

# MNCKW

MNCKW DIAGNOSTIC  
MD-11-DVMNC-A

EP-DVMNC-A-DL-A

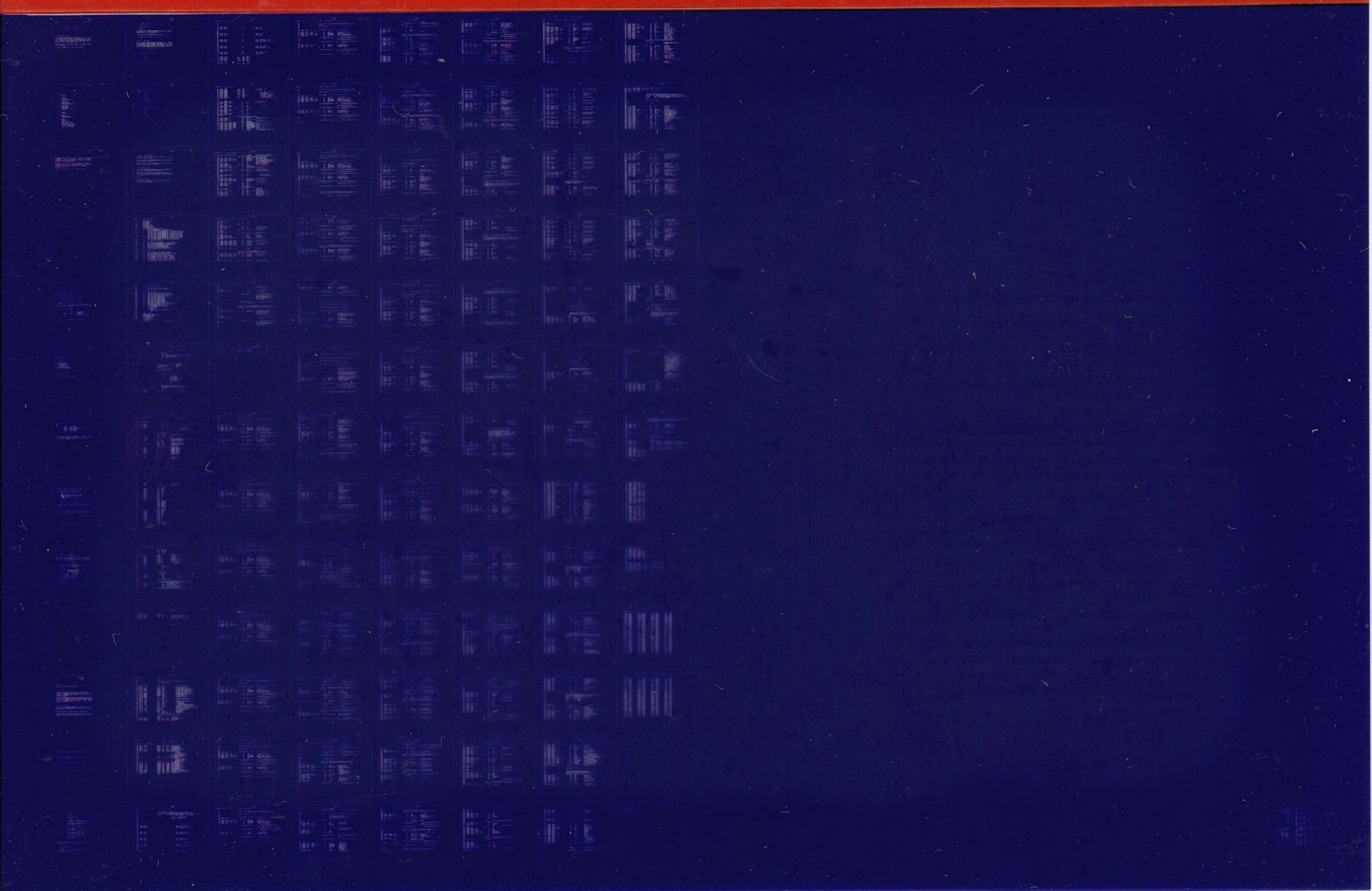
APR 1978

COPYRIGHT © 1978

**digital**

FICHE 1 OF 1

MADE IN USA



B01

REF ID: A0256041

00010000

780330

IDENTIFICATION

33MOR10VMNCASE0

00010000

780330  
SER 0001

PRODUCT CODE: MAINDEC-11-DVMNC-A-D  
PRODUCT NAME: MNCKW DIAGNOSTIC  
DATE CREATED: MARCH 1978  
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1978  
DIGITAL EQUIPMENT CORPORATION

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

1.0 ABSTRACT

2.0 REQUIREMENTS  
 2.1 EQUIPMENT  
 2.2 STORAGE

3.0 LOADING PROCEDURE  
 3.1 METHOD  
 3.2 NON-STANDARD ADDRESS, VECTOR, OR USE OF SOFTWARE SWITCH REGISTER

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

5.0 OPERATING PROCEDURE  
 5.1 SWITCH REGISTER FUNCTION  
 5.2 SCOPE LOOPS  
 5.3 PROGRAM AND/OR OPERATOR ACTION  
 5.3.1 LOGIC TEST  
 5.4 INHIBITING AUTO-SIZE FEATURE

6.0 ERRORS  
 6.1 ERROR PRINTOUT  
 6.1.1 EXAMPLE  
 6.2 NON-STANDARD ERROR HALTS

7.0 RESTRICTIONS  
 7.1 EXTERNAL INPUTS  
 7.2 STARTING RESTRICTION  
 7.3 POSSIBLE PROGRAM "BOMBS"

8.0 MISCELLANEOUS  
 8.1 POWER FAIL  
 8.2 XXDP, ACT, APT  
 8.3 EXECUTION TIME  
 8.4 LSI-11 "ODT" COMMANDS  
 8.5 ENTERING LSI-11 "ODT"  
 8.6 USE OF PROGRAM SOFTWARE SWR  
 8.7 SPECIAL I/O SIGNAL TESTS  
 8.8 PRODUCTION STARTING ADDRESS  
 8.9 TESTOR STARTING ADDRESS  
 8.10 DWARF STARTING ADDRESS

. C ABSTRACT  
-----

THIS PROGRAM ALLOWS THE USER TO CHECK-OUT OR DEBUG THE MNCKW PROGRAMMABLE REAL-TIME CLOCK. THE LOGIC TEST IS SELF CONTAINED AND NEEDS TO EXTERNAL MAINTENANCE HARDWARE OR OPERATOR INTERVENTION WITH ONLY ONE EXCEPTION: IF THE CUSTOMER HARDWARE CONNECTED TO THE MNCKW COULD INJECT SIGNALS ON ST2, ST1, OR SLAVE IN INPUTS, IT MUST BE DISCONNECTED.

EVEN THOUGH THE MNCKW IS A Q BUS OPTION, THIS PROGRAM WAS DESIGNED TO RUN ON ANY PDP-11 FAMILY COMPUTER. IF THE USER IS UNFAMILIAR WITH AN LSI-11 HE SHOULD REVIEW SECTIONS 8.4 AND 8.5. A SOFTWARE SWITCH REGISTER IS INCLUDED WITH THIS PROGRAM. IT CAN BE USED ON AN LSI-11 OR BY CPU'S THAT HAVE HARDWARE SWITCH REGISTERS. SEE SECTION 8.6.

EVERY EFFORT WAS MADE TO MAKE THIS PROGRAM CONFORM TO LSI-11 PROGRAMMING RESTRICTIONS, HOWEVER: THE USER SHOULD READ SECTIONS 7.2 AND 7.3.



2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP-11 FAMILY COMPUTER WITH 8K OF MEMORY (OR MORE) AND I/O FACILITIES (A SWITCH REGISTER OR TTY).
2. MNCKW UNDER TEST.

2.2 STORAGE

THIS PROGRAM OCCUPIES AND USES ONLY THE LOWER 8K OF MEMORY.

3.0 LOADING PROCEDURE

3.1 METHOD

STANDARD PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED.

1. ABSOLUTE LOADER MUST BE IN MEMORY.
2. PLACE BINARY TAPE IN READER.
3. TYPE ADDRESS \*7500 (5\* DETERMINE BY LOCATION OF LOADER .
4. TYPE "G" (PROGRAM WILL BE LOADED INTO MEMORY).

THE PROGRAM CAN ALSO BE LOADED BY XXDP, ACT, OR APT.

3.2 NON-STANDARD ADDRESS, VECTOR, OR USE OF SOFTWARE SWITCH REGISTER

THIS PROGRAM IS SET TO TEST A MNCKW WITH A STANDARD ADDRESS AND VECTOR. IF ANY OF THESE ARE DIFFERENT ON THE MNCKW YOU ARE TESTING, CHANGE THE CORRESPONDING LOCATION IN MEMORY BEFORE STARTING THIS TEST.

<u>LOCATION</u>	<u>TAG</u>	<u>CURRENT CONTENTS</u>	<u>COMMENTS</u>
1250	\$BASE:	170420	:: BASE ADDRESS OF EQUIPMENT :: UNDER TEST
1244	\$VEcT1:	000440	:: INTERRUPT VECTOR #1
176	\$SWREG:	000000	:: MANUAL SWR.
1157	\$TPFLG:	.BYTE 0	:: "TERMINAL AVAILABLE" :: FLAG (BIT<0:7>=0=YES)

NOTE

SINCE NO HARDWARE SWITCH REGISTER EXISTS, YOU MAY SET ANY BIT IN "\$SWREG" AS YOU WOULD HAVE SET IT IN THE SWR.

4.0 STARTING PROCEDURE  
-----

4.1 CONTROL SWITCH SETTING

BEFORE STARTING THE DIAGNOSTIC, SET ALL SWITCH REGISTER BITS AS DESIRED. SEE SECTION 5.1.

4.2 STARTING ADDRESSES

200 START OF LOGIC TESTS  
204 RESTART ADDRESS FOR LOGIC TEST  
210 I/O SIGNAL TEST #1  
214 I/O SIGNAL TEST #2  
220 I/O SIGNAL TEST #3  
230 PRODUCTION STARTING ADDRESS  
240 TESTOR STARTING ADDRESS  
250 DWARF STARTING ADDRESS

4.3 PROGRAM AND/OR OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY.
2. ENTER KEYBOARD "ODT".
3. ALTER LOCATION "SWREG" TO REFLECT DESIRED OPTIONS OF A SWITCH REGISTER - SEE SECTION 5.1.
4. TYPE STARTING ADDRESS, FOLLOWED BY "G" TO START PROGRAM.

5.0 OPERATING PROCEDURE

5.1 SWITCH REGISTER FUNCTION

SWR BIT	OCTAL	FUNCTION WHEN SET
15	100000	HALT ON ERROR
14	040000	LOOP ON TEST
13	020000	INHIBIT ERROR TYPEOUT
12	010000	ENABLE LINE FREQ. RATE TESTING
11	004000	INHIBIT ITERATIONS (SHORT PASS)
10	002000	BELL ON ERROR
09	001000	LOOP ON ERROR
08	000400	LOOP ON TEST IN SWR <7:0

5.2 SCOPE LOOPS

IF AN ERROR OCCURS AND THE USER WISHES TO SCOPE THE ERROR, "\$SWREG" SHOULD BE ALTERED TO "100000" AT THE START OF THE TEST TO HALT ON ERROR. THEN WHEN THE PROGRAM HALTS ON ERROR AND THE CPU ENTERS "ODT", "\$SWREG" SHOULD BE ALTERED TO "060000" TO LOOP ON CURRENT TEST AND INHIBIT ERROR TYPEOUT. THEN TYPE "P" TO CONTINUE PROGRAM EXECUTION.



5.3 PROGRAM AND CR OPERATOR ACTION

5.3.1 LOGIC TEST

THE FIRST PASS THROUGH THE PROGRAM WILL BE MADE WITH ITERATIONS INHIBITED. SUCCESSIVE PASSES WILL ENABLE ITERATIONS IF SWR11=0.

IF NOT INHIBITED BY APT, THE PROGRAM WILL LOOK FOR MORE HWV11'S TO EXERCISE. ONE PASS WILL EXERCISE ALL MNCKW'S.

IF FOUR UNITS ARE DETECTED, THE FOLLOWING WILL BE TYPED:

UNIT #000001 COMPLETED TESTING UNIT #000002  
UNIT #000002 COMPLETED TESTING UNIT #000003  
UNIT #000003 COMPLETED TESTING UNIT #000004  
UNIT #000004 COMPLETED

AT END OF PASS WHEN ALL UNITS HAVE BEEN TESTED, THE FOLLOWING OUTPUT WILL OCCUR:

"ENDPASS 12 - TOTAL ERRORS 4 ;THERE ARE 4 (OCTAL) UNITS.

THE GOOD UNITS ARE ( L TO R) 000000000001011

THIS INDICATES THAT THE PROGRAM HAS COMPLETED 12 (OCTAL DECIMAL) PASSES. DURING THAT TIME 4(OCTAL) ERRORS WERE DETECTED. ALSO WE TESTED 4 UNITS AND THE THIRD UNIT WAS THE ONLY UNIT TO FAIL.

5.4 INHIBITING AUTO-SIZE FEATURE

THIS PROGRAM WILL AUTOMATICALLY AUTO-SIZE AND TEST EACH MNCKW IT DETECTS ON THE SYSTEM. TO INHIBIT THIS FEATURE, SET BIT 15 OF LOCATION "SENVM". ALSO, TO TEST AN INDIVIDUAL MNCKW IN A GROUP, SET THIS BIT AND REFER TO SECTION 3.2 FOR CHANGING THE BASE ADDRESS OF THE MNCKW UNDER TEST.

6.0 ERRORS  
-----

6.1 ERROR PRINTOUT

PRINTOUT VARIES WITH THE ERROR DETECTED. THE ERROR PC TYPED OUT IS THE ACTUAL LOCATION OF THE ERROR CALL.

A HALT AT LOCATION "STYPE"+10 WHEN RUNNING WITH NO TERMINAL INDICATES AN ERROR HAS OCCURRED. TO FIND OUT THE NUMBER OF THE ERROR, EXAMINE LOCATION "STSTNM". THIS IS THE ITEM NUMBER OF THE ERROR. TO FIND OUT WHAT THE ERROR TYPEOUT WOULD HAVE BEEN GO TO THE ERROR POINTER TABLE BEGINNING AT LOCATION "ERRTB".

6.1.1 EXAMPLE

IF WE EXAMINED LOCATION "STSTNM" AND FOUND A 5(101) WE GO TO LOCATION "ERRTB" AND LOOK THROUGH THE ERROR POINTER TABLE UNTIL WE FOUND ITEM 5. THE INFORMATION WOULD LOOK LIKE:

:ITEM 5

EMS :CLOCK SR DATA ERROR  
DMS :ERRPC ASR WAS 5/B  
DTS :SERRPC,ASR,\$BDDAT,\$GDDAT  
DFO :ALL NUMBERS ARE IN OCTAL FORM

TO FIND OUT THE INFORMATION SPECIFIED BY DTS (SERRPC,BSR,\$BDADR,\$GDADR) FOLLOW THESE STEPS:

1. LOOK UP THE ADDRESS OF THE LABEL (I.E., SERRPC) IN THE SYMBOL TABLE WHICH FOLLOWS THE LISTING.
2. \*PUT THIS ADDRESS IN THE SWITCH REGISTER AND DEPRESS THE LOAD ADDRESS SWITCH ON THE PROCESSOR'S CONSOLE.
3. \*NOW DEPRESS THE EXAMINE SWITCH.
4. \*THE DATA DISPLAYED IN THE DATA LIGHTS IS THE INFORMATION THAT WOULD HAVE BEEN PRINTED FOR HIS LABEL IF YOU HAD A INPUT/OUTPUT TERMINAL.

