530
(1) 027410 010637 027520
(1) 027414 012737 027426 000024
(1) 027422 000000
(1) 027424 000776
(1)
(2)
(1)
(1) 027426 012737 027514 000024
(1) 027434 013706 027520
(1) 027440 005037 027520
(1) 027444 005237 027520
(1) 027450 001375
(3) 027452 012677 151462
(3) 027456 012605
(3) 027460 012604
(3) 027462 012603
(3) 027464 012602
(3) 027466 012601
(3) 027470 012600
(1) 027472 012737 027354 000024
(1) 027500 012737 000340 000026
(1) 027506 104401
(1) 027510 027522
(1) 027512 000002
(1) 027514 000000
(1) 027516 000776
(1) 027520 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 # $ILLUP, @PWRVEC ::SET FOR FAST UP
MOV #340, @PWRVEC+2 ::PRIO:7
MOV R0, -(SP) ::PUSH R0 ON STACK
MOV R1, -(SP) ::PUSH R1 ON STACK
MOV R2, -(SP) ::PUSH R2 ON STACK
MOV R3, -(SP) ::PUSH R3 ON STACK
MOV R4, -(SP) ::PUSH R4 ON STACK
MOV R5, -(SP) ::PUSH R5 ON STACK
MOV @SWR, -(SP) ::PUSH @SWR ON STACK
MOV SP, $SAVR6 ::SAVE SP
MOV # $PWRUP, @PWRVEC ::SET UP VECTOR
HALT
BR .-2 ::HANG UP

:*****
:POWER UP ROUTINE
$PWRUP: MOV # $ILLUP, @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 # $PWRDN, @PWRVEC ::SET UP THE POWER DOWN VECTOR
MOV #340, @PWRVEC+2 ::PRIO:7
TYPE $POWER ::REPORT THE POWER FAILURE
$PWRMG: .WORD $POWER ::POWER FAIL MESSAGE POINTER
RTI
$ILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
BR .-2 ::BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: C ::PUT THE SP HERE
  
```

```
(1) 027522 005015 047520 042527 $POWER: .ASCIZ <15><12>'POWER'  
(1) 027530 000122  
(1)  
2460 .EVEN  
(1) .SBTTL APT COMMUNICATIONS ROUTINE  
(2) :*****  
(1) 027532 112737 000001 027776 $ATY1: MCVB #1,$FFLG ;;TO REPORT FATAL ERROR  
(1) 027540 112737 000001 027774 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE  
(1) 027546 000403 BR $ATYC  
(1) 027550 112737 000001 027776 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR  
(1) 027556 $ATYC:  
(3) 027556 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK  
(3) 027560 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK  
(1) 027562 105737 027774 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?  
(1) 027566 001450 BEQ 5$ ;;IF NOT: BR  
(1) 027570 122737 000001 001250 (MPB #APTENV,$ENV ;;OPERATING UNDER APT?  
(1) 027576 001031 BNE 3$ ;;IF NOT: BR  
(1) 027600 132737 000100 001251 BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?  
(1) 027606 001425 BEQ 3$ ;;IF NOT: BR  
(1) 027610 017600 000004 MOV @4(SP),R0 ;;GET MESSAGE ADDR.  
(1) 027614 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.  
(1) 027622 005737 001230 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?  
(1) 027626 001375 BNE 1$ ;;IF NOT: WAIT  
(1) 027630 010037 001244 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX  
(1) 027634 105720 2$: TSTB (R0)+ ;;FIND END OF MESSAGE  
(1) 027636 001376 BNE 2$  
(1) 027640 163700 001244 SUB $MSGAD,R0 ;;SUB START OF MESSAGE  
(1) 027644 006200 ASR R0 ;;GET MESSAGE LNGTH IN WORDS  
(1) 027646 010037 001246 MOV R0,$MSG LGT ;;PUT LENGTH IN MAILBOX  
(1) 027652 012737 000004 001230 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.  
(1) 027660 000413 BR 5$  
(1) 027662 017637 000004 027706 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE  
(1) 027670 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS  
(3) 027676 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK  
(1) 027702 004737 026336 JSR PC,$TYPE ;;CALL TYPE MACRO  
(1) 027706 000000 4$: .WORD 0  
(1) 027710 5$:  
(1) 027710 105737 027776 10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?  
(1) 027714 001416 BEQ 12$ ;;IF NOT: BR  
(1) 027716 005737 001250 TST $ENV ;;RUNNING UNDER APT?  
(1) 027722 001413 BEQ 12$ ;;IF NOT: BR  
(1) 027724 005737 001230 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?  
(1) 027730 001375 BNE 11$ ;;IF NOT: WAIT  
(1) 027732 017637 000004 001232 MOV @4(SP),$FATAL ;;GET ERROR #  
(1) 027740 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.  