-----  
\* SEE SECTION 8.4 FOR LSI-11 ODT COMMANDS.

6.2 NON-STANDARD ERROR HALTS

ANY HALT IN THE TRAP CATCHER AREA LOCATIONS 000000-C01000,  
INDICATES:

1. THE MCKW INTERRUPTED TO A WRONG VECTOR ADDRESS, OR
2. TIME-OUT OR ILLEGAL INSTRUCTION HARDWARE TRAP.

## 7.0 RESTRICTIONS

### 7.1 EXTERNAL INPUTS

EXTERNAL INPUTS SUCH AS "SLAVE IN", "ST1" AND "ST2" MUST NOT BE CONNECTED TO ANY CUSTOMER HARDWARE THAT MIGHT GENERATE THESE SIGNALS WHILE THE DIAGNOSTIC IS RUNNING.

### 7.2 STARTING RESTRICTION

IF A FREE-RUNNING CLOCK, SUCH AS 60HZ FROM THE POWER SUPPLY, IS ATTACHED TO THE "BEVNT" BUS LINE ON BOTH REV LEVEL C/D AND E SYSTEMS, AN INTERRUPT TO LOCATION 100 WILL OCCUR WHEN USING THE "G" AND "L" COMMANDS PRIOR TO EXECUTING THE FIRST INSTRUCTION. THEREFORE, THIS PROGRAM CANNOT DISABLE THE BEVNT BUS LINE BY INHIBITING INTERRUPTS.

USER SYSTEMS REQUIRING A FREE-RUNNING CLOCK ATTACHED TO THE BEVNT BUS LINE CAN TEMPORARILY AVOID THIS SITUATION BY SETTING THE PSW(PS) TO 200, LOADING THE PC WITH THE STARTING ADDRESS INSTEAD OF USING THE "G" COMMAND, AND THEN USING THE "P" COMMAND. BEFORE USING THE "I" COMMAND, THE PSW(PS) CAN BE SET TO 200, THEREBY INHIBITING INTERRUPTS, TO AVOID RECEIVING THE EVENT INTERRUPT AFTER LOADING THE ABS LOADER.

### 7.3 POSSIBLE PROGRAM "BOMBS"

THE FIRST TWO TESTS OF THIS PROGRAM CHECK TO SEE IF THE MNCKW RESPONDS TO THE ADDRESS THE PROGRAM THINKS ITS AT. IF THE MNCKW DOES NOT RESPOND, A BUS ERROR OCCURS. ALSO BUS ERRORS CAN OCCUR DURING THE TIME THE PROGRAM SIZES TO SEE HOW MANY KVV11'S ARE ON YOUR SYSTEM.

FOR MORE INFORMATION ON THE NEXT SUBJECT, SEE JAN. 1976 LSI-11 ENGINEERING BULLETIN ISSUED BY THE DIGITAL COMPONENTS GROUP.

BUS ERRORS MAY ALTER THE PRESET CONTENTS OF LOCATION 4 BEFORE THE TRAP IS EXECUTED, THEREBY TRANSFERRING PROGRAM CONTROL TO AREA IN THE PROGRAM THAT WAS NOT SET UP TO HANDLE THE TRAP. IF THIS HAPPENS, THE PROGRAM WILL "BOMB" AND POSSIBLY REWRITE PARTS OF ITSELF.

8.0 MISCELLANEOUS  
-----

8.1 POWER FAIL

AFTER A POWER FAILURE OCCURS, THE PROGRAM EXECUTION WILL CONTINUE AT THE POINT WHERE THE POWER OCCURRED. THE PROGRAM WILL TYPE "POWER".

8.2 XXDP, ACT, APT

THE PROGRAM IS CHAINABLE UNDER XXDP, ACT, OR APT. ALTHOUGH "APT HOOKS" HAVE BEEN INSTALLED, THEY HAVE NOT BEEN TESTED.

8.3 EXECUTION TIME

0.5 MINUTES (30 SEC) ITERATION INHIBITED - NO ERRORS  
2.5 MINUTES (150 SEC) WITH ITERATIONS - NO ERRORS

8.4 LSI-11 "ODT" COMMANDS

FORMAT	DESCRIPTION
<CR> RETURN	CLOSE OPENED LOCATION AND ACCEPT NEXT COMMAND.
<LF> LINE FEED	CLOSE CURRENT LOCATION; OPEN NEXT SEQUENTIAL LOCATION.
↑ (UPARROW)	OPEN PREVIOUS LOCATION.
← (LEFT ARROW)	TAKE CONTENTS OF OPENED LOCATION, INDEXED BY CONTENTS OF PC, AND OPEN THAT LOCATION.
Ⓜ	TAKE CONTENTS OF OPENED LOCATION AS ABSOLUTE ADDRESS AND OPEN THAT LOCATION.
R	OPEN THE WORD AT LOCATION R. REOPEN THE LAST LOCATION.
\$N OR RN	OPEN GENERAL REGISTER N(0-7) OR S(PS REGISTER).
R:G OR RG	GOTO LOCATION R AND START PROGRAM.
NL	EXECUTE BOOTSTRAP LOADER USING N AS DEVICE CSR. CONSOLE DEVICE IS 177560.
:P OR P	PROCEED WITH PROGRAM EXECUTION.
RUBOUT	ERASES PREVIOUS NUMERIC CHARACTER. RESPONSE IS A BACKSLASH (\).

8.5 ENTERING LSI-11 "ODT"

THE HALT OR ODT MICROCODE STATE OF THE KD11F (LSI-11 MODULE) CAN BE ENTERED IN FIVE DIFFERENT WAYS (OTHERS ARE A SUBSET OF THESE), FROM THE RUN STATE:

1. EXECUTION OF A LSI-11 HALT INSTRUCTION,
2. A DOUBLE BUS ERROR,
3. AS A POWER UP OPTION,
4. ASCII BREAK WITH DLV11 FRAMING ERROR ASSERTING THE B HALT LINE (ENABLED BY JUMPER OF DLV11).

UPON ENTERING THE HALT STATE, THE KD11F RESPONDS THROUGH THE SET OF COMMANDS LISTED IN SECTION 8.4.

8.6 USE OF PROGRAM SOFTWARE SWR

THE PROGRAM SOFTWARE SWITCH REGISTER IS ENABLED IF

1. NO HARDWARE SWR EXISTS;
2. IF YOU START WITH ALL ONES (SWR=177777) IN THE SWITCH REGISTER.

THE SOFTWARE SWITCH REGISTER MAY BE CHANGED BY TYPING ↑G (CONTROL AND LETTER G KEYS TYPED SIMULTANEOUSLY). WHEN ↑G IS TYPED, THE PROGRAM RESPONDS BY TYPING "SWR=XXXXXX" WHERE XXXXXX EQUALS THE FORMER CONTENTS OF THE SWITCH REGISTER.

IF YOU WISH TO KEEP THE CURRENT VALUE, TYPE <CR>. IF YOU WISH TO CHANGE THE VALUE, TYPE THE NEW VALUE FOLLOWED BY A <CR>.

IT IS IMPORTANT TO NOTE THAT THE DIAGNOSTIC IS NOT RUNNING AFTER THE ↑G UNTIL A <CR> IS TYPED.

8.7 SPECIAL I/O SIGNAL TESTS

THREE TESTS WERE INCLUDED TO ENABLE CHECKOUT OF I/O SIGNALS: ST1, ST2, AND CLOCK OVERFLOW. THESE TESTS HAVE A SPECIAL STARTING ADDRESS. SINCE END-PASSES ARE IMMEDIATE, NO "END OF PASS" MESSAGE IS REPORTED. ERRORS ARE REPORTED BY TYPING OUT THE PC WHERE THE ERROR WAS DETECTED. WHEN STARTED, THE PROGRAM REMAINS IN A LOOP GENERATING AND DETECTING THE SPECIFIED SIGNALS. HALT ON ERROR AND INHIBIT ERROR TIMEOUT OPTIONS MAY BE USED.

LOGIC TESTS MUST HAVE ALREADY BEEN RUN ON THE MNCKW.

8.7.1 I/O SIGNAL TEST #1 ST1 IN, ST2 OUT

SWITCH PACK S2 MUST BE SET UP AS FOLLOWS:

SWITCH	STATE
1	OFF
2	ON
3	OFF
4	OFF
5	ON
6	ON
7	NOT USED

THE FOLLOWING JUMPER MUST BE INSTALLED.

J1-SS (ST2 OUT) TO J1-VV (ST1 IN)

LOAD AND START THE PROGRAM AT 210.



B.7.2 I/O SIGNAL TEST #2 CLOCK OVERFLOW TEST  
SWITCH PACK S2 MUST BE SET UP AS FOLLOWS:

SWITCH	STATE
1	OFF
2	OFF
3	OFF
4	ON
5	ON
6	OFF
7	NOT USED

THE FOLLOWING JUMPER MUST BE INSTALLED.

J1-RR (CLOCK OVERFLOW) TO J1-TT (ST2 IN)  
LOAD AND START AT LOCATION 214.

B.7.3 I/O SIGNAL TEST #3 ST1 OUT AND ST2 IN  
SWITCH PACK S2 MUST BE SET UP AS FOLLOWS:

SWITCH	STATE
1	OFF
2	OFF
3	OFF
4	ON
5	ON
6	ON
7	NOT USED

THE FOLLOWING JUMPER MUST BE INSTALLED:

J1-UU (ST1 OUT) TO J1-TT (ST2 IN)  
LOAD AND START AT LOCATION 220.

E.8 PRODUCTION STARTING ADDRESS

A SPECIAL STARTING ADDRESS HAS BEEN PROVIDED FOR IN-HOUSE PRODUCTION TO USE TO START THE LOGIC DIAGNOSTIC AND INFORM THE TEST THAT PRODUCTION IS USING IT.

IN THE FIELD ONLY ENOUGH ADDRESSES WERE ALLOTTED FOR 4 SEQUENTIAL MNCKW'S. WHEN THE LOGIC TESTS ARE STARTED AT LOCATION 200, WE ONLY AUTO-SIZE UP TO 4 MNCKW'S.

IN HOUSE TESTING MAY WISH TO EXERCISE UP TO 16 MNCKW'S AT ONE TIME. THE LOGIC TESTS MAY BE STARTED AT LOCATION 230 AND THE PROGRAM WILL AUTO SIZE UP TO 16 MNCKW'S

E.9 TESTOR STARTING ADDRESS

A SPECIAL STARTING ADDRESS HAS BEEN PROVIDED FOR MANUFACTURING TO USE TO START THE LOGIC DIAGNOSTIC AND INFORM THE PROGRAM THAT THE CLOCK MODULE IS CABLED TO AN IN-HOUSE TESTOR.

MANUAL INTERVENTION IS NEEDED IN THIS SEQUENCE OF TESTING. THE PROGRAM WILL TYPE OUT ALL INSTRUCTIONS. A CABLE SHOULD CONNECT J1 ON THE CLOCK MODULE TO J10 ON THE TESTOR. SWITCHES 1 AND 3 OF S2 (ON THE CLOCK MODULE) SHOULD BE ON, ALL OTHER SWITCHES ON S2 SHOULD BE OFF.

E.10 DWARF STARTING ADDRESS

MORE COMPLETE TESTING OF THE CLOCKS I/O SIGNALS CAN BE MADE IF A DWARF MODULE IS CONNECTED TO THE CLOCK. IF YOU DO THIS, START THE DWARF'S STARTING ADDRESS. A SERIES OF INSTRUCTIONS WILL BE TYPED OUT FOR YOU TO FOLLOW.

27	OPERATIONAL SWITCH SETTINGS
39	TRAP CATCHER
1	BASIC DEFINITIONS
189	ACT11 HOOKS
200	APT PARAMETER BLOCK
222	COMMON TAGS
266	APT MAILBOX-ETABLE
315	ERROR POINTER TABLE
458	INITIALIZE THE COMMON TAGS
558	TYPE PROGRAM NAME
523	GET VALUE FOR SOFTWARE SWITCH REGISTER
0	T1 *TEST THE ADDRESSABILITY OF CLOCK CSR
611	T2 *TEST THE ADDRESSABILITY OF CLOCK BUFFER REG.
649	T3 *TEST THAT CLOCK A STATUS REGISTER BIT 14 CAN BE SET AND CLEARED
695	T4 *TEST THAT CLOCK A STATUS REGISTER BIT 13 CAN BE SET AND CLEARED
741	T5 *TEST THAT CLOCK A STATUS REGISTER BIT 11 CAN BE SET AND CLEARED
791	T6 *TEST THAT CLOCK A STATUS REGISTER BIT 6 CAN BE SET AND CLEARED
836	T7 *TEST THAT CLOCK A STATUS REGISTER BIT 5 CAN BE SET AND CLEARED
879	T10 *TEST THAT CLOCK A STATUS REGISTER BIT 4 CAN BE SET AND CLEARED
925	T11 *TEST THAT CLOCK A STATUS REGISTER BIT 3 CAN BE SET AND CLEARED
971	T12 *TEST THAT CLOCK A STATUS REGISTER BIT 2 CAN BE SET AND CLEARED
1017	T13 *TEST THAT CLOCK A STATUS REGISTER BIT 1 CAN BE SET AND CLEARED
1063	T14 *TEST THAT CLOCK A STATUS REGISTER BIT 0 CAN BE SET AND CLEARED
1110	T15 *TEST THAT PATERN 125252 WILL SET AND CLEAR IN BUFFER REG.
1150	T16 *TEST THAT PATERN 052525 WILL SET AND CLEAR IN BUFFER REG.
1191	*
1192	* PHASE 2 ADVANCED BASIC LOGIC TESTS
1193	*
1196	T17 *TEST THE LOW BYTE OPERATION OF CLOCK'S STATUS REGISTER
1236	T20 *TEST THE HIGH BYTE OPERATION OF A'S STATUS REGISTER
1280	T21 *TEST CLOCK'S COUNT REGISTER WITH 125252 PATTERN
1329	T22 *TEST CLOCKS COUNTER REGISTER WITH 052525 PATTERN
1379	T23 *TEST THAT INIT CLEARS STATUS REGISTER
1427	T24 *TEST THAT INIT CLEARS BUFFER REGISTER
1464	T25 *TEST THE SETTING OF MAINTENANCE ST2 IN CLOCK BIT 15 TO SET
1514	T26 *TEST THAT ST1 FLAG SETS ON MAINTENANCE ST1
1544	T27 *TEST THAT BIT00 IN CLOCK STATUS REG. WILL SET WHEN BIT13 AND MAIN. ST2
1577	*
1578	*PHASE 3 COUNT TESTS
1579	*
1581	T30 *TEST TO SEE IF THE COUNTER WILL INCREMENT
1619	T31 *SEE IF CLOCK WILL COUNT UP FROM A ZERO BASE, RATE:ST1
1685	T32 *TEST THAT OVERFLOW (CSR BIT07) WILL SET ON OVERFLOW
1733	T33 *TEST THAT OVERFLOW WILL CLEAR THE GO BIT
1760	T34 *TEST THAT GO BIT DOES NOT CLEAR ON OVERFLOW, IF MODE 1
1788	T35 *TEST THE ABILITY OF CLOCK TO COUNT AT 1MHZ RATE
1843	T36 *TEST THE ABILITY OF CLOCK TO COUNT AT 100KHZ RATE
1890	T37 *TEST THE ABILITY OF CLOCK TO COUNT AT 10KHZ RATE
1953	T40 *TEST THE ABILITY OF CLOCK TO COUNT AT 1KHZ RATE
2008	T41 *TEST THE ABILITY OF CLOCK TO COUNT AT 100HZ RATE
2063	T42 *TEST THE ABILITY OF CLOCK TO COUNT AT LINEFREQ RATE
2122	T43 *TEST THAT COUNTER DOESN'T COUNT WHEN "SLAVE IN" RATE IS SELECTED
2172	T44 *TEST THAT THE CLOCK WILL COUNT IN MODE 1
2193	*