(1) 027746 005237 001230 INC $MSGTYPE ;;TELL APT TO TAKE ERROR  
(1) 027752 105037 027776 12$: CLRB $FFLG ;;CLEAR FATAL FLAG  
(1) 027756 105037 027775 CLRB $LFLG ;;CLEAR LOG FLAG  
(1) 027762 105037 027774 CLRB $MFLG ;;CLEAR MESSAGE FLAG  
(3) 027766 012601 MOV (SP)+,R1 ;;POP STACK INTO R1  
(3) 027770 012600 MOV (SP)+,R0 ;;POP STACK INTO R0  
(1) 027772 000207 RTS PC ;;RETURN  
(1) 027774 000 $MFLG: .BYTE 0 ;;MESSG. FLAG  
(1) 027775 000 $LFLG: .BYTE 0 ;;LOG FLAG  
(1) 027776 000 $FFLG: .BYTE 0 ;;FATAL FLAG  
(1) 030000 .EVEN
```

MAINDEC-11-CRIIA
CRIIAA.P11

30-NOV-78

MACY11 30A(1052)

30-NOV-78 11:06

PAGE 59-16

J 8

APT COMMUNICATIONS ROUTINE

SEQ 0100

(1) 000200
(1) 000001
(1) 000100
(1) 000040

APTSIZE=200
APTENV=001
APTSPOOL=100
APTSUP=040

```

2462                                     .SBTTL  MESSAGES
2463
2464 030000 041501 020122 044502 EM1:  .ASCIZ      /ACR BIT3-0 LOAD-READ ERROR/
      030006 031524 030055 046040
      030014 040517 026504 042522
      030022 042101 042440 051122
      030030 051117      000
2465 030033      101 051103 041040 EM2:  .ASCIZ      /ACR BIT3-0 WERE NOT CLEARED BY RESET (ACR-CLR)/
      030040 052111 026463 020060
      030046 042527 042522 047040
      030054 052117 041440 042514
      030062 051101 042105 041040
      030070 020131 042522 042523
      030076 020124 040450 051103
      030104 041455 051114 000051
2466 030112 042511 024040 043520 EM7:  .ASCIZ      /IE (PGTE BIT2) CANNOT BE SET/
      030120 042524 041040 052111
      030126 024462 041440 047101
      030134 047516 020124 042502
      030142 051440 052105      000
2467 030147      111 020105 050050 EM10: .ASCIZ     /IF (PGTE BIT2) CANNOT BE WRITTEN TO 0/
      030154 052107 020105 044502
      030162 031124 020051 040503
      030170 047116 052117 041040
      030176 020105 051127 052111
      030204 042524 020116 047524
      030212 030040      000
2468 030215      111 020105 050050 EM11: .ASCIZ     /IE (PGTE BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)/
      030222 052107 020105 044502
      030230 031124 020051 040503
      030236 047116 052117 041040
      030244 020105 046103 040505
      030252 042522 020104 054502
      030260 051040 051505 052105
      030266 024040 041501 020122
      030274 046103 024522      000
2469 030301      120 050124 024040 EM12: .ASCIZ     /PTP (PGTE BIT1) CANNOT BE SET/
      030306 043520 042524 041040
      030314 052111 024461 041440
      030322 047101 047516 020124
      030330 042502 051440 052105
      030336      000
2470 030337      120 050124 024040 EM13: .ASCIZ     /PTP (PGTE BIT1) CANNOT BE WRITTEN TO 0/
      030344 043520 042524 041040
      030352 052111 024461 041440
      030360 047101 047516 020124
      030366 042502 053440 044522
      030374 052124 047105 052040
      030402 020117 000060
2471 030406 052120 020120 050050 EM14: .ASCIZ     /PTP (PGTE BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)/
      030414 052107 020105 044502
      030422 030524 020051 040503
      030430 047116 052117 041040
      030436 020105 046103 040505
      030444 042522 020104 054502
      030452 051040 051505 052105

```

	030460	024040	041501	020122			
	030466	046103	024522	000			
2472	030473	120	052107	020105	EM3:	.ASCIZ	/PGTE BIT11-8 LOAD-READ ERROR/
	030500	044502	030524	026461			
	030506	020070	047514	042101			
	030514	051055	040505	020104			
	030522	051105	047522	000122			
2473	030530	043520	042524	041040	EM4:	.ASCIZ	/PGTE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	030536	052111	030461	034055			
	030544	053440	051105	020105			
	030552	047516	020124	046103			
	030560	040505	042522	020104			
	030566	054502	051040	051505			
	030574	052105	024040	041501			
	030602	020122	046103	024522			
	030610	000					
2474	030611	120	052107	020105	EM5:	.ASCIZ	/PGTE BIT3-0 LOAD-READ ERROR/
	030616	044502	031524	030055			
	030624	046040	040517	026504			
	030632	042522	042101	042440			
	030640	051122	051117	000			
2475	030645	120	052107	020105	EM6:	.ASCIZ	/PGTE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	030652	044502	031524	030055			
	030660	053440	051105	020105			
	030666	047516	020124	046103			
	030674	040505	042522	020104			
	030702	054502	051040	051505			
	030710	052105	024040	041501			
	030716	020122	046103	024522			
	030724	000					
2476	030725	123	052124	020105	EM15:	.ASCIZ	/SITE BIT11-8 LOAD-READ ERROR/
	030732	044502	030524	026461			
	030740	020070	047514	042101			
	030746	051055	040505	020104			
	030754	051105	047522	000122			
2477	030762	052123	042524	041040	EM16:	.ASCIZ	/SITE BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	030770	052111	030461	034055			
	030776	053440	051105	020105			
	031004	047516	020124	046103			
	031012	040505	042522	020104			
	031020	054502	051040	051505			
	031026	052105	024040	041501			
	031034	020122	046103	024522			
	031042	000					
2478	031043	123	052124	020105	EM17:	.ASCIZ	/SITE BIT3-0 LOAD-READ ERROR/
	031050	044502	031524	030055			
	031056	046040	040517	026504			
	031064	042522	042101	042440			
	031072	051122	051117	000			
2479	031077	123	052124	020105	EM20:	.ASCIZ	/SITE BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	031104	044502	031524	030055			
	031112	053440	051105	020105			
	031120	047516	020124	046103			
	031126	040505	042522	020104			
	031134	054502	051040	051505			
	031142	052105	024040	041501			

	031150	020122	046103	024522		
2480	031156	000			EM21:	.ASCIZ /LKE (STCS BIT2) CANNOT BE SET/
	031157	114	042513	024040		
	031164	052123	051503	041040		
	031172	052111	024462	041440		
	031200	047101	047516	020124		
	031206	042502	051440	052105		
2481	031214	000			EM22:	.ASCIZ /LKE (STCS BIT2) CANNOT BE WRITTEN TO 0/
	031215	114	042513	024040		
	031222	052123	051503	041040		
	031230	052111	024462	041440		
	031236	047101	047516	020124		
	031244	042502	053440	044522		
	031252	052124	047105	052040		
2482	031260	020117	000060		EM23:	.ASCIZ /LKE (STCS BIT2) CANNOT BE CLEARED BY RESET (ACR CLR)/
	031264	045514	020105	051450		
	031272	041524	020123	044502		
	031300	031124	020051	040503		
	031306	047116	052117	041040		
	031314	020105	046103	040505		
	031322	042522	020104	054502		
	031330	051040	051505	052105		
	031336	024040	041501	020122		
2483	031344	046103	024522	000	EM24:	.ASCIZ /STP (STCS BIT1) CANNOT BE SET/
	031351	123	050124	024040		
	031356	052123	051503	041040		
	031364	052111	024461	041440		
	031372	047101	047516	020124		
	031400	042502	051440	052105		
2484	031406	000			EM25:	.ASCIZ /STP (STCS BIT1) CANNOT BE WRITTEN TO 0/
	031407	123	050124	024040		
	031414	052123	051503	041040		
	031422	052111	024461	041440		
	031430	047101	047516	020124		
	031436	042502	053440	044522		
	031444	052124	047105	052040		
2485	031452	020117	000060		EM26:	.ASCIZ /STP (STCS BIT1) CANNOT BE CLEARED BY RESET (ACR CLR)/
	031456	052123	020120	051450		
	031464	041524	020123	044502		
	031472	030524	020051	040503		