2200	*PHASE 4 CLOCK INTERRUPT TEST.
2201	*
2204	T45 *TEST THAT THE CLOCK WILL INTERRUPT ON CVERFLOW
2208	T46 *TEST THAT ST2 WILL CAUSE AN INTERRUPT
2209	T47 *TEST THAT ST1 WILL CAUSE AN INTERRUPT
2211	*
2215	*PHASE 5 ADVANCED TESTING
2216	*
2218	T50 *TEST THAT THE "FOR" BIT WILL SET ON 2 ST2'S
2219	T51 *TEST THAT THE "FOR" BIT WILL SET ON 2 ST1'S
2220	T52 *TEST THAT FOR BIT WILL SET ON TWO OVERFLOWS
2221	T53 *TEST THAT FOR BIT WILL CLEAR IF GO BIT IS SET
2222	T54 *TEST THAT WE CAN DISABLE THE INTERNAL OSC
2223	T55 *TEST THAT CLOCK CAN BE COUNTED USING MAINTENANCE OSC
2224	T56 *TEST THE CLOCK'S 1MHZ DIVIDER
2225	T57 *TEST THE CLOCK'S 100KHZ DIVIDER
2226	T60 *TEST THE CLOCK'S 10KHZ DIVIDER
2227	T61 *TEST THE CLOCK'S 1KHZ DIVIDER
2228	T62 *TEST THE CLOCK'S 100HZ DIVIDER
2229	T63 *TEST THE CLOCK'S MODE 2 OPERATION
2230	T64 *TEST THE CLOCK'S MODE 3 OPERATION
2231	T65 *DWARF TEST OF OVERFLOW OUT,ST2 IN AND OUT,AND ST1 IN
2232	T66 *DWARF TEST OF OVERFLOW OUT,ST1 IN AND OUT AND ST2 IN.
2233	T67 *IF ENABLED,CHECK THRESHOLD ST1 FROM TESTOR
2234	T70 *ST1,ST2 THRESHOLD TEST #2,POTS Ck
2235	T71 *ST1,ST2 THRESHOLD TEST #3,MID RANGE
2236	T72 *TEST CLOCK REPEATABILITY IF ON TESTOR
2237	T73 END OF TESTS
2238	END OF PASS ROUTINE
2239	:I/O SIGNAL TEST #1 ST1 IN AND ST2 OUT IN AND OUT
2240	:I/O SIGNAL TEST #2 CLOCK OVFLOW OUT TEST.
2241	:I/O SIGNAL TEST #3 ST1 OUT AND ST2 IN
2242	
2243	*SYSMAC ROUTINES
2244	
2245	BINARY TO OCTAL (ASCII) AND TYPE
2246	BINARY TO ASCII AND TYPE ROUTINE
2247	ERROR HANDLER ROUTINE
2248	ERROR MESSAGE TYPEOUT ROUTINE
2249	SCOPE HANDLER ROUTINE
2250	TTY INPUT ROUTINE
2251	TYPE ROUTINE
2252	APT COMMUNICATIONS ROUTINE
2253	POWER DOWN AND UP ROUTINES
2254	TRAP DEC0DER
2255	TRAP TABLE

```

.NLIST MC,MD,CND
.LIST ME
.ENABL ABS
.ENABL AMA
.MCALL .HEADER,.SETUP,.SETTRAP,.TRMTRP,.STRAP,.SRDOCT,.STYPBIN
.MCALL TYPOCS,.SPOWER,.SCATCH,.STYDOCT,.EQUAT,.SCMTAG,.SWFHI
.MCALL .SEUP,.SERROR,.SERRTYP,.STYPDEC,.SSCOPE,.SREAD,.STYPE
.MCALL .SACT11,.SAPTHOR,.SAPTYP
$SWR= 167400

```

167400

```

.TITLE MAINDEC-11-DVMNC-A
*COPYRIGHT (C) 1977
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*PROGRAM BY EDWARD C. BADGER
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
*
$TN=1

```

000001

```

.SBTTL OPERATIONAL SWITCH SETTINGS
*
* SWITCH USE
* -----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TYPEOUTS
* 11 INHIBIT ITERATIONS
* 10 BELL ON ERROR
* 9 LOOP ON ERROR
* 8 LOOP ON TEST IN SWR 7:0
*

```

.SBTTL TRAP CATCHER

000000

```

.=0
*ALL UNUSED LOCATIONS FROM 4-776 CONTAIN A ".+2"
*AND "JSR PC,R0" SEQUENCE TO CATCH ILLEGAL INTERRUPTS.
*AND INTERRUPTS TO THE WRONG VECTOR.
*LOCATION 0 CONTAINS A 0 TO CATCH IMPROPERLY LOADED
*VECTORS.

```

```

000004 000004 000200
020032
000174
000174 000000
000176 000000
000100

```

```

.=4
.WORD IOTRD,200 ;HANDLE BUSS ERROR.
.=174
DISPREG: .WORD 0 ;SOFTWARE DISPLAY REGISTER.
SWREG: .WORD 0 ;SOFTWARE SWITCH REGISTER.
.=100

```

Vertical text on the left margin, likely a page number or identifier, appearing as a series of characters.

```

55 000100 000104 000200 000002 .WORD 104,200,2 ;IF "B EVENT"ON Q-BUS IS
56 ;CONNECTED,WE NEED A WAY OF
57 ;IGNORING ITS INTERRUPTS.
58
59 000200 000137 001534 .=200
60 JMP @#START
61 000204 000137 002162 JMP @#RSTART
62
63 000230 000137 001514 .=230
64 JMP @#WSTART ;WESTFIELD STARTING ADDRESS
65 000240 000137 001474 .=240
66 JMP @#TSTSTR ;ALL TESTER TESTS
67 000250 000137 001454 .=250
68 JMP @#DWARFT ;TEST MNCKW WITH DWARF MODULE.
69 ;IF STARTED HERE.
70 ;ALLOWS PRODUCTIC , TO EXERCISE
71 ;UP TO 16 CLOCKS.NORMAL=4.

```

.SBTTL BASIC DEFINITIONS

```

72 001100 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
73 STACK= 1100
74 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
75 .EQUIV TOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL

```

;\*MISCELLANEOUS DEFINITIONS

```

76 000011 HT= 11 ;:CODE FOR HORIZONTAL TAB
77 000012 LF= 12 ;:CODE FOR LINE FEED
78 000015 CR= 15 ;:CODE FOR CARRIAGE RETURN
79 000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
80 177776 PS= 177776 ;:PROCESSOR STATUS WORD
81 .EQUIV PS,PSW
82 177774 STKLMT= 177774 ;:STACK LIMIT REGISTER
83 177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
84 177570 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
85 177570 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER

```

;\*GENERAL PURPOSE REGISTER DEFINITIONS

```

86 000000 R0= %0 ;:GENERAL REGISTER
87 000001 R1= %1 ;:GENERAL REGISTER
88 000002 R2= %2 ;:GENERAL REGISTER
89 000003 R3= %3 ;:GENERAL REGISTER
90 000004 R4= %4 ;:GENERAL REGISTER
91 000005 R5= %5 ;:GENERAL REGISTER
92 000006 R6= %6 ;:GENERAL REGISTER
93 000007 R7= %7 ;:GENERAL REGISTER
94 000006 SP= %6 ;:STACK POINTER
95 000007 PC= %7 ;:PROGRAM COUNTER

```

;\*PRIORITY LEVEL DEFINITIONS

```

96 000000 PRO= 0 ;:PRIORITY LEVEL 0
97 000040 PR1= 40 ;:PRIORITY LEVEL 1
98 000100 PR2= 100 ;:PRIORITY LEVEL 2
99 000140 PR3= 140 ;:PRIORITY LEVEL 3

```

109 000200  
110 000240  
111 000300  
112 000340

PR4= 200 :: PRIORITY LEVEL 4  
PR5= 240 :: PRIORITY LEVEL 5  
PR6= 300 :: PRIORITY LEVEL 6  
PR7= 340 :: PRIORITY LEVEL 7

:"SWITCH REGISTER" SWITCH DEFINITIONS

113 100000  
114 040000  
115 020000  
116 010000  
117 004000  
118 002000  
119 001000  
120 000400  
121 000200  
122 000100  
123 000040  
124 000020  
125 000010  
126 000004  
127 000002  
128 000001

SW15= 100000  
SW14= 40000  
SW13= 20000  
SW12= 10000  
SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1

.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

:"DATA BIT DEFINITIONS (BIT00 TO BIT15

129 100000  
130 040000  
131 020000  
132 010000  
133 004000  
134 002000  
135 001000  
136 000400  
137 000200  
138 000100  
139 000040  
140 000020  
141 000010  
142 000004  
143 000002  
144 000001

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1

.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6



163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000

000004  
000010  
000014  
000014  
000014  
000020  
000024  
000030  
000034  
000060  
000064  
000240  
  
170420  
000440  
000200  
  
167400  
000001  
  
000254  
000046  
014664  
000052  
000000  
000254  
001000  
  
001000  
000024  
000200  
000044  
001000  
001000

```
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

;*BASIC "CFU" TRAP VECTOR ADDRESSES
ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC=14        ;; "T" BIT
TRIVEC= 14        ;; TRACE TRAP
BPTVEC= 14        ;; BREAKPOINT TRAP (BPT)
IOTVEC= 20        ;; INPUT/OUTPUT TRAP (IOT) **SCOFF**
PWRVEC= 24        ;; POWER FAIL
EMTVEC= 30        ;; EMULATOR TRAP (EMT) **ERROR**
TRAPVEC=34        ;; "TRAP" TRAP
TKVEC= 60         ;; TTY KEYBOARD VECTOR
TPVEC= 64         ;; TTY PRINTER VECTOR
PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR

ABASE= 170420
AVECT1= 440
APRIOR= 200

$SWR= 167400
$TN= 1

.SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
$SVPC= .           ;SAVE PC
.=46              ;;1)SET LOC.46 TO ADDRESS OF SENDAC IN .SEOp
$ENDAD           ;;
.=52              ;;2)SET LOC.52 TO ZERO
.WORD 0           ;;
.= $SVPC          ;; RESTORE PC
.=1000

.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.$X= .           ;;SAVE CURRENT LOCATION
.=24            ;;SET POWER FAIL TO pPOINT TO START OF PROGRAM
200            ;;FOR APT START UP
.=44            ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR        ;;POINT TO APT HEADER BLOCK
.=.$X          ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PCP11 DIAGNOSTIC
;INTERFACE SPEC.
```

217	001000		\$APTHD:		
218	001000	000000	\$HIBTS:	.WORD	0
219	001002	001174	\$MBADR:	.WORD	\$MAIL
220	001004	000002	\$STSM:	.WORD	2
221	001006	000170	\$PASTM:	.WORD	120.
222	001010	000170	\$UNITM:	.WORD	120.
223	001012	000031			

: : TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
 : : ADDRESS OF APT MAILBOX (BITS 0-15)  
 : : RUN TIME OF LONGEST TEST  
 : : RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
 : : ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
 \$ETEND-\$MAIL\*2 ; ; LENGTH MAILBOX-ETABLE (WORDS)

```

224
225
226
227
228
229
230
231 001100
232 001100 000000
233 001102 000
234 001103 000
235 001104 000000
236 001106 000000
237 001110 000000
238 001112 000000
239 001114 000
240 001115 001
241 001116 000000
242 001120 000000
243 001122 000000
244 001124 000000
245 001126 000000
246 001130 000000
247 001132 000000
248 001134 000
249 001135 000
250 001136 000000
251 001140 177570
252 001142 177570
253 001144 177560
254 001146 177562
255 001150 177564
256 001152 177566
257 001154 000
258 001155 002
259 001156 012
260 001157 000
261 001160 000000
262 001162 000000
263 001164 177607 000377
264 001170 077
265 001171 015
266 001172 000012
267
268
269
270
271
272 001174
273 001174 000000
274 001176 000000
275 001200 000000
276 001202 000000
277 001204 000000

```

.SBTTL COMMON TAGS

```

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

```

```

SCMTAG:      =1100
              .WORD      0
              .BYTE      0
SERFLG:      .BYTE      0
SICNT:       .WORD      0
SLPADR:      .WORD      0
SLPERR:      .WORD      0
SERTTL:      .WORD      0
SITEMB:      .BYTE      0
SERMAX:      .BYTE      1
SERRPC:      .WORD      0
SGDADR:      .WORD      0
SBCADR:      .WORD      0
SGODAT:      .WORD      0
SBODAT:      .WORD      0
              .WORD      0
              .WORD      0
SAUTOB:      .BYTE      0
SINTAG:      .BYTE      0
              .WORD      0
SWR:         .WORD      DSWR
DISPLAY:     .WORD      DDI:P
STKS:        177560
STKB:        177562
STPS:        177564
STPB:        177566
SNUL:        .BYTE      0
SFILLS:      .BYTE      2
SFILLC:      .BYTE      12
STPFLG:      .BYTE      0
STIMES:      0
SESCAPE:     0
SBELL:       .ASCIZ    <207><377><377>
SQUES:       .ASCII    /?/
SCRLF:       .ASCII    <15>
SLF:         .ASCIZ    <12>

```

```

;; START OF COMMON TAGS
;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL E TE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED
;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR
;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0='ES.
;; MAX. NUMBER OF ITERATIONS
;; ESCAPE ON ERROR ADDRESS
;; CODE FOR BELL
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

```

.SBTTL APT MAILBOX-ETABLE

```

*****
EVEN
$MAIL:
$MSGTY:      .WORD      AMSGTY
$FATAL:      .WORD      AFATAL
$TESTN:      .WORD      ATESTN
$PASS:       .WORD      APASS
$DEVCT:      .WORD      ADEVCT

```

```

;; APT MAILBOX
;; MESSAGE TYPE CODE
;; FATAL ERROR NUMBER
;; TEST NUMBER
;; PASS COUNT
;; DEVICE COUNT

```

278 001206 000000  
 279 001210 000000  
 280 001212 000000  
 281 001214 000  
 282 001214 000  
 283 001215 000  
 284 001216 000000  
 285 001220 000000  
 286 001222 000000  
 287  
 288  
 289  
 290  
 291  
 292  
 293 001224 000  
 294 001225 000  
 295  
 296  
 297  
 298  
 299 001226 000000  
 300  
 301 001230 000  
 302 001231 000  
 303 001232 000000  
 304 001234 000  
 305 001235 000  
 306 001236 000000  
 307 001240 000  
 308 001241 000  
 309 001242 000000  
 310 001244 000440  
 311 001246 000000  
 312 001250 170420  
 313 001252 000000  
 314 001254 000000  
 315 001256  
 316