	031500	047116	052117	041040		
	031506	020105	046103	040505		
	031514	042522	020104	054502		
	031522	051040	051505	052105		
	031530	024040	041501	020122		
2486	031536	046103	024522	000	EM27:	.ASCIZ /ENB (STCS BIT0) CANNOT BE SET/
	031543	105	041116	024040		
	031550	052123	051503	041040		
	031556	052111	024460	041440		
	031564	047101	047516	020124		
	031572	042502	051440	052105		
2487	031600	000			EM30:	.ASCIZ /ENB (STCS BIT0) CANNOT BE WRITTEN TO 0/
	031601	105	041116	024040		
	031606	052123	051503	041040		
	031614	052111	024460	041440		
	031622	047101	047516	020124		

	031630	042502	053440	044522		
	031636	052124	047105	052040		
	031644	020117	000060			
2488	031650	047105	020102	051450	EM31:	.ASCIZ /ENB (STCS BIT0) CANNOT BE CLEARED BY RESET (ACR CLR)/
	031656	041524	020123	044502		
	031664	030124	020051	040503		
	031672	047116	052117	041040		
	031700	020105	046103	040505		
	031706	042522	020104	054502		
	031714	051040	051505	052105		
	031722	024040	041501	020122		
	031730	046103	024522	000		
2489	031735	123	041524	020123	EM32:	.ASCIZ /STCS BIT15-8 LOAD-READ ERROR/
	031742	044502	030524	026465		
	031750	020070	047514	042101		
	031756	051055	040505	020104		
	031764	051105	047522	000122		
2490	031772	052123	051503	041040	EM33:	.ASCIZ /STCS BIT15-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	032000	052111	032461	034055		
	032006	053440	051105	020105		
	032014	047516	020124	046103		
	032022	040505	042522	020104		
	032030	054502	051040	051505		
	032036	052105	024040	041501		
	032044	020122	046103	024522		
	032052	000				
2491	032053	111	051515	020113	EM34:	.ASCIZ /IMSK BIT11-8 LOAD-READ ERROR/
	032060	044502	030524	026461		
	032066	020070	047514	042101		
	032074	051055	040505	020104		
	032102	051105	047522	000122		
2492	032110	046511	045523	041040	EM35:	.ASCIZ /IMSK BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	032116	052111	030461	034055		
	032124	053440	051105	020105		
	032132	047516	020124	046103		
	032140	040505	042522	020104		
	032146	054502	051040	051505		
	032154	052105	024040	041501		
	032162	020122	046103	024522		
	032170	000				
2493	032171	111	051515	020113	EM36:	.ASCIZ /IMSK BIT3-0 LOAD-READ ERROR/
	032176	044502	031524	030055		
	032204	046040	040517	026504		
	032212	042522	042101	042440		
	032220	051122	051117	000		
2494	032225	111	051515	020113	EM37:	.ASCIZ /IMSK BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	032232	044502	031524	030055		
	032240	053440	051105	020105		
	032246	047516	020124	046103		
	032254	040505	042522	020104		
	032262	054502	051040	051505		
	032270	052105	024040	041501		
	032276	020122	046103	024522		
	032304	000				
2495	032305	115	041524	041040	EM40:	.ASCIZ /MTC BIT11-8 LOAD-READ ERROR/
	032312	052111	030461	034055		

	032320	046040	040517	026504		
	032326	042522	042101	042440		
	032334	051122	051117	000		
2496	032341	115	041524	041040	EM41: .ASCIZ	/MTC BIT11-8 WERE NOT CLEARED BY RESET (ACR CLR)/
	032346	052111	030461	034055		
	032354	053440	051105	020105		
	032362	047516	020124	046103		
	032370	040505	042522	020104		
	032376	054502	051040	051505		
	032404	052105	024040	041501		
	032412	020122	046103	024522		
	032420	000				
2497	032421	115	041524	041040	EM42: .ASCIZ	/MTC BIT3-0 LOAD-READ ERROR/
	032426	052111	026463	020060		
	032434	047514	042101	051055		
	032442	040505	020104	051105		
2498	032450	047522	000122		EM43: .ASCIZ	/MTC BIT3-0 WERE NOT CLEARED BY RESET (ACR CLR)/
	032454	052115	020103	044502		
	032462	031524	030055	053440		
	032470	051105	020105	047516		
	032476	020124	046103	040505		
	032504	042522	020104	054502		
	032512	051040	051505	052105		
	032520	024040	041501	020122		
	032526	046103	024522	000		
2499	032533	111	051511	020124	EM44: .ASCII	/IIST ERROR - REFER TO THE LISTING AT 'PC'/'
	032540	051105	047522	020122		
	032546	020055	042522	042506		
	032554	020122	047524	052040		
	032562	042510	046040	051511		
	032570	044524	043516	040440		
	032576	020124	050042	021103		
2500	032604	052200	020117	042504	.ASCIZ	<CRLF>/TO DETERMINE THE ERROR./
	032612	042524	046522	047111		
	032620	020105	044124	020105		
	032626	051105	047522	027122		
	032634	000				

2502	032635	120	051501	020123	DH1:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	ACR/
	032642	020040	050040	020103							
	032650	020040	020040	042440							
	032656	050130	052103	020104							
	032664	040440	052103	040525							
	032672	020114	040440	051103							
	032700	000									
2503	032701	120	051501	020123	DH2:	.ASCIZ	/PASS	PC	ACR/		
	032706	020040	050040	020103							
	032714	020040	020040	040440							
	032722	051103	000								
2504	032725	120	051501	020123	DH3:	.ASCIZ	/PASS	PC	PGTE/		
	032732	020040	050040	020103							
	032740	020040	020040	050040							
	032746	052107	000105								
2505	032752	040520	051523	020040	DH4:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	PGTE/
	032760	020040	041520	020040							
	032766	020040	020040	054105							
	032774	041520	042124	020040							
	033002	041501	052524	046101							
	033010	020040	043520	042524							
	033016	000									
2506	033017	120	051501	020123	DH5:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	STTE/
	033024	020040	050040	020103							
	033032	020040	020040	042440							
	033040	050130	052103	020104							
	033046	040440	052103	040525							
	033054	020114	051440	052124							
	033062	000105									
2507	033064	040520	051523	020040	DH6:	.ASCIZ	/PASS	PC	STTE/		
	033072	020040	041520	020040							
	033100	020040	020040	052123							
	033106	042524	000								
2508	033111	120	051501	020123	DH7:	.ASCIZ	/PASS	PC	STCS/		
	033116	020040	050040	020103							
	033124	020040	020040	051440							
	033132	041524	000123								
2509	033136	040520	051523	020040	DH8:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	STCS/
	033144	020040	041520	020040							
	033152	020040	020040	054105							
	033160	041520	042124	020040							
	033166	041501	052524	046101							
	033174	020040	052123	051503							
	033202	000									
2510	033203	120	051501	020123	DH9:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	IMSK/
	033210	020040	050040	020103							
	033216	020040	020040	042440							
	033224	050130	052103	020104							
	033232	040440	052103	040525							
	033240	020114	044440	051515							
	033246	000113									
2511	033250	040520	051523	020040	DH10:	.ASCIZ	/PASS	PC	IMSK/		
	033256	020040	041520	020040							
	033264	020040	020040	046511							
	033272	045523	000								
2512	033275	120	051501	020123	DH11:	.ASCIZ	/PASS	PC	EXPCTD	ACTUAL	MTCE/


```
2528 033736 001236 001116 023614 DT3: .WORD $PASS,$ERRPC,LINMBR,$GDDAT,$BDDAT,DISPLY,$TMPO,$TMP1,0
      033744 001124 001126 023610
      033752 001176 001200 000000
2529 033760 001 000 000 DAF1: .BYTE 1,0,0,0,0,0,0,0
      033763 000 000 000
      033766 000 000
2530 .EVEN
2531
2532 000001 .END ;THAT'S ALL FOLKS!
```

ABASE = 000000	ASWREG= 000000	DCFH 023600	EM33 031772	KIPDR3= 172306