\$UNIT: .WORD AUNIT ;: I/O UNIT NUMBER  
 \$MSGAD: .WORD AMSGAD ;: MESSAGE ADDRESS  
 \$MSGLG: .WORD AMSGLG ;: MESSAGE LENGTH  
 \$ETABLE: ;: APT ENVIRONMENT TABLE  
 \$ENV: .BYTE AENV ;: ENVIRONMENT BYTE  
 \$ENVM: .BYTE AENVM ;: ENVIRONMENT MODE BITS  
 \$SWREG: .WORD ASWREG ;: APT SWITCH REGISTER  
 \$USWR: .WORD AUSWR ;: USER SWITCHES  
 \$CPUOP: .WORD ACPUOP ;: CPU TYPE, OPTIONS  
 ;: BITS 15-11=CPU TYPE  
 ;: 11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05  
 ;: 11/70=06, PDQ=07, Q=10  
 ;: BIT 10=REAL TIME CLOCK  
 ;: BIT 9=FLOATING POINT PROCESSOR  
 ;: BIT 8=MEMORY MANAGEMENT  
 \$MAMS1: .BYTE AMAMS1 ;: HIGH ADDRESS, M.S. BYTE  
 \$MTYP1: .BYTE AMTYP1 ;: MEM. TYPE, BLK#1  
 ;: MEM. TYPE BYTE -- (HIGH BYTE)  
 ;: 900 NSEC CORE=001  
 ;: 300 NSEC BIPOLAR=002  
 ;: 500 NSEC MOS=003  
 \$MADR1: .WORD AMADR1 ;: HIGH ADDRESS, BLK#1  
 ;: MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE  
 \$MAMS2: .BYTE AMAMS2 ;: HIGH ADDRESS, M.S. BYTE  
 \$MTYP2: .BYTE AMTYP2 ;: MEM. TYPE, BLK#2  
 \$MADR2: .WORD AMADR2 ;: MEM. LAST ADDRESS, BLK#2  
 \$MAMS3: .BYTE AMAMS3 ;: HIGH ADDRESS, M.S. BYTE  
 \$MTYP3: .BYTE AMTYP3 ;: MEM. TYPE, BLK#3  
 \$MADR3: .WORD AMADR3 ;: MEM. LAST ADDRESS, BLK#3  
 \$MAMS4: .BYTE AMAMS4 ;: HIGH ADDRESS, M.S. BYTE  
 \$MTYP4: .BYTE AMTYP4 ;: MEM. TYPE, BLK#4  
 \$MADR4: .WORD AMADR4 ;: MEM. LAST ADDRESS, BLK#4  
 \$VECT1: .WORD AVECT1 ;: INTERRUPT VECTOR#1, BUS PRIORITY#1  
 \$VECT2: .WORD AVECT2 ;: INTERRUPT VECTOR#2, BUS PRIORITY#2  
 \$BASE: .WORD ABASE ;: BASE ADDRESS OF EQUIPMENT UNDER TEST  
 \$DEVM: .WORD ADEVM ;: DEVICE MAP  
 \$CDW1: .WORD ACDW1 ;: CONTROLLER DESCRIPTION WORD#1  
 \$ETEND:  
 .MEXIT

317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370

.SBTTL ERROR POINTER TABLE

: \*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
 : \*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
 : \*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
 : \*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
 : \*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

: \* EM : POINTS TO THE ERROR MESSAGE  
 : \* DH : POINTS TO THE DATA HEADER  
 : \* DT : POINTS TO THE DATA  
 : \* DF : POINTS TO THE DATA FORMAT

001256

\$ERRTB:

: ITEM 1

001256 020272  
 001260 020615  
 001262 021142  
 001264 021236

EM1  
 DH1  
 DT1  
 DF0

: CLOCK SR FUNCTION ERROR  
 : ERRPC ASR WAS S/B  
 : \$ERRPC, ASR, \$BDDAT, \$GDDAT  
 : ALL NUMBERS ARE IN OCTAL FORM

: ITEM 2

001266 020324  
 001270 020615  
 001272 021142  
 001274 021236

EM2  
 DH1  
 DT1  
 DF0

: CLOCK SR DATA ERROR  
 : ERRPC ASR WAS S/B  
 : \$ERRPC, ASR, \$BDDAT, \$GDDAT  
 : ALL NUMBERS ARE IN OCTAL FORM

: ITEM 3

001276 020352  
 001300 020641  
 001302 021154  
 001304 021236

EM3  
 DH3  
 DT3  
 DF0

: CLOCK BR DATA ERROR  
 : ERRPC ABR WAS  
 : \$ERRPC, ABR, \$BDDAT, \$GDDAT  
 : ALL NUMBERS ARE IN OCTAL FORM

: ITEM 4

001306 020400  
 001310 020665  
 001312 021166  
 001314 021236

EM4  
 DH4A  
 DT4  
 DF0

: INTERRUPT ERROR.  
 : ERRPC TO ROM ADDR.  
 : \$ERRPC, TRTO, TRFRO  
 : ALL NUMBERS ARE IN OCTAL FORM

: ITEM 5

001316 020421  
 001320 020615  
 001322 021142

EM5  
 DH1  
 DT1

: CLOCK COUNT REG ERROR  
 : ERRPC ASR WAS S/B  
 : \$ERRPC, ACR, \$BDDAT, \$GDDAT

371	001324	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
372					
373				: ITEM 6	
374					
375					
376	001326	020463	EM12		: CLOCK COUNT FUNCTION ERROR
377	001330	020721	DH12		: ERRPC ASR
378	001332	021176	DT12		: ERRPC ASR
379	001334	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
380					
381				: ITEM 7	
382					
383					
384	001336	020512	EM16		: CLOCK INTERRUPT ERROR
385	001340	020721	DH12		: ERRPC ASR
386	001342	021176	DT12		: SERRPC ASR
387	001344	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
388					
389					
390				: ITEM 10	
391					
392	001346	020543	EM20		: CLOCK REPEATABILITY ERROR
393	001350	020736	DH20		: ERROR ASR 2ND CNT 1ST CNT 3RD CNT
394	001352	021204	DT20		: SERRPC ASR, SBDDAT, SGDDAT, STMPC
395	001354	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
396					
397					
398				: ITEM 11	
399					
400	001356	020444	EM11		: CLOCK COUNT ERROR
401	001360	020615	DH1		: ERRPC ASR WAS S/B
402	001362	021216	DT22		: SERRPC ASR, SBDDAT, STMPC
403	001364	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
404					
405					
406				: ITEM 12	
407					
408	001366	020572	EM26		: CLOCK ADDRESSING ERROR
409	001370	020771	DH26		: ERRPC CLOCK ADDR.
410	001372	021230	DT26		: SERRPC, STMPC
411	001374	021236	DF0		: ALL NUMBERS ARE IN OCTAL FORM
412					
413					
414	001376	170420	ASR:	.WORD	ABASE
415	001400	170422	ABR:	.WORD	ABASE+2
416	001402	000440	VECT1:	.WORD	AVECT1
417	001404	000442	VECTP:	.WORD	AVECT1+2
418	001406	000444	VECT2:	.WORD	AVECT1+4
419	001410	000446	VECT2P:	.WORD	AVECT1+6
420	001412	000200	PRIOR:	.WORD	APRIOR
421					
422	001414	167774	DR:	.WORD	167774
423	001416	167772	DR2:	.WORD	167772
424	001420	170430	TSCLC:	.WORD	170430
					: ADR. OF TESTOR CLOCK

425	001422	170432			TSCLD: .WORD	170432		: BUFFER PRESET REG.
426	001424	000000			STMP0: .WORD	0		: TEMP STORAGE.
427	001426	000000			STMP1: .WORD	0		: TMP STORAGE.
428	001430	000000			STMP3: .WORD	0		
429	001432	000000			ROTATE: .WORD	0		: POINT TO DEVICE UNDER TEST.
430	001434	000000			UTEST: .WORD	0		: KEEPS TRACK OF GOOD UNITS.
431	001436	000000			ERCNT: .WORD	0		: COUNTS ERRORS.
432	001440	000000			MDEVCT: .WORD	0		: COUNTS DEVICES TESTED.
433	001442	000000			TSTCNT: .WORD	0		: MAX DEVICES TO BE TESTED.
434	001444	000000			EXS: .WORD	0		: =0, NORMAL: =1 SPECIAL TESTOR START, BY L+S 2 2
435	001446	000000			LCNT: .WORD	0		: TOTAL UNITS TESTED.
436	001450	000000			DWARF: .WORD	0		: INDICATE IF DWARF MODULE PRESENT (=1, YES.
437	001452	000000			ASK: .WORD	0		: =1 WHEN QUESTION ASKED IN RUN.
438								
439								
440		001454			DWARF=.			
441	001454	005237	001450		INC	DWARF		: INDICATE DWARF TESTS.
442	001460	005237	001444		INC	EXS		: PLUS REST OF TESTS.
443	001464	012737	000004	001442	MOV	#4, TSTCNT		: MAX TO BE TESTED.
444	001472	000427			BR	IS		
445		001474			TSTSTR=.			
446	001474	005237	001444		INC	EXS		: SET FOR TESTOR.
447	001500	005037	001450		CLR	DWARF		
448	001504	012737	000020	001442	MOV	#16., TSTCNT		: ALLOW 16 UNITS
449	001512	000417			BR	IS		
450		001514			WSTART=.			
451	001514	012737	000020	001442	MOV	#16., TSTCNT		: TEST UP TO 16 UNITS.
452	001522	005037	001450		CLR	DWARF		
453	001526	005037	001444		CLR	EXS		
454	001532	000407			BR	IS		
455		001534			START=.			
456	001534	012737	000004	001442	MOV	#4, TSTCNT		: TEST UP TO FOUR UNITS.
457	001542	005037	001450		CLR	DWARF		
458	001546	005037	001444		CLR	EXS		
459	001552				IS:			
460					.SBTTL	INITIALIZE THE COMMON TAGS		
461					:: CLEAR	THE COMMON TAGS (\$CMTAG) AREA		
462	001552	012706	001100		MOV	#SCMTAG, R6		: FIRST LOCATION TO BE CLEARED
463	001556	005026			CLR	(R6)+		: CLEAR MEMORY LOCATION
464	001560	022706	001140		CMP	#SWR, R6 ;: DONE?		
465	001564	001374			BNE	-6		: LOOP BACK IF NO
466	001566	012706	001100		MOV	#STACK, SP		: SETUP THE STACK POINTER
467					:: INITIALIZE A FEW VECTORS			
468	001572	012737	016066	000020	MOV	#SCOPE, @#IOTVEC		: IOT VECTOR FOR SCOPE ROUTINE
469	001600	012737	000340	000022	MOV	#340, @#IOTVEC+2		: LEVEL 7
470	001606	012737	015524	000030	MOV	#ERROR, @#EMTVEC		: EMT VECTOR FOR ERROR ROUTINE
471	001614	012737	000340	000032	MOV	#340, @#EMTVEC+2		: LEVEL 7
472	001622	012737	020212	000034	MOV	#STRAP, @#TRAPVEC		: TRAP VECTOR FOR TRAP CALLS
473	001630	012737	000340	000036	MOV	#340, @#TRAPVEC+2		: LEVEL 7
474	001636	012737	017654	000024	MOV	#SPWRDN, @#PWRVEC		: POWER FAILURE VECTOR
475	001644	012737	000340	000026	MOV	#340, @#PWRVEC+2		: LEVEL 7
476	001652	005037	001160		CLR	\$TIMES		: INITIALIZE NUMBER OF ITERATIONS
477	001656	005037	001162		CLR	\$ESCAPE		: CLEAR THE ESCAPE ON ERROR ADDRESS
478	001662	112737	000001	001115	MOVB	#1, \$ERMAX		: ALLOW ONE ERROR PER TEST



```

479 001670 012737 001670 001106      MOV      #,$LPADR      ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
480 001676 012737 001676 001110      MOV      #,$LPERR      ;; SETUP THE ERROR LOOP ADDRESS
481                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
482                                     ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
483 001704 013746 000004                MOV      @ERRVEC-(SP)  ;; SAVE ERROR VECTOR
484 001710 012737 001744 000004      MOV      #64$, @ERRVEC ;; SET UP ERROR VECTOR
485 001716 012737 177570 001140      MOV      #DSWR, SWR    ;; SETUP FOR A HARDWARE SWICH REGISTER
486 001724 012737 177570 001142      MOV      #DDISP, DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
487 001732 022777 177777 177200      CMP      #-1, @SWR    ;; TRY TO REFERENCE HARDWARE SWR
488 001740 001012                BNE      66$          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
489                                     ;; AND THE HARDWARE SWR IS NOT = -1
490 001742 000403                BR       65$          ;; BRANCH IF NO TIMEOUT
491 001744 012716 001752                64$:  MOV      #65$, (SP) ;; SET UP FOR TRAP RETURN
492 001750 000002                RTI
493 001752 012737 000176 001140      65$:  MOV      #SWREG, SWR   ;; POINT TO SOFTWARE SWR
494 001760 012737 000174 001142      MOV      #DISPREG, DISPLAY
495 001766 012637 000004      66$:  MOV      (SP)+, @ERRVEC ;; RESTORE ERROR VECTOR
496
497 001772 005037 001202                CLR      $PASS        ;; CLEAR PASS COUNT
498 001776 132737 000200 001215      BITB    #APTSIZE, $ENVM ;; TEST USER SIZE UNDER APT
499 002004 001403                BEQ      67$          ;; YES, USE NON-APT SWITCH
500 002006 012737 001216 001140      MOV      #SSWREG, SWR ;; NO, USE APT SWITCH REGISTER
501 002014                67$:
502
503
504 002014 012746 000340                MOV      #340, -(SP)   ;; SET CPU PRIORITY ON RETURN.
505 002020 012746 002026                MOV      #68$, -(SP)   ;; SHOW RETURN ADDRESS.
506 002024 000002                RTI                    ;; CAUSE A RETURN PUTS STATUS IN STATUS REG...
507 002026                68$:
508
509 002026 005037 001204                CLR      $DEVCT        ;; ZERO DEVICE COUNT.
510 002032 012737 020032 000004      MOV      #IOTRD, @ERRVEC ;; FIX TRAP CATCHER.
511 002040 012737 000340 000006      MOV      #340, @ERRVEC+2
512 002046 013737 001244 001402      MOV      $VECT1, VECT1 ;; NOW FIX VECTOR ADDR.
513 002054 013737 001250 001376      MOV      $BASE, ASR    ;; FIX ADDRESS OF CSR.
514
515 002062 005737 000042                TST      @#42          ;; RUNNIGN UNDER APT, XXDP?
516 002066 001035                BNE      RSTART
517 002070 005227 177777                INC      #-1          ;; ONLY TYPE ON 1ST START JP
518 002074 001032                BNE      RSTART
519 002076 104401 021015                TYPE,   WRNM1
520
521 .SBTTL TYPE PROGRAM NAME
522 ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
523 INC      #-1          ;; FIRST TIME
524 BNE      69$          ;; BRANCH IF NO
525 TYPE    , 70$        ;; TYPE ASCIZ STRING
526 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
527 TST      @#42          ;; ARE WE RUNNING UNDER XXDP ACT?
528 BNE      71$          ;; BRANCH IF YES
529 CMPB    $ENV, #1     ;; ARE WE RUNNING UNDER APT?
530 BEQ      71$          ;; BRANCH IF YES
531 CMP     SWR, #SWREG  ;; SOFTWARE SWITCH REG SELECTED?
532 BNE      72$          ;; BRANCH IF NO
                    GTSWP ;; GET SOFT-SWR SETTINGS