ACDW1 = 000000	AESTN= 000000	DCFTAB 020526	EM34 032053	KIPDR4= 172310
ACDW2 = 000000	AUNIT = 000000	DCF0 = 000001	EM35 032110	KIPDR5= 172312
ACPUOP= 000000	AUSWR = 000000	DCF1 = 000002	EM36 032171	KIPDR6= 172314
ACR 023572	AVECT1= 000000	DCF2 = 000004	EM37 032225	KIPDR7= 172316
ACRH 023574	AVECT2= 000000	DCF3 = 000010	EM4 030530	LF = 000012
ADDW0 = 000000	BEGIN 002524	DDISP = 177570	EM40 032305	LINE0 007234
ADDW1 = 000000	BIT0 = 000001	DH1 032635	EM41 032341	LINE1 007274
ADDW10= 000000	BIT00 = 000001	DH10 033250	EM42 032421	LINE2 007306
ADDW11= 000000	BIT01 = 000002	DH11 033275	EM43 032454	LINMBR 023614
ADDW12= 000000	BIT02 = 000004	DH12 033342	EM44 032533	LINTST 007324
ADDW13= 000000	BIT03 = 000010	DH13 033367	EM5 030611	LKE = 000004
ADDW14= 000000	BIT04 = 000020	DH2 032701	EM6 030645	MAXCNT 023666
ADDW15= 000000	BIT05 = 000040	DH3 032725	EM7 030112	MDSD = 000001
ADDW2 = 000000	BIT06 = 000100	DH4 032752	ENB = 000001	MENB = 000004
ADDW3 = 000000	BIT07 = 000200	DH5 033017	ENDTST 020546	MFRM = 002000
ADDW4 = 000000	BIT08 = 000400	DH6 033064	ERR = 100000	MIDO = 000000
ADDW5 = 000000	BIT09 = 001000	DH7 033111	ERRVEC= 000004	MID1 = 000400
ADDW6 = 000000	BIT1 = 000002	DH8 033136	EXC = 023576	MID2 = 001000
ADDW7 = 000000	BIT10 = 002000	DH9 033203	EXCE = 000010	MID3 = 001400
ADDW8 = 000000	BIT11 = 004000	DISPLA 001142	EXCH 023600	MINCNT 023664
ADDW9 = 000000	BIT12 = 010000	DISPLY 023610	GETPAR 022336	MLEN = 000002
ADEVCT= 000000	BIT13 = 020000	DISPRE 000174	GETSID 023474	MMVEC = 000250
ADEVVM 000000	BIT14 = 040000	DMPREG 022234	GO = 000001	MNTBTS 022202
ADR 023576	BIT15 = 100000	DMPSID 033674	GR1 = 040000	MTC 023576
ADRH 023600	BIT2 = 000004	DSBT = 000010	GTSWR = 104406	MTCE = 000015
AENV 000000	BIT3 = 000010	DSWR = 177570	HDRMES 033561	MTCH 023600
AENVM = 000000	BIT4 = 000020	DT1 033712	HT 000011	MTYP = 004000
AFATAL 000000	BIT5 = 000040	DT2 033726	IE = 000004	N = 000044
AMADR1= 000000	BIT6 = 000100	DT3 033736	IIIL 023606	NOPAR 002502
AMADR2= 000000	BIT7 = 000200	EMTVEC= 000030	IIIP 023604	PARCAL 022214
AMADR3= 000000	BIT8 = 000400	EM1 030000	IIIV 023602	PARFLG 023656
AMADR4= 000000	BIT9 = 001000	EM10 030147	IMITAB 020526	PBFTAB 020536
AMAMS1= 000000	BMITAB 020536	EM11 030215	IMSK 023576	PBF0 = 000400
AMAMS2 000000	BM0 = 000400	EM12 030301	IMSKE = 000004	PBF1 = 001000
AMAMS3= 000000	BM1 = 001000	EM13 030337	IMSKH 023600	PBF2 = 002000
AMAMS4 000000	BM2 = 002000	EM14 030406	IMO = 000001	PBF3 = 004000
AMSGAD= 000000	BM3 004000	EM15 030725	IM1 = 000002	PB0 = 000400
AMSGLG 000000	BPTVEC 000014	EM16 030762	IM2 = 000004	PB1 = 001000
AMSGTY= 000000	BRKTAB 020536	EM17 031043	IM3 = 000010	PB2 = 002000
AMTYP1= 000000	BRK0 000400	EM2 030033	IOTVEC= 000020	PB3 = 004000
AMTYP2 000000	BRK1 = 001000	EM20 031077	IP = 000010	PGBTAB 020536
AMTYP3 000000	BRK2 = 002000	EM21 031157	KIPAR0= 172340	PGCS 023576
AMTYP4= 000000	BRK3 = 004000	EM22 031215	KIPAR1= 172342	PGCSE = 000001
APASS = 000000	CARET 033710	EM23 031264	KIPAR2= 172344	PGCSH 023600
APRIOR 000000	CKSWR = 104407	EM24 031351	KIPAR3= 172346	PGF 023576
APTCSU 000040	CLR = 100000	EM25 031407	KIPAR4 172350	PGFF = 000005
APTENV= 000001	CONF IG 023660	EM26 031456	KIPAR5= 172352	PGFH 023600
APTSIZ= 000200	CR = 000015	EM27 031543	KIPAR6= 172354	PGITAB 020526
APTSPO= 000100	CRLF = 000200	EM3 030473	KIPAR7= 172356	PGMR = 020000
ASKBRL 033540	DAF1 033760	EM30 031601	KIPDR0= 172300	PGONL 016072
ASKDVA 033465	DCF 023576	EM31 031650	KIPDR1= 172302	PGTE 023576
ASKPIV 033505	DCFE = 000007	EM32 031735	KIPDR2 172304	PGTEE 000000

PIFTAB	020526	SID2	=	001000	SW2	=	000004	TST5	003414	\$DDW13	001346									
PIFO	=	000001	SID3	=	001400	SW3	=	000010	TST50	015226	\$DDW14	001350								
PIF1	=	000002	SIFTAB	020526	SW4	=	000020	TST51	015500	\$DDW15	001352									
PIF2	=	000004	SIFO	=	000001	SW5	=	000040	TST52	015666	\$DDW2	001320								
PIF3	=	000010	SIF1	=	000002	SW6	=	000100	TST53	016072	\$DDW3	001322								
PIRQ	=	177772	SIF2	=	000004	SW7	=	000200	TST54	016716	\$DDW4	001324								
PIRQVE	=	000240	SIF3	=	000010	SW8	=	000400	TST55	017674	\$DDW5	001326								
PI0	=	000001	SIO	=	000001	SW9	=	001000	TST56	020546	\$DDW6	001330								
PI1	=	000002	SI2	=	000002	TBITVE	=	000014	TST6	003500	\$DDW7	001332								
PI2	=	000004	SI3	=	000010	TCR	022334	TST7	003556	TWOSP	033671	\$DDW8	001334							
PI3	=	000010	SNAPC	023616	TIMPER	023612	TKVEC	=	000060	TYPDS	=	104405	\$DDW9	001336						
POP	=	005726	SPACE	033667	TMO	=	000010	TMT	=	000016	TYPE	=	104401	\$DEVCT	001240					
POPPOP	=	022626	SRO	177572	TPVEC	=	000064	TRAPVE	=	000034	TYPDC	=	104402	\$DEVN	001306					
PRDY	=	004000	SR1	=	177574	TRTVEC	=	000014	UI0	=	000400	TYPON	=	104404	\$DOAGN	023766				
PRO	=	000000	SR2	=	177576	TST1	002524	UI1	=	001000	TYPON	=	104404	\$DTBL	027252					
PR1	000040	SR3	172516	SSTMNT	022204	TST10	003634	UI2	=	002000	UI3	=	004000	\$ENDAD	023756					
PR2	=	000100	STACK	=	001100	TST11	003720	UVITAB	020536	W	=	000005	\$ENDCT	023724						
PR3	=	000140	START	002014	TST12	003776	TST13	004212	\$APTHD	001000	\$ATYC	027556	\$ENDMG	023775						
PR4	=	000200	STBTAB	020536	TST14	004426	TST15	004504	\$ATY1	027532	\$ATY3	027540	\$ENULL	023772						
PR5	=	000240	STCCNT	023662	TST16	004570	TST17	004646	\$ATY4	027550	\$AUTOB	001134	\$ENV	001250						
PR6	=	000300	STCS	023576	TST2	002706	TST20	004724	\$BASE	001304	\$BDADR	001122	\$ENVN	001251						
PR7	=	000340	STCSE	=	000003	TST21	005010	TST22	005066	\$BDADR	001122	\$BELL	001220	\$EOP	023670					
PS	=	177776	STCSH	023600	TST23	005144	TST24	005230	\$BDAT	001126	\$CDW1	001310	\$EOPCT	023716						
PSW	=	177776	STF	023576	TST25	005306	TST26	005522	\$CDW2	001312	\$CHARC	026614	\$ERFLG	001103						
PTP	000002	STFE	=	000006	TST27	005736	TST3	003122	\$CM1	=	000006	\$CKSWR	025430	\$ERMAX	001115					
PWRVEC	000024	STITAB	020526	STITAB	020526	TST30	006152	TST31	006366	\$CM2	=	000014	\$CMTAG	001100	\$ERROR	024264				
RDCHR	=	104410	STKLMT	=	177774	TST32	006606	TST33	007332	\$CM3	=	000006	\$CPUOP	001256	\$ERRPC	001116				
RDLIN	=	104411	STMR	=	010000	TST34	010222	TST35	010550	\$CM4	=	000007	\$CROUT	025220	\$ERRTB	001354				
RDOCT	=	104412	STONL	017674	TST36	011344	TST4	003336	\$CNTLG	026307	\$CNTLU	026302	\$ERRTY	024606	\$ERTTL	001112				
RESVEC	000010	STP	=	000002	TST37	011672	TST40	012166	\$CDW1	001310	\$CDW2	001312	\$ESCAP	001216	\$FATAL	001232				
RTETAB	020526	STTE	023576	STTEE	000002	TST38	011772	TST41	012414	\$CDW2	001312	\$CHARC	026614	\$ETABL	001250	\$FILLC	001156			
RTE0	=	000001	SWR	001140	SWREG	000176	TST39	003122	\$CORE	025170	\$CRLF	001225	\$SETEND	001354	\$FILLS	001155				
RTE1	=	000002	SW00	000001	SW01	=	000002	TST42	012710	\$CPUOP	001256	\$CROUT	025220	\$FATAL	001232	\$FILLG	027776			
RTE2	000004	SW02	=	000004	SW02	=	000004	TST43	013136	\$DBLK	027262	\$DDW0	001314	\$FFLG	027776	\$FILLC	001156			
RTE3	000010	SW03	=	000010	SW03	=	000010	TST44	013406	\$DDW1	001316	\$DDW10	001340	\$FILLS	001155	\$GDADR	001120			