```

E03

```

533 002144 000403          BR      725
534 002146 112737 000001 001134 715:  MOVB  #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
535 002154          725:
536 002154 000402          BR      695      ;;GET OVER THE ASCIZ
537          ;;705:  .ASCIZ  <CRLF>##<CRLF>
538 002162          695:
539 002162          RSTART:
540 002162 005037 001452          CLR      ASK
541 002166 005737 001444          TST     EXS      ;;TESTOR MODE ENABLED^^
542 002172 001417          BEQ     15       ;;NO DON'T TYPE NEXT MESSAGE.
543 002174 104401 002202          TYPE   655      ;;TYPE ASCIZ STRING
544 002200 000414          BR      645      ;;GET OVER THE ASCIZ
545          ;;655:  .ASCIZ  <15><12>#TESTOR MODE ENABLED--#
546 002232          645:
547 002232          15:
548 002232 104401 002240          TYPE   675      ;;TYPE ASCIZ STRING
549 002236 000411          BR      665      ;;GET OVER THE ASCIZ
550          ;;675:  .ASCIZ  <15><12>#TEST RUNNING...#
551 002262          665:
552 002262 005037 001440          CLR      MDEVCT  ;;TESTING FIRST UNIT.
553 002266 005037 001436          CLR      ERCNT   ;;NO ERRORS.
554 002272 005037 001202          CLR      $PASS   ;;NO PASSES.
555 002276 012737 000001 001432          MOV     #1,ROTATE ;;POINT TO FIRST UNIT.
556 002304 013737 001432 001434          MOV     ROTATE,UTEST
557 002312          LOOP:
558          ;;COME HERE FOR NEXT UNIT,OR ENC PASS.
559 002312 042737 170000 001402          BIC     #170000,VECT1 ;;CLEAR OUT PRIORITY BITS.
560 002320 013737 001402 001404          MOV     VECT1,VECTP ;;NOW FIX VECTOR +2 ADDR.
561 002326 062737 000002 001404          ADD     #2,VECTP
562 002334 013737 001402 001405          MOV     VECT1,VECT2 ;;LETS FIX ST2 VECTOR ADDR.
563 002342 062737 000004 001406          ADD     #4,VECT2  ;;ITS 4 GREATER THEN THE 1ST.
564 002350 013737 001406 001410          MOV     VECT2,VECT2P ;;VECTOR +2 ADDR.
565 002356 062737 000002 001410          ADD     #2,VECT2P
566
567
568 002364 013737 001376 001400          MOV     ASR,ABR   ;;FIX ADDR OF PRESET REG=
569 002372 062737 000002 001400          ADD     #2,ABR   ;;CSR + 2
570
571
572          ;;*****
573          ;;*TEST 1 *TEST THE ADDRESSABILITY OF CLOCK CSR
574          ;;*****
575          ;;ST1:
576 002400 000240          NOP
577 002402 012737 000050 001160          MOV     #50,$TIMES ;;DO 50 ITERATIONS
578 002410 012737 002440 001106          MOV     #15,$LPADR ;;SET SCOPE LOOP ADDRESS
579 002416 112737 000001 001102          MOVB   #1,$STNM  ;;SET TEST #1.
580 002424 112737 000001 001200          MOVB   #1,$TESTN ;;DON'T FORGET APT!
581 002432 012737 002440 001110          MOV     #15,$LPERR
582
583
584 002440 013746 000004          15:  MOV     @#ERRVEC, -(SP) ;;SAVE CONTENTS OF ADDRS 6.
585 002444 012737 002460 000004          MOV     #25,@#ERRVEC ;;SET TIME-OUT TRAP VECTOR TO HANDLER IN CASE.
586          ;;WE TIME-OUT WHEN ADDRESSING THE HW11.
  
```





648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
  
669  
670  
671  
  
674  
675  
676  
677  
678  
679  
680  
681  
  
684  
685  
686  
687  
  
690  
691  
692

002542 000004  
002544 012737 000100 001160  
  
002552 005077 176620  
002556 052777 040000 176612  
002564 012737 040000 001124  
002572 017737 176600 001126  
002600 023737 001124 001126  
002606 001402  
  
002610 104002  
  
002612 000412  
002614 042777 040000 176554  
002622 005037 001124  
002626 017737 176544 001126  
002634 001401  
  
002636 104002  
  
002640

```
*****
: *TEST 3 *TEST THAT CLOCK A STATUS REGISTER BIT 14 CAN BE SET AND CLEARED
: *CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
: *F/FS OR GATES
: *
: *****
: ST3: SCOPE
: MOV #100,$TIMES ;;DO 100 ITEPATIONS
: CLR JASR ;/CLEAR THE STATUS REGISTER.
: BIS #BIT14,JASR ;/SET BIT 14.
: MOV #BIT14,$GDDAT ;/SET FOR ERROR TYPEOUT S/B.
: MOV JASR,$BDDAT ;/READ THE STATUS REGISTER.
: CMP $GDDAT,$BDDAT ;/DID BIT 14 AND ONLY BIT 14 SET?
: BEQ 15 ;/IF SO-LETS TRY CLEARING IT.
: *****
: : *****
: ERROR <<< *****
: ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
: ;/BIT 14 FAILED TO BIT SET.
: : *****
: ERROR <<< *****
: BR 25 ;/BR TO END SUBTEST.
: IS: BIC #BIT14,JASR ;/TRY CLEARING BIT 14.
: CLR $GDDAT ;/CLEAR S/B FOR TYPEOUT IF ANY.
: MOV JASR,$BDDAT ;/NOW READ IT BACK.
: BEQ 25 ;/IF ZERO - NO ERROR!
: : *****
: ERROR <<< *****
: ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
: ;/BIT 14 FAILED TO CLEAR.
: : *****
: ERROR <<< *****
: 25:
```

```

693
694
695
696
697
698
699
700
701
702 002640 000004
703 002642 012737 000100 001160
704
705 002650 005077 176522
706 002654 052777 020000 176514
707 002662 012737 020000 001124
708 002670 017737 176502 001126
709 002676 023737 001124 001126
710 002704 001402
711
714 002706 104002
715
716
719 002710 000412
720
721 002712 042777 020000 176456 13:
722 002720 005037 001124
723 002724 017737 176446 001126
724 002732 001401
725
726
729 002734 104002
730
731
732
735 002736
736
737

```

```

:*****i/*****
:TEST 4 *TEST THAT CLOCK A STATUS REGISTER BIT 13 CAN BE SET AND CLEARED
:
:CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
:F/FS OR GATES
:
:*****
134: SCOPE
MOV #100,STIMES ;:DO 100 ITERATIONS
CLR #ASR ;/CLEAR THE STATUS REGISTER.
BIS #BIT13,ASR ;/SET BIT 13.
MOV #BIT13,$GDDAT ;/SET FOR ERROR TIMEOUT S/B.
MOV #ASR,$BDDAT ;/READ THE STATUS REGISTER.
CMP $GDDAT,$BDDAT ;/DID BIT 13 AND ONLY BIT 13 SET?
BEQ 15 ;/IF SO-LETS TRY CLEARING IT.
:
:*****
:ERROR <<<*****>>>
ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
;/BIT 13 FAILED TO BIT SET.
:
:*****
:ERROR <<<*****>>>
BR 25 ;/BR TO #ND SUBTEST.
13: BIC #BIT13,ASR ;/TRY CLEARING BIT 13.
CLR $GDDAT ;/CLEAR S/B FOR TIMEOUT IF ANY.
MOV #ASR,$BDDAT ;/NOW READ IT BACK.
BEQ 25 ;/IF ZERO - NO ERROR!
:
:*****
:ERROR <<<*****>>>
ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
;/BIT 13 FAILED TO CLEAR.
:
:*****
:ERROR <<<*****>>>
25:

```

J03

MAINDEC-11-DVMNC-A  
DVMNCA.P11 T4

MACY11 27 6541 27-DEC-77 09:32 PAGE 17  
\*TEST THAT CLOCK A STATUS REGISTER BIT 13 CAN BE SET AND CLEARED

SEG 0035

738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
  
759  
760  
761  
  
764  
765  
766  
767  
768  
769  
770  
771  
  
774  
775  
776  
777  
778  
779  
780  
781  
782

\*\*\*\*\*  
: \*TEST 5 \*TEST THAT CLOCK A STATUS REGISTER BIT 11 CAN BE SET AND CLEARED  
: \*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL  
: \*FFS OR GATES  
: \*

\*\*\*\*\*  
\*STS: SCOPE #100,\$TIMES ;;DO 100 ITERATIONS  
MOV #BIT11,\$ASR ;/CLEAR THE STATUS REGISTER.  
CLR \$ASR ;/SET BIT 11.  
BIS #BIT11,\$ASR ;/SET FOR ERROR TIMEOUT S/B.  
MOV #BIT11,\$GDDAT ;/READ THE STATUS REGISTER.  
MOV \$ASR,\$BDDAT ;/DID BIT 11 AND ONLY BIT 11 SET?  
CMP \$GDDAT,\$BDDAT ;/IF SO-LETS TRY CLEARING IT.  
BEQ 15

:: \*\*\*\*\* >> ERROR <<< \*\*\*\*\*

ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER  
;/BIT 11 FAILED TO BIT SET.

:: \*\*\*\*\* >> ERROR <<< \*\*\*\*\*

BR 25 ;/BR TO END SUBTEST.  
15: BIC #BIT11,\$ASR ;/TRY CLEARING BIT 11.  
CLR \$GDDAT ;/CLEAR S/B FOR TIMEOUT IF ANY.  
MOV \$ASR,\$BDDAT ;/NOW READ IT BACK.  
BEQ 25 ;/IF ZERO - NO ERROR!

:: \*\*\*\*\* >> ERROR <<< \*\*\*\*\*

ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.  
;/BIT 11 FAILED TO CLEAR.

:: \*\*\*\*\* >> ERROR << \*\*\*\*\*

25:

K03

MAINDEC-11-DVMNC-A  
DVMNCA.P11 TS

MACY11 27 654 27-DEC-77 09:32 PAGE 18  
\*TEST THAT CLOCK A STATUS REGISTER BIT 11 CAN BE SET AND CLEARED

FE2 DC36

```

783
784
785
786
787
788
789
790
791
792
793 003034 000004
794 003036 012737 000100 001160
795
796 003044 005077 176326
797 003050 052777 000100 176320
798 003056 012737 000100 001124
799 003064 017737 176306 001126
800 003072 023737 001124 001126
801 003100 001402
804 003102 10400
805
806
809 003104 000412
810
811 003106 042777 000100 176262
812 003114 005037 001124
813 003120 017737 176252 001126
814 003126 001401
815
816
819 003130 104002
820
821
822
825 003132
826
827

```

```

*****
: /#
: *TEST 6 *TEST THAT CLOCK A STATUS REGISTER BIT 6 CAN BE SET AND CLEARED
: *
: *CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
: *F/FS OR GATES
: *
*****
↑ST6: SCOPE
MOV #100,STIMES ;;DO 100 ITERATIONS
CLR JASR ;/CLEAR THE STATUS REGISTER.
BIS #BIT6,JASR ;/SET BIT 6.
MOV #BIT6,$GDDAT ;/SET FOR ERROR TIMEOUT S/B.
MOV JASR,$BDDAT ;/READ THE STATUS REGISTER.
CMP $GDDAT,$BDDAT ;/DID BIT 6 AND ONLY BIT 6 SET?
BEQ IS ;/IF SO-LETS TRY CLEARING IT.

:: ***** >>> ERROR <<< *****
ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
;/BIT 6 FAILED TO BIT SET.

:: ***** >>> ERROR <<< *****
BR 25 ;/BR TO END SUBTEST.

IS: BIC #BIT6,JASR ;/TRY CLEARING BIT 6.
CLR $GDDAT ;/CLEAR S/B FOR TIMEOUT IF ANY.
MOV JASR,$BDDAT ;/NOW READ IT BACK.
BEQ 25 ;/IF ZERO - NO ERROR!

:: ***** >>> ERROR <<< *****
ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
;/BIT 6 FAILED TO CLEAR.

:: ***** >>> ERROR <<< *****
25:

```



L03

MAINDEC-11-DVMNC-A  
DVMNCA.P11 T6

MACY11 27 654 27-DEC-77 09:32 PAGE 19  
\*TEST THAT CLOCK A STATUS REGISTER BIT 6 CAN BE SET AND CLEARED

SEQ 0037

```

828 ;/8
829 :*****
830 :*TEST 7 *TEST THAT CLOCK A STATUS REGISTER BIT 5 CAN BE SET AND CLEARED
831 :*
832 :*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
833 :*F'S OR GATES
834 :*
835 :*****
836
837 003132 000004
838 003134 012737 000100 001160
839
840 003142 005077 176230 CLR JASR ;/CLEAR THE STATUS REGISTER.
841 003146 052777 000040 176222 BIS #BITS, JASR ;/SET BIT 5.
842 003154 012737 000040 001124 MOV #BITS, $GDDAT ;/SET FOR ERROR TIMEOUT S/B.
843 003162 017737 176210 001126 MOV JASR, $BDDAT ;/READ THE STATUS REGISTER.
844 003170 023737 001124 001126 CMP $GDDAT, $BDDAT ;/DID BIT 5 AND ONLY BIT 5 SET?
845 003176 001402 BEQ IS ;/IF SO-LETS TRY CLEARING IT.
846
::*****
849 003200 104002
850 ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
851 ;/BIT 5 FAILED TO BIT SET.
::*****
854 003202 000412 BR 25 ;/BR TO END SUBTEST.
855
856 003204 042777 000040 176164 15: BIC #BITS, JASR ;/TRY CLEARING BIT 5.
857 003212 005037 001124 CLR $GDDAT ;/CLEAR S/B FOR TIMEOUT IF ANY.
858 003216 017737 176154 001126 MOV JASR, $BDDAT ;/NOW READ IT BACK.
859 003224 001401 BEQ 25 ;/IF ZERO - NO ERROR!
860
::*****
864 003226 104002
865 ERROR 2 ; ERROR - CLOCK A STATUS REGISTER.
866 ;/BIT 5 FAILED TO CLEAR.
867
::*****
870 00323C 25:
871
872

```

```

873
874
875
876
877
878
879
880
881
882 003230 000004
883 003232 012737 000100 001160
884
885 003240 005077 176132
886 003244 052777 000020 176124
887 003252 012737 000020 001124
888 003260 017737 176112 001126
889 003266 023737 001124 001126
890 003274 001402
891
894 003276 104002
895
896
899 003300 000412
900
901 003302 042777 000020 176066
902 003310 005037 001124
903 003314 017737 176056 001126
904 003322 001401
905
906
909 003324 104002
910
911
912
915 003326
916
917

```

```

:*****
:TEST 10 *TEST THAT CLOCK A STATUS REGISTER BIT 4 CAN BE SET AND CLEARED
:
:CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
:F/FS OR GATES
:
:*****
↑ST10: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
CLR $ASR ;/CLEAR THE STATUS REGISTER.
BIS #BIT4,$ASR ;/SET BIT 4.
MOV #BIT4,$GDDAT ;/SET FOR ERROR TIMEOUT S/B.
MOV $ASR,$BDDAT ;/READ THE STATUS REGISTER.
CMP $GDDAT,$BDDAT ;/DID BIT 4 AND ONLY BIT 4 SET?
BEQ 15 ;/IF SO-LETS TRY CLEARING IT.

:*****
: *****
ERROR <<<*****
ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
;/BIT 4 FAILED TO BIT SET.

: *****
ERROR <<<*****
BR 25 ;/BR TO END SUBTEST.
15: BIC #BIT4,$ASR ;/TRY CLEARING BIT 4.
CLR $GDDAT ;/CLEAR S/B FOR TIMEOUT IF ANY.
MOV $ASR,$BDDAT ;/NOW READ IT BACK.
BEQ 25 ;/IF ZERO - NO ERROR!

: *****
ERROR <<<*****
ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
;/BIT 4 FAILED TO CLEAR.

: *****
ERROR <<<*****
25:

```

N03

MAINDEC-11-DVMNC-A  
DVMnCA.P11 T10

MACY11 27(654) 27-DEC-77 09:32 PAGE 21  
\*TEST THAT CLOCK A STATUS REGISTER BIT 4 CAN BE SET AND CLEARED

SE2 0039

```

918
919
920      ;*****
921      ;*TEST 11      *TEST THAT CLOCK A STATUS REGISTER BIT 3 CAN BE SET AND CLEARED
922      ;*
923      ;*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
924      ;*F/FS OR GATES
925      ;*
926      ;*****
927 003326 000004
928 003330 012737 000100 001160
929
930 003336 005077 176034
931 003342 052777 000010 176026
932 003350 012737 000010 001124
933 003356 017737 176014 001126
934 003364 023737 001124 001126
935 003372 001402
936
          ST11: SCOPE
          MOV     #100,$TIMES      ;;DO 100 ITERATIONS
          CLR     @ASR             ;/CLEAR THE STATUS REGISTER.
          BIS     @BIT3,@ASR      ;/SET BIT 3.
          MOV     @BIT3,$GDDAT    ;/SET FOR ERROR TYPEOUT S/B.
          MOV     @ASR,$BDDAT     ;/READ THE STATUS REGISTER.
          CMP     $GDDAT,$BDDAT   ;/DID BIT 3 AND ONLY BIT 3 SET?
          BEQ     1$              ;/IF SO-LETS TRY CLEARING IT.

          ;*****
          ;*****
939 003374 104002
940
          ERROR 2                  ;/ERROR CLOCK AS STATUS REGISTER
941
          ;*****
          ;*****
944 003376 000412
945
          BR      2$                ;/BR TO END SUBTEST.
946 003400 042777 000010 175770 1$: BIC     @BIT3,@ASR      ;/TRY CLEARING BIT 3.
947 003406 005037 001124
948 003412 017737 175760 001126
949 003420 001401
950
          CLR     $GDDAT          ;/CLEAR S/B FOR TYPEOUT IF ANY.
          MOV     @ASR,$BDDAT     ;/NOW READ IT BACK.
          BEQ     2$              ;/IF ZERO - NO ERROR!
951
          ;*****
          ;*****
954 003422 104002
955
          ERROR 2                  ;/ERROR - CLOCK A STATUS REGISTER.
956
          ;*****
          ;*****
960 003424
961
          ;*****
          ;*****
          ERROR 2                  ;/ERROR - CLOCK A STATUS REGISTER.
962
          ;*****
          ;*****

```

MAINDEC-11-DVMNC-A  
DVMNCA.P11 T11

MACY11 27.654) 27-DEC-77 09:32 PAGE 22  
\*TEST THAT CLOCK A STATUS REGISTER BIT 3 CAN BE SET AND CLEARED

SEG 0040

```

963
964
965
966
967
968
969
970
971
972 003424 000004
973 003426 012737 000100 001160
974
975 003434 005077 175736
976 003440 052777 000004 175730
977 003446 012737 000004 001124
978 003454 017737 175716 001126
979 003462 023737 001124 001126
980 003470 001402
981

```

```

*****
*TEST 12 *TEST THAT CLOCK A STATUS REGISTER BIT 2 CAN BE SET AND CLEARED
*
*CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
*F/FS OR GATES
*
*****
ST12: SCOPE
MOV #100,$TIMES ;;DO 100 ITERATIONS
CLR JASR ;/CLEAR THE STATUS REGISTER.
BIS #BIT2,JASR ;/SET BIT 2.
MOV #BIT2,$GDDAT ;/SET FOR ERROR TYPEOUT S/B.
MOV JASR,$BDDAT ;/READ THE STATUS REGISTER.
CMP $GDDAT,$BDDAT ;/DID BIT 2 AND ONLY BIT 2 SET?
BEQ IS ;/IF SO-LETS TRY CLEARING IT.

```

:::\*\*\*\*\*>>> ERROR <<<\*\*\*\*\*

```

984 003472 104002
985
986

```

```

ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
;/BIT 2 FAILED TO BIT SET.

```

:::\*\*\*\*\*>>> ERROR <<<\*\*\*\*\*

```

989 003474 000412
990
991 003476 042777 000004 175672
992 003504 005037 001124
993 003510 017737 175662 001126
994 003516 001401
995
996

```

```

BR 25 ;/BR TO END SUBTEST.
IS: BIC #BIT2,JASR ;/TRY CLEARING BIT 2.
CLR $GDDAT ;/CLEAR S/B FOR TYPEOUT IF ANY.
MOV JASR,$BDDAT ;/NOW READ IT BACK.
BEQ 25 ;/IF ZERO - NO ERROR!

```

:::\*\*\*\*\*>>> ERROR <<<\*\*\*\*\*

```

999 003520 104002
1000
1001
1002

```

```

ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
;/BIT 2 FAILED TO CLEAR.

```

:::\*\*\*\*\*>>> ERROR <<<\*\*\*\*\*

```

1005 003522
1006
1007

```

```

25:

```

```

1008 ;/0
1009 ;:*****
1010 ;:TEST 13 *TEST THAT CLOCK A STATUS REGISTER BIT 1 CAN BE SET AND CLEARED
1011 ;*
1012 ;:CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
1013 ;:F/FS OR GATES
1014 ;*
1015 ;:*****
1016 ;:*****
1017 003522 000004 ST13: SCOPE
1018 003524 012737 000100 001160 MOV #100,$TIMES ;:DO 100 ITERATIONS
1019
1020 003532 005077 175640 CLR JASR ;/CLEAR THE STATUS REGISTER.
1021 003536 052777 000002 175632 BIS #BIT1,JASR ;/SET BIT 1.
1022 003544 012737 000002 001124 MOV #BIT1,$GDDAT ;/SET FOR ERROR TYPEOUT S/B.
1023 003552 017737 175620 001126 MOV JASR,$BDDAT ;/READ THE STATUS REGISTER.
1024 003560 023737 001124 001126 CMP $GDDAT,$BDDAT ;/DID BIT 1 AND ONLY BIT 1 SET?
1025 003566 001402 BEQ 1$ ;/IF SO-LETS TRY CLEARING IT.
1026
;:*****
1029 003570 104002 ERROR 2 ;/ERROR CLOCK AS STATUS REGISTER
1030 ;/BIT 1 FAILED TO BIT SET.
1031
;:*****
1034 003572 000412 BR 2$ ;/BR TO END SUBTEST.
1035
1036 003574 042777 000002 175574 1$: BIC #BIT1,JASR ;/TRY CLEARING BIT 1.
1037 003602 005037 001124 CLR $GDDAT ;/CLEAR S/B FOR TYPEOUT IF ANY.
1038 003606 017737 175564 001126 MOV JASR,$BDDAT ;/NOW READ IT BACK.
1039 003614 001401 BEQ 2$ ;/IF ZERO - NO ERROR!
1040
;:*****
1044 003616 104002 ERROR 2 ;/ERROR - CLOCK A STATUS REGISTER.
1045 ;/BIT 1 FAILED TO CLEAR.
1046
;:*****
1050 003620 2$:
1051
1052

```

D04

MAINDEC-11-DVMNC-A  
DVMNCA.F11 13

MACY11 27(654) 27-DEC-77 09:32 PAGE 24  
\*TEST THAT CLOCK A STATUS REGISTER BIT 1 CAN BE SET AND CLEARED

SEG 0042

```

1053          ;/
1054          ;*****
1055          *TEST 14      *TEST THAT CLOCK A STATUS REGISTER BIT 0 CAN BE SET AND CLEARED
1056          *
1057          *CLOCK STATUS REGISTER BIT EXERCISE. ON FAILURE-SUSPECT INDIVIDUAL
1058          *F/FS OR GATES
1059          *
1060          ;*****
1061          ;ST14: SCOPE
1062 003620 000004          MOV      #100,$TIMES      ;;DO 100 ITERATIONS
1063 003622 012737 000100 001160          CLR      @ASR          ;;CLEAR THE STATUS REGISTER.
1064          BIS      @BIT0,@ASR          ;;SET BIT 0.
1065 003630 005077 175542          MOV      @BIT0,$GDDAT  ;;SET FOR ERROR TIMEOUT S/B.
1066 003634 052777 000001 175534          MOV      @ASR,$BDDAT  ;;READ THE STATUS REGISTER.
1067 003642 012737 000001 001124          CMP      $GDDAT,$BDDAT  ;;DID BIT 0 AND ONLY BIT 0 SET?
1068 003650 017737 175522 001126          BEQ      1$           ;;IF SO-LETS TRY CLEARING IT.
1069 003656 023737 001124 001126
1070 003664 001402
1071          ;:*****
1072          ;*****
1073          ;*****
1074 003666 104002          ERROR 2           ;;ERROR CLOCK AS STATUS REGISTER
1075          ;/BIT 0 FAILED TO BIT SET.
1076          ;:*****
1077          ;*****
1078          ;*****
1079 003670 000412          BR      2$           ;;BR TO END SUBTEST.
1080          ;*****
1081 003672 042777 000001 175476 1$:  BIC      @BIT0,@ASR          ;;TRY CLEARING BIT 0.
1082 003700 005037 001124          CLR      $GDDAT      ;;CLEAR S/B FOR TIMEOUT IF ANY.
1083 003704 017737 175466 001126          MOV      @ASR,$BDDAT  ;;NOW READ IT BACK.
1084 003712 001401          BEQ      2$           ;;IF ZERO - NO ERROR!
1085          ;*****
1086          ;*****
1087          ;*****
1088          ;*****
1089 003714 104002          ERROR 2           ;;ERROR - CLOCK A STATUS REGISTER.
1090          ;/BIT 0 FAILED TO CLEAR.
1091          ;:*****
1092          ;*****
1093          ;*****
1094          ;*****
1095 003716          2$:
1096          .RADIX 8
1097          ;*****
1098          ;*****
1099          ;*****
1100          ;*****
1101          *TEST 15      *TEST THAT PATTERN 125252 WILL SET AND CLEAR IN BUFFER REG.
1102          ;*****
1103 003716 000004          ;ST15: SCOPE
1104          ;*****
1105 003720 005077 175454          CLR      @ABR          ;;CLEAR THE BUFFER REG.
1106 003724 012737 125252 001124          MOV      #125252,$GDDAT  ;;RECORD PATTERN: 125252 .

```



F04

MAINDEC-11-DVMNC-A  
DVMNCA.P11 T16

MACY11 27.654) 27-DEC-77 09:32 PAGE 36  
\*TEST THAT PATERN 052525 WILL SET AND CLEAR IN BUFFER REG.

SEG 0044

```

1161
1162 004052 042777 052525 175320 15: BIC #052525, @ABR ;/TRY CLEARING PATTERN.
1163 004060 005037 001124 CLR $GDDAT ;/EXPECT ZERO BACK.
1164 004064 017737 175310 001126 MOV @ABR, @BDDAT ;/READ BUFFER REG., WAS IT ZERO?
1165 004072 001401 BEQ 25 ;/YES-NEXT TEST.
1166
1167
1168
1169
1170 004074 104003 ERROR 3 ;/BUFFER REG. COULD NOT BE LOADED
1171 ;/TO A ZERO.
1172
1173
1174
1175 004076 25:
1176
1177
1178
1179
1180 .SBTTL *
1181 .SBTTL * PHASE 2 ADVANCED BASIC LOGIC TESTS
1182 .SBTTL *
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193 004076 000004 *TEST 17 *TEST THE LOW BYTE OPERATION OF CLOCK'S STATUS REGISTER
1194 004100 012737 000050 001160 ;*
1195
1196 004106 005077 175264 CLP @ASR ;/MAKE SURE THE STATUS REGISTER IS CLEAR.
1197 004112 112777 127677 175256 MCVB @127677, @ASR ;/TRY WRITING ALL BITS IN THE
1198 ;/STATUS REGISTER. LOGIC SHOULD PREVENT IT
1199 ;/FROM BEING WRITTEN INTO BECAUSE
1200 ;/WE ARE USING A DATAB INSTRUCTION.
1201
1202 004120 017777 175252 175000 MOV @ASR, @BDDAT ;/NOW EXAMINE THE
1203 ;/STATUS REGISTER.
1204 004126 013737 001126 001124 MOV @BDDAT, $GDDAT ;/FIX $GDDAT FOR ERROR TYPEOUT IF
1205 004134 105037 001125 CLR @B $GDDAT+1 ;/ANY RROR HAS OCCURRED. UPPER BYTE CLEARED.
1206
1207 004140 105737 001127 TST @B $BDDAT+1 ;/ARE ANY BITS IN THE UPPER BYTE
1208 ;/OF THE STATUS REGISTER SET?
1209 004144 001401 BEQ 15 ;/BRANCH NEXT TEST IF UPPER BYTE=0.
1210
1211
1212
1213
1214 004146 104001 ERROR 1 ;/ERROR - WROTE INTO UPPER BYTE OF

```





```

1269 004232 000004 TST21: SCOPE
1270 004234 012737 000100 001160 MOV #100,$TIMES ;;DO 100 ITERATIONS
1271
1272 004242 005077 175130 CLR @ASR ;SELECT MODE 0.
1273 004246 012777 125252 175124 MOV #125252,@ABR ;LOAD THE BUFFER REGISTER WITH
;PATTERN 125252. IT WILL BE
1274 ;TRANSFERRED TO THE COUNT REGISTER
1275 ;SINCE THIS IS MODE 0.
1276 004254 052777 000001 175114 BIS #BIT0,@ASR ;SET GO BIT(ALLOWS BUFFER-COUNT REG XFER .
1277
1278 004262 012737 125252 001124 MOV #125252,$GDDAT ;SET EXPECTED TO PATTERN IN CASE OF
1279 ;NEED OF ERROR TYPEOUT.
1280 ;/-RDCLK-
1281 004270 017746 175102 MOV @ASR,-(6) ;/SAVE CSR
1282 004274 052777 004007 175074 BIS #4007,@ASR ;/SET MODE 3,DIS INTR OSC NO RATE
1283 ;/THIS MUST BE DONE IN
1284 ;/ORDER TO XFERR COUNTER
1285 ;/TO BUFFER ON ST2.
1286 004302 052777 001000 175066 BIS #BIT9,@ASR ;/GENERATE ON ST2 PULSE
1287 004310 012746 000010 MOV #8,-(SP) ;/NOW GENERATE
1288 004314 052777 000400 175054 64$: BIS #BIT8,@ASR ;/EIGHT ST1 PULSES
1289 004322 005316 DEC (SP)
1290 004324 001373 BNE 64$
1291 004326 005726 TST (SP)+ ;/RESET STACK
1292 004330 017737 175044 001126 MOV @ABR,$BDDAT ;/READ THE PRESET BUFFER.
1293 ;/PREVIOUS COUNTER
1294 004336 012677 175034 MOV (6)+@ASR ;/CONTENTS ARE IN $BDDAT.
1295 004342 005737 001126 TST $BDDAT ;/RESTORE CSR
1296
1297
1298 004346 023737 001124 001126 CMP $GDDAT,$BDDAT ;DID ALL THE BITS AND NO OTHER BITS
1299 ;COME THROUGH?
1300 004354 001401 BEQ 1$ ;BR IF YES TO NEXT TEST.
1301
1302

```

:: \$ > ERROR <<< \$

```

1305 004356 104005 ERROR 5 ;DATA ERROR CLOCK - PATTERN "125252"
1306 ;FAILED TO TRANSFER PROPERLY BETWEEN
1307 ;BUFFER AND COUNT REGISTERS.
1308
1309

```

:: \$ >> ERROR <<< \$

```

1312 004360 1$:
1313
1314
1315 ;*****
1316 ;*TEST 22 *TEST CLOCKS COUNTER REGISTER WITH 052525 PATTERN
1317 ;*****
1318 004360 000004 TST22: SCOPE
1319 004362 012737 000050 001160 MOV #50,$TIMES ;;DO 50 ITERATIONS
1320
1321 004370 005077 175002 CLR @ASR ;SELECT MODE 0.
1322 004374 012777 052525 174776 MOV #052525,@ABR ;LOAD THE BUFFER REGISTER WITH
;PATTERN 052525. IT WILL BE