R6	=	000006	SW04	=	000020	SW04	=	000020	TST45	013634	\$DDW11	001342	\$GDAT	001124	\$GDADR	001120	\$GDDAT	001124		
R7	=	000007	SW05	=	000040	SW05	=	000040	TST46	014032	\$DDW12	001344	\$GET42	023746	\$GDDAT	001124	\$GET42	023746		
SBFTAB	020536	SW06	=	000100	SW06	=	000100	TST47	014560	\$DDW13	001346	\$SHD	=	000000	\$GDSWR	025500	\$GTSWR	025500		
SBFO	=	000001	SW07	=	000200	SW07	=	000200					\$SHIBTS	001000	\$HD	=	000000	\$SHD	=	000000
SBF1	=	000002	SW08	=	000400	SW08	=	000400					\$SHIOCT	025426	\$IBLK	027262	\$IBLK	027262	\$IBLK	027262
SBF2	=	000004	SW09	=	001000	SW09	=	001000					\$SCORE	025170	\$DDW0	001314	\$DDW1	001316	\$DDW10	001340
SBF3	000010	SW10	=	002000	SW10	=	002000						\$SCOUT	025220	\$DDW11	001342	\$DDW12	001344	\$DDW13	001346
SBO	000400	SW11	=	004000	SW11	=	004000						\$SCOUT	025220	\$DDW14	001350	\$DDW15	001352	\$DDW2	001320
SB1	=	001000	SW12	=	010000	SW12	=	010000					\$SDBLK	027262	\$DDW3	001322	\$DDW4	001324	\$DDW5	001326
SB2	=	002000	SW13	=	020000	SW13	=	020000					\$SDBLK	027262	\$DDW6	001330	\$DDW7	001332	\$DDW8	001334
SB3	=	004000	SW14	=	040000	SW14	=	040000					\$SDBLK	027262	\$DDW9	001336	\$DDW13	001346	\$DDW14	001350
SELPAR	002512	SW15	=	100000	SW15	=	100000						\$SDBLK	027262	\$DEVCT	001240	\$DEVN	001306	\$DOAGN	023766
SETMNT	022042												\$SDBLK	027262	\$DTBL	027252	\$ENDAD	023756	\$ENDCT	023724
SETMNT1	022046												\$SDBLK	027262	\$ENDMG	023775	\$ENULL	023772	\$ENV	001250
SIDNUM	023570												\$SDBLK	027262	\$ENVN	001251	\$EOP	023670	\$EOPCT	023716
SIDTAB	020516												\$SDBLK	027262	\$ERFLG	001103	\$ERMAX	001115	\$ERROR	024264
SIDO	=	000000											\$SDBLK	027262	\$ERRPC	001116	\$ERRTB	001354	\$ERRTY	024606
SID1	=	000400											\$SDBLK	027262	\$ERTTL	001112	\$ESCAP	001216	\$ETABL	001250

SLPADR 001106	SMTYP1 001261	\$RDSZ = 000014	\$TESTN 001234	\$TSTM 001102
SLPERR 001110	SMTYP2 001265	\$REGAD 001160	\$TIMES 001214	\$TTYIN 026266
SLSTAD 025264	SMTYP3 001271	\$REG0 001162	\$TKB 001146	\$TYPDS 027046
SLSTBK 025266	SMTYP4 001275	\$REG1 001164	\$TKS 001144	\$TYPE 026336
SMADR1 001262	SMXCNT 024262	\$REG2 001166	\$TMP0 001176	\$TYPEC 026550
SMADR2 001266	\$NULL 001154	\$REG3 001170	\$TMP1 001200	\$TYPEX 026616
SMADR3 001272	\$NWTST= 000001	\$REG4 001172	\$TMP2 001202	\$TYPOC 026644
SMADR4 001276	\$OCNT 027042	\$REG5 001174	\$TMP3 001204	\$TYPON 026660
\$MAIL 001230	\$OMODE 027044	\$RTNAD 023770	\$TMP4 001206	\$TYPOS 026620
\$MAMS1 001260	\$OVER 024246	\$SAVR6 027520	\$TMP5 001210	\$UNIT 001242
\$MAMS2 001264	\$PASS 001236	\$SCOPE 024012	\$TMP6 001212	\$UNITM 001010
\$MAMS3 001270	\$PASTM 001006	\$SETUP= 000137	\$TN = 000057	\$USWR 001254
\$MAMS4 001274	\$POWER 027522	\$SIZE 024762	\$TPB 001152	\$VECT1 001300
\$MEADR 001002	\$PWRDN 027354	\$SIZFX 025224	\$TPFLG 001157	\$VECT2 001302
\$MFLG 027774	\$PWRMG 027510	\$STUP = 177777	\$TPS 001150	\$XTSTR 024036
\$MNEW 026325	\$PWRUP 027426	\$SVLAD 024212	\$TRAP 027272	\$GET4= 000000
\$MSGAD 001244	\$QUES 001224	\$SVP = 000210	\$TRAP2 027314	\$OFILL 027043
\$MSGLG 001246	\$RDCHR 025712	\$SWR = 167000	\$TRP 000013	\$OCAT 000000
\$MSGTY 001230	\$RDLIN 026032	\$SWREG 001252	\$TRPAD 027326	033770
\$MSWR 026314	\$RDOCT 025270	\$SWRMK - 000000	\$TSTM 001004	\$.SX 001000

. ABS. 033770 000

ERRORS DETECTED: 0

CRIIAA.BIC,CRIIAA,CRIIAA.P11
 RUN-TIME: 129 74 2 SECONDS
 RUN-TIME RATIO: 787/206=3.8
 CORE USED: 28K (55 PAGES)