```

104

MAINDEC-11-DVMNC-A  
DVMNCA.P11 '22

MAY11 27 654) 27-DEC-77 09:32 PAGE 29  
\*TEST CLOCKS COUNTER REGISTER WITH 052525 PATTERN

SEG 0047

```

1323                                     : TRANSFERRED TO THE COUNT REGISTER
1324                                     : SINCE THIS IS MODE 0.
1325 004402 052777 000001 174766      BIS    #BIT0,CSR    : SET BC BIT(ALLOWS BUFFER-COUNT REG XFER).
1326                                     :
1327 004410 012737 052525 001124      MOV    #052525,$GDDAT : SET EXPECTED TO PATTERN IN CASE OF
1328                                     : NEED OF ERROR TYPEOUT.
1329                                     : /-RDCLK-
1330 004416 017746 174754                MOV    JASR, -(6)    : /SAVE CSR
1331 004422 052777 004007 174746      BIS    #4007,JASR   : /SET MODE 3,DIS INTR OSC NO RATE
1332                                     : /THIS MUST BE DONE IN
1333                                     : /ORDER TO XFERR COUNTER
1334                                     : /TO BUFFER ON ST2.
1335 004430 052777 001000 174740      BIS    #BIT9,JASR   : /GENERATE ON ST2 PULSE
1336 004436 012746 000010                MOV    #8, -(SP)    : /NOW GENERATE
1337 004442 052777 000400 174726 64$ : BIS    #BIT8,JASR
1338 004450 005316                DEC    (SP)         : /EIGHT ST1 PULSES
1339 004452 001373                BNE    64$
1340 004454 005726                TST   (SP)+
1341 004456 017737 174716 001126      MOV    JABR,$BDDAT : /RESET STACK
1342                                     : /READ THE PRESET BUFFER.
1343 004464 012677 174706                MOV    (6)+,JASR   : /PREVIOUS COUNTER
1344 004470 005737 001126                TST   $BDDAT      : /CONTENTS ARE IN $BDDAT.
1345                                     : /RESTORE CSR
1346 004474 023737 001124 001126      CMP    $GDDAT,$BDDAT : DID ALL THE BITS AND NO OTHER BITS
1347                                     : CAME THROUGH?
1348 004502 001401                BEQ    1$         : BR IF YES TO NEXT TEST.
1349
1350

```

;; \$ >>> ERROR <<< \$

```

1353 004504 104005                ERROR 5            : DATA ERROR CLOCK - PATTERN "052525"
1354                                     : FAILED TO TRANSFER PROPERLY BETWEEN
1355                                     : BUFFER AND COUNT REGISTERS.
1356
1357

```

;; \$ >>> ERROR <<< \$

```

1360 004506
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370 004506 000004
1371 004510 012737 000005 001160     ST23: SCOPE
1372                                     MOV    #5,$TIMES    ;; DO 5 ITERATIONS
1373 004516 005037 001124                CLR    $GDDAT      : EXPECTED DATA IS ZERO.
1374 004522 012777 176377 174646      MOV    #176377,JASR : SET ALL BITS IN THE STATUS REG.
1375
1376 004530 000005                RESET            : SYSTEM INITIALIZE.

```

: \*\*\*\*\*
: \*TEST 23 \*TEST THAT INIT CLEARS STATUS REGISTER
: \*
: \*TESTING OF THE INIT LOGIC AS RECEIVED FROM THE QBUS AND BUFFERED
: \*TO STATUS REGISTER F/FS.
: \*
: \*\*\*\*\*

```

1377      004532  017737  174640  001126      MOV     @ASR,$BDDAT   ;READ THE STATUS REG.. ALL BITS SHOULD
1378                                     ;HAVE BEEN CLEARED BY INIT.
1379                                     ;BR IF YES TO NEXT TEST.
1380      004540  001402      BEQ     IS
1381
1382

```

::SSSSSSSSSSSSSSSSSSSSSSSSSSSSS ERROR <<SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

```

1385      004542  104002      ERROR 2               ;ERROR - SYSTEM INIT FAILED TO CLEAR
1386                                     ;STATUS REGISTER CLOCK A.
1387
1388
1389

```

::SSSSSSSSSSSSSSSSSSSSSSSSSSSSS ERROR <<SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

```

1392      004544  000416      BR      TST24         ;;
1393      004546      IS:              TST     EXS           ;TEST EXTERNAL SIGS?
1394      004546  005737  001444      BEQ     TST24         ;;
1395      004552  001413      TST     DWARF         ;;
1396      004554  005737  001450      BNE     TST24         ;;
1397      004560  001010      BIS     @BIT11!BIT12!BIT10!BIT1,@DR2 ;ENABLE ST1,ST2 TO LATCH.
1398      004562  052777  016002  174626      BIT     @6,@DR        ;ST1,ST2, OVERFLOW SET?
1399      004570  032777  000006  174616      BEQ     TST24         ;;
1400
1401

```

::SSSSSSSSSSSSSSSSSSSSSSSSSSSSS ERROR <<SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

```

1404      004600  104006      ERROR 6               ;INIT FAILED TO CLEAR
1405                                     ;ST1,ST2, AND/OR OVERFLOW
1406

```

::SSSSSSSSSSSSSSSSSSSSSSSSSSSSS ERROR <<SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

```

1409      *****
1410      *TEST 24      *TEST THAT INIT CLEARS BUFFER REGISTER
1411      *
1412      *THIS TEST IS DESIGNED TO SEE IF "INIT H"
1413      *CLEARS THE BUFFER REGISTER. WE ALREADY
1414      *KNOW IT CLEARS THE STATUS REG.
1415      *
1416      *****

```

```

1418      004602  000004      ST24:  SCOPE
1419      004604  012737  000005  001160      MOV     @5,$TIMES   ;;DO 5 ITERATIONS
1420
1421      004612  005037  001124      CLR     $GDDAT      ;CLEAR EXPECTED DATA.
1422      004616  012777  177777  174554      MOV     @177777,@ABR ;SET ALL BITS IN THE BUFFER REGISTER.
1423
1424      004624  000005      RESET                                ;ISSUE SYSTEM INITIALIZE.
1425
1426      004626  017737  174546  001126      MOV     @ABR,$BDDAT ;READ THE BUFFER REGISTER. ALL BITS
1427      004634  001401      BEQ     IS           ;SHOULD HAVE BEEN CLEARED BY INIT.
1428                                     ;BR IF YES TO NEXT SUBTEST.
1429
1430

```





;AND A MAINTENANCE ST2 GENERATED.

1539  
1540  
1541

:: SSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1544  
1545  
1546  
1547

005064 032777 020000 174304  
005072 001401

1S: BIT #BIT13, QASR ;DID BIT 13 CLEAR ITSELF??  
BEQ 2S ;YES GO AHEAD.

:: SSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1550  
1551

005074 104001

ERROR 1 ;BIT 13 FAILED TO CLEAR.

:: SSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1554  
1555  
1556  
1557  
1558  
1559

005076 005077 174274

2S: CLR QASR ;LEAVE SUBTEST WITH CLOCK CLEAR.

.SBTTL \*  
.SBTTL \*PHASE 3 COUNT TESTS  
.SBTTL \*

1560  
1561  
1562

:: \*\*\*\*\*  
\*TEST 30 \*TEST TO SEE IF THE COUNTER WILL INCREMENT  
\*\*\*\*\*  
↑ST30: SCOPE

1563

005102 000004

CLR QASR ;CLEAR THE CSR.  
CLR QABP ;CLEAR THE BUFFER  
BIS #BITS!BIT4!BIT0, QASR ;SET RATE: ST1, GO.  
BIS #BITB, QASR ;GENERATE A MAINTENANCE ST1.  
;DID THE CLOCK COUNT?

1570  
1571  
1572

005130 017746 174242  
005134 052777 004007 174234

MOV QASR, -(6) ;/-ROCLK-  
BIS #4007, QASR ;SAVE CSR  
;SET MODE 3, DIS INTR OSC NO RATE  
;THIS MUST BE DONE IN  
;ORDER TO XFERA COUNTER  
;TO BUFFER ON ST2.  
;GENERATE ON ST2 PULSE  
;NOW GENERATE

1576  
1577  
1578

005142 052777 001000 174226  
005150 012746 000010  
005154 052777 000400 174214

64S: BIS #BITB, QASR ;/EIGHT ST1 PULSES  
DEC (SP)  
BNE 64S

1580  
1581  
1582

005164 001373  
005166 005726  
005170 017737 174204 001126

TST (SP)+ ;/RESET STACK  
MOV QABR, \$BDDAT ;/READ THE PRESET BUFFER,  
;/PREVIOUS COUNTER  
MOV (6)+, QASR ;/CONTENTS ARE IN \$BDDAT.  
TST \$BDDAT ;/RESTORE CSR  
BNE 1S ;YES. NEXT TEST.

1583  
1584  
1585  
1586  
1587

005176 012677 174174  
005202 005737 001126  
005206 001001

:: SSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1590  
1591  
1592

005210 104006

ERROR 6 ;CLOCK FAILED TO INCREMENT.





```

1647      ::::::::::::::::::::::::::::::::::::>>> ERROR <<<::::::::::::::::::::::::::::::::::
1650      005400 104011      ERROR 11      ;CLOCK FAILED TO COUNT UP PROPERLY.
1651      ::::::::::::::::::::::::::::::::::::>>> ERROR <<<::::::::::::::::::::::::::::::::::

1654      005402 000410      BR 35      ;GOTO SCOPE LOOP.
1655
1656      005404 005077 173766 25: CLR QASR
1657      005410 013777 001124 173762 MOV $GDDAT,QABR
1658      005416 005737 001124 TST $GDDAT ;ALL DONE?
1659      005422 001311 BNE 15 ;NO DO NEXT INCREMENT.
1660      005424 35:
1661
1662      ::*****
1663      ;*TEST 32 *TEST THAT OVERFLOW (CSR BIT07) WILL SET ON OVERFLOW
1664      ;*****
1665      005424 000004 †TST32: SCOPE
1666
1667      005426 005737 001444 TST EXS ;TESTOR MODE ENABLED??
1668      005432 001410 BEQ 25 ;NO-THEN SKIP NEXT SECTION OF CODE.
1669      005434 005737 001450 TST DWARF
1670      005440 001042 BNE TST33 ;:
1671
1672      005442 052777 020002 173746 BIS #BIT13!BIT1,QDR2
1673      005450 012700 000010 MOV #8.,R0 ;SET TIME OUT NUMBER.
1674      005454 25:
1675      005454 005077 173716 CLR QASR ;CLEAR THE CSR
1676      005460 012777 177777 MOV #-1,QABR ;SET PRESET BUFFER TO ALL ONES.
1677
1678      005466 052777 000061 173702 BIS #BITS!BIT4!BIT0,QASR ;START CLOCK, RATE ST1.
1679
1680      005474 052777 000400 173674 BIS #BIT8,QASR ;COUNT CLOCK ONCE, OVERFLOW
1681      ;SHOULD OCCUR.
1682      005502 105777 173670 TSTB QASR ;DID OVERFLOW SET?
1683      005506 100402 BMI 15 ;YES - THEN NEXT TEST
1684
1685      ::::::::::::::::::::::::::::::::::::>>> ERROR <<<::::::::::::::::::::::::::::::::::

1688      005510 104006      ERROR 6      ;ERROR - OVERFLOW, CSR BIT0?
1689      BR TST33 ;FAILED TO SET ON OVERFLOW
1690      005512 000415 15: TST EXS ;TEST EXTERNAL SIGNALS?
1691      005514 005737 001444 BEQ TST33 ;:
1692      005520 001412 TST DWARF ;:
1693      005522 005737 001450 BNE TST33 ;:
1694      005526 001007 BIT #BIT3,QDR ;:
1695      005530 032777 000010 173656 BNE TST33 ;:
1696      005536 001003 DEC R0 ;DID WE ALLOW ENOUGH TIME??
1697      005540 005300 BNE 15 ;NO-THEN WAIT.
1698      005542 001364
1699      ::::::::::::::::::::::::::::::::::::>>> ERROR <<<::::::::::::::::::::::::::::::::::

```

C05

1702 005544 104006  
1703  
1704

ERROR 6 ;OVERFLOW OUT NOT DETECTED  
;BY TESTOR

:: SSSSSSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1707  
1708  
1709  
1710  
1711  
1712 005546 000004  
1713  
1714 005550 005077 173622  
1715  
1716 005554 012777 177777 173616  
1717  
1718 005562 052777 000061 173606  
1719  
1720 005570 052777 000400 173600  
1721  
1722  
1723  
1724 005576 032777 000001 173572  
1725 005604 001401  
1726  
1727

:: \*\*\*\*\*  
;TEST 33 \*TEST THAT OVERFLOW WILL CLEAR THE GO BIT  
;\*\*\*\*\*  
↑ST33: SCOPE

CLR QASR ;CLEAR THE CSR.  
MOV #-1,QABR ;PRESET CLOCK TO -1.  
BIS #BITS!BIT4!BIT0,QASR ;START CLOCK, RATE:ST1  
BIS #BIT8,QASR ;COUNT ONCE, OVERFLOW  
;SHOULD OCCUR CLEARING  
;ENABLE (CSR BIT0)  
BIT #BIT0,QASR ;DID THE ENABLE CLEAR?  
BEQ IS ;YES - NEXT TEST.

:: SSSSSSSSSSSSSSSSSSSSSSSSSSSSS >> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1730 005606 104006  
1731  
1732  
1733 005610  
1734  
1735  
1736  
1737  
1738 005610 000004  
1739  
1740 005612 005077 173560  
1741 005616 012777 177777 173554  
1742 005624 052777 000063 173544  
1743  
1744 005632 052777 000400 173536  
1745  
1746 005640 032777 000001 173530  
1747 005646 001001  
1748

ERROR 6 ;ERROR - OVERFLOW FAILED  
;TO CLEAR ENABLE (CSR BIT0C)

IS.

:: \*\*\*\*\*  
;TEST 34 \*TEST THAT GO BIT DOES NOT CLEAR ON OVERFLOW, IF MODE 1  
;\*\*\*\*\*  
↑ST34: SCOPE

CLR QASR ;CLEAR THE CSR.  
MOV #-1,QABR ;PRESET BUFFER=ONE COUNT FROM OVERFLOW.  
BIS #63,QASR ;MODE 1, RATE:ST1, GO.  
BIS #BIT8,QASR ;GENERATE MAINTENANCE ST1.  
BIT #BIT0,QASR ;DID ENABLE (GO BIT) CLEAR?  
BNE IS ;NO (GOOD) NEXT TEST.

:: SSSSSSSSSSSSSSSSSSSSSSSSSSSSS >>> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

1751 005650 104006  
1752  
1753

ERROR 6 ;GO BIT CLEARED ON OVERFLOW  
;WHEN MODE 1 WAS SELECTED

:: SSSSSSSSSSSSSSSSSSSSSSSSSSSSS >>> ERROR <<< SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS

```

1756 005652 005077 173520 18: CLR QASR ;CLEAR THE CLOCK.
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770 005656 000004
1771 005660 012737 000005 001160
1772
1773
1774 005666 005077 173504 CLR QASR ;/CLEAR CLOCK
1775 005672 005077 173502 CLR QABR ;/CLEAR PRESET BUFFER
1776 005676 012777 000011 173472 MOV #BIT0!10,QASR ;/START CLOCK, MODE0, RATE:1MHZ
1777 005704 005000 CLR RO ;/NOW WE'LL DO A LITTLE DELAY. THIS DELA
1778
1779 005706 005200 18: INC RO ;/WILL AMOUNT TO APPROXIMATELY
1780 005710 001376 BNE 18 ;/369 MS.
1781
1782
1783 005712 017746 173460 MOV QASR,-(6) ;/-RDCLK-
1784 005716 052777 004007 173452 BIS #4007,QASR ;/SAVE CSR
1785 ;/SET MODE 3,DIS INTR OSC NO RATE
1786 ;/THIS MUST BE DONE IN
1787 ;/ORDER TO XFERR COUNTER
1788 ;/TO BUFFER ON ST2.
1789 005724 052777 001000 173444 BIS #BIT9,QASR ;/GENERATE ON ST2 PULSE
1790 005732 012746 000010 MOV #8,-(SP) ;/NOW GENERATE
1791 005736 052777 000400 173432 64$: BIS #BIT8,QASR
1792 005744 005316 DEC (SP) ;/EIGHT ST1 PULSES
1793 005746 001373 BNE 64$
1794 005750 005726 TST (SP)+ ;/RESET STACK
1795 005752 017737 173422 001126 MOV QABR,$BDDAT ;/READ THE PRESET BUFFER,
1796 ;/PREVIOUS COUNTER
1797 005760 012677 173412 MOV (6)+,QASR ;/CONTENTS ARE IN $BDDAT.
1798 005764 005737 001126 TST $BDDAT ;/RESTORE CSR
1799 005770 001004 BNE 2$ ;/YES - NEXT TEST.
1800 005772 105766 177776 TSTB -2(6) ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
1801 ;/NOTE: CSR HAD BEEN PUT ON STACK.
1802 005776 100401 BMI 2$ ;/NEXT TEST IF OVERFLOW.
1803
1804
1805
1806 006000 104006 ERROR 6 ;/CLOCK FAILED TO COUNT AT
1807 ;/RATE:1MHZ
1808
1809
1810

```

\*\*\*\*\*  
:TEST 35 \*TEST THE ABILITY OF CLOCK TO COUNT AT 1MHZ RATE

\*\*\*\*\*  
:THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY  
:TO COUNT AT 1MHZ RATE.  
\*\*\*\*\*

ST35: SCOPE ;:DO 5 ITERATIONS  
MOV #5,\$TIMES

:-RDCLK-  
:/SAVE CSR  
:/SET MODE 3,DIS INTR OSC NO RATE  
:/THIS MUST BE DONE IN  
:/ORDER TO XFERR COUNTER  
:/TO BUFFER ON ST2.  
:/GENERATE ON ST2 PULSE  
:/NOW GENERATE

:/EIGHT ST1 PULSES  
:/RESET STACK  
:/READ THE PRESET BUFFER,  
:/PREVIOUS COUNTER  
:/CONTENTS ARE IN \$BDDAT.  
:/RESTORE CSR  
:/YES - NEXT TEST.  
:/AT HIGH RATE MAY HAVE HAD OVERFLOW  
:/NOTE: CSR HAD BEEN PUT ON STACK.  
:/NEXT TEST IF OVERFLOW.

\*\*\*\*\* ERROR \*\*\*\*\*

:::\*\*\*\*\*  
EPROR <<<\*\*\*\*\*

```

1811
1812 006002 005077 173370 25: CLR QASR ; /CLEAR THE CLOCK.
1813
1814
1815
1816 :::*****
1817 *TEST 36 *TEST THE ABILITY OF CLOCK TO COUNT AT 100KHZ RATE
1818
1819
1820
1821 *THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
1822 *TO COUNT AT 100KHZ RATE.
1823 :::*****
1824 006006 000004 ST36: SCOPE
1825 006010 012737 000005 001160 MOV #5,$TIMES ;,DO 5 ITERATIONS
1826
1827
1828 006016 005077 173354 CLR QASR ; /CLEAR CLOCK
1829 006022 005077 173352 CLR QABR ; /CLEAR PRESET BUFFER
1830 006026 012777 000021 173342 MOV #BIT0!20,QASR ; /START CLOCK, MODE0, RATE:100KHZ
1831 006034 005000 CLR RO ; /NOW WE'LL DO A LITTLE DELAY. THIS DELAY
1832
1833 006036 005200 15: INC RO ; /WILL AMOUNT TO APPROXIMATELY
1834 006040 001376 BNE 15 ; /369 MS.
1835
1836
1837 006042 017746 173330 MOV QASR,-(6) ; /-RDCLK-
1838 006046 052777 004007 173322 BIS #4007,QASR ; /SAVE CSR
1839 ; /SET MODE 3,DIS INTR OSC NC RATE
1840 ; /THIS MUST BE DONE IN
1841 ; /ORDER TO XFERR COUNTER
1842 ; /TO BUFFER ON ST2.
1843 006054 052777 001000 173314 BIS #BIT9,QASR ; /GENERATE ON ST2 PULSE
1844 006062 012746 000010 MOV #8,-(SP) ; /NOW GENERATE
1845 006066 052777 000400 173302 645: BIS #BIT8,QASR ;
1846 006074 005316 DEC (SP) ; /EIGHT ST1 PULSES
1847 006076 001373 BNE 645 ;
1848 006100 005726 TST (SP)+ ; /RESET STACK
1849 006102 017737 173272 001126 MOV QABR,$BDDAT ; /READ THE PRESET BUFFER,
1850 ; /PREVIOUS COUNTER
1851 006110 012677 173262 MOV (6)+,QASR ; /CONTENTS ARE IN $BDDAT.
1852 006114 005737 001126 TST $BDDAT ; /RESTORE CSR
1853 006120 001004 BNE 25 ; /YES - NEXT TEST.
1854 006122 105766 177776 TSTB -2(6) ; /AT HIGH RATE MAY HAVE HAD OVERFLOW
1855 ; /NOTE: CSR HAD BEEN PUT ON STACK.
1856 006126 100401 BMI 25 ; /NEXT TEST IF OVERFLOW.
1857

```

:::\*\*\*\*\*  
EPROR <<<\*\*\*\*\*

```

1860 006130 104006 ERROR 6 ; /CLOCK FAILED TO COUNT AT
1861 ; /RATE:100KHZ
1862

```

F05

MAINDEC-11-DVMNC-A  
DVMNCA.F11 T36

MAY11 27 654 27-DEC-77 09:32 PAGE 39  
\*TEST THE ABILITY OF CLOCK TO COUNT AT 100KHZ RATE

326 0057

```

1865
1866 006132 005077 173240 25: CLR QASR ;/CLEAR THE CLOCK.
1867
1868
1869
1870
1871 *****
1872 :/TEST 37 */TEST THE ABILITY OF CLOCK TO COUNT AT 10KHZ RATE
1873 *****
1874 :/THIS TEST IS DESIGNED TO TEST THE CLOCK'S ABILITY
1875 :/TO COUNT AT 10KHZ RATE.
1876 *****
1877
1878 006136 000004 ST37: SCOPE ;/DO 5 ITERATIONS
1879 006140 012737 000005 001160 MOV #5,STIMES
1880
1881
1882 006146 005077 173224 CLR QASR ;/CLEAR CLOCK
1883 006152 005077 173222 CLR QASR ;/CLEAR PRESET BUFFER
1884 006156 012777 000031 173212 MOV #BIT0:30,QASR ;/START CLOCK, MODE0, RATE:10KHZ
1885 006164 005000 CLR R0 ;/NOW WE'LL DO A LITTLE DELAY. THIS DELAY
1886
1887 006166 005200 15: INC R0 ;/WILL AMOUNT TO APPROXIMATELY
1888 006170 001376 BNE 15 ;/369 MS.
1889
1890
1891 006172 017746 173200 MOV QASR,-(6) ;/RDCLK-
1892 006176 052777 004007 173172 BIS #4007,QASR ;/SAVE CSR
1893 ;/SET MODE 3,DIS INTR OSC NO RATE
1894 ;/THIS MUST BE DONE IN
1895 ;/ORDER TO XFERR COUNTER
1896 ;/TO BUFFER ON ST2.
1897 006204 052777 001000 173164 BIS #BIT9,QASR ;/GENERATE ON ST2 PULSE
1898 006212 012746 000010 MOV #8,-(SP) ;/NOW GENERATE
1899 006216 052777 000400 173152 645: BIS #BIT8,QASR
1900 006224 005316 DEC (SP) ;/EIGHT ST1 PULSES
1901 006226 001373 BNE 645
1902 006230 005726 TST (SP)+ ;/RESET STACK
1903 006232 017737 173142 001126 MOV QASR,$BDDAT ;/READ THE PRESET BUFFER,
1904 ;/PREVIOUS COUNTER
1905 ;/CONTENTS ARE IN $BDDAT.
1906 006244 005737 001126 TST $BDDAT ;/RESTORE CSR
1907 006250 001004 BNE 25 ;/YES - NEXT TEST.
1908 006252 105766 177776 TSTB -2(6) ;/AT HIGH RATE MAY HAVE HAD OVERFLOW
1909 ;/NOTE: CSR HAD BEEN PUT ON STACK.
1910 006256 100401 BMI 25 ;/NEXT TEST IF OVERFLOW.
1911
1912
1913 *****
1914 006260 104006 ERROR 6 ;/CLOCK FAILED TO COUNT AT
1915 ;/RATE:10KHZ
1916
1917 *****

```









J05

```

2080 006730 104006          ERROR 6           :CLOCK FAILED TO COUNT AT
2081                                     :RATE:LINEFREQ
2082
:::*****ERROR*****
2085
2086 006732 005077 172470 25:   CLR   JASR           : CLEAR THE CLOCK.
2087
2088
2089
2090
2091 ::*****
2092 : *TEST 43          *TEST THAT COUNTER DOESN'T COUNT WHEN "SLAVE IN" RATE IS SELECTED
2093 : *****
2094 006736 000004          †ST43: SCOPE
2095 006710 012737 000020 001160 MOV   #20,$TIMES          ;;DO 20 ITERATIONS
2096 006716 005077 172454          CLR   JASR           ;CLEAR CSR.
2097 006722 005077 172452          CLR   JABR           ;CLEAR PRESET BUFFER.
2098
2099 006726 052777 000001 172442 BIS   #BIT0,JASR        ;SET GO BIT, RATE:"SLAVE IN"
2100
2101 006734 052777 000400 172434 BIS   #BIT8,JASR        ;GENERATE A MAINTENANCE ST1.
2102 006742 005000          CLR   R0
2103
2104 006744 005200          15:   INC   RC
2105 006746 001376          BNE   15             ;NOW DO A SHORT DELAY.
2106
2107
2108 006750 017746 172422          MOV   JASR,-(6)        ; -RDCLK-
2109 006754 052777 004007 172414 BIS   #4007,JASR       ; /SAVE CSR
2110                                     ; /SET MODE 3,DIS INTR OSC NO RATE
2111                                     ; /THIS MUST BE DONE IN
2112                                     ; /ORDER TO XFERR COUNTER
2113                                     ; /TO BUFFER ON ST2.
2114 006762 052777 001000 172406 BIS   #BIT9,JASR       ; /GENERATE ON ST2 PULSE
2115 006770 012746 000010          MOV   #8,-(SP)        ; /NOW GENERATE
2116 006774 052777 000400 172374 64$: BIS   #BIT8,JASR
2117 007002 005316          DEC   (SP)
2118 007004 001373          BNE   64$
2119 007006 005726          TST   (SP)+
2120 007010 017737 172364 001126 MOV   JABR,$BDDAT     ; /RESET STACK
2121                                     ; /READ THE PRESET BUFFER,
2122                                     ; /PREVIOUS COUNTER
2123 007016 012677 172354          MOV   (6)+,JASR      ; /CONTENTS ARE IN $BDDAT.
2124 007022 005737 001126          TST   $BDDAT
2125 007026 001003          BNE   25             ; /RESTORE CSR
2126 007030 105777 172342          TSTB  JASR           ; /IF ANY COUNT, ERROR.
2127 007034 100003          BPL   35             ; /CHECK FOR OVERFLOW.
2128                                     ; /IF NO OVERFLOW, NEXT TEST
2129
2130 007036 005037 001124 23:   CLR   $GDDAT        ; CLEAR FOR TYPEOUT, EXP'D ZERO.
2131
2132 :::*****ERROR*****
2133          ERROR 11           :CLOCK COUNTED WHEN RATE:"SLAVE IN"
2134                                     :WAS SELECTED.

```

K05

MAINDEC-11-DVMNC-A  
DVMNCA.F11

MA 11 27 654 27-DEC-77 09:32 PAGE 44  
\*TEST THAT COUNTER DOESN'T COUNT WHEN "SLAVE IN" RATE IS SELECTED

TEO 0062

```

2133          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2136 007044 005077 172326      35:      CLR      QASR           ;CLEAR CLOCK'S CSR.
2137
2138
2139          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2140          :: *TEST 44            *TEST THAT THE CLOCK WILL COUNT IN MODE 1
2141          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2142 007050 000004
2143          †ST44: SCOPE
2144          CLR      QASR           ;CLEAR THE CSR.
2145 007056 012777 172320      MOV      #-1, QABR         ;PRESET BUFFER REG.
2146 007064 052777 177777 172314  BIS      #13, QASR         ;START CLOCK, RATE: 1 MHZ, MODE 1.
2147 007072 005000      CLR      RO
2148 007074 005200      INCB   RO
2149 007076 001376      BNE     IS
2150          ;NOW DO A SHORT DELAY SO THAT THE
2151          ;CLOCK CAN COUNT TO OVERFLOW.
2152 007100 105777 172272      TSTB   QASR           ;OVERFLOW SHOULD HAVE SET.
2153 007104 100401      BMI     25             ;GOTO NEXT TEST IF SO.
2154
2157          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2158          ERROR 6                ;CLOCK FAILED TO COUNT UP AND SET
2159          ;OVERFLOW IN MODE 1.
2160
2161          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2162          ;
2163          ;
2164          ;
2165          .SBTTL #
2166          .SBTTL #PHASE 4 CLOCK INTERRUPT TEST.
2167          .SBTTL #
2168
2169          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2170          :: *TEST 45            *TEST THAT THE CLOCK WILL INTERRUPT ON OVERFLOW
2171          :: ::::::::::::::::::::::::::::::::::::) ) ERROR <<< ::::::::::::::::::::::::::::::::::::::::::::
2172          :: *TEST 45            *TEST THAT THE CLOCK WILL INTERRUPT ON OVERFLOW
2173          :: *TEST 45            *TEST THAT THE CLOCK WILL INTERRUPT ON OVERFLOW
2174          †ST45: SCOPE
2175          MOV      #20, $TIMES      ;;DO 20 ITERATIONS
2176
2177          MOV      #340, -(SP)     ;PUT PRIORITY ON STACK.
2178          MOV      #645, -(SP)     ;PUT RETURN ADDRESS ON STACK
2179          RTI                      ;DO AN RTI, PUTS PRIORITY IN CPU.
2180          645:
2181
2182          CLR      QASR           ;CLEAR CLOCK'S CSR.
2183 007136 012777 177777 172234  MOV      #-1, QABR         ;SET PRESET BUFFER TO ALL ONES.
2184
2185          MOV      #161, QASR        ;START CLOCK, RATE: ST1.
2186 007152 052777 000400 172216  BIS      #BIT8, QASR       ;GENERATE A MAINTENANCE ST1.
  
```



M05

::**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**>>> ERROR <<<**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**

2243 007326 104007 ERROR 7 ;CLOCK FAILED TO INTERRUPT ON ST2.  
2244

::**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**>>> ERROR <<<**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**

2247 007330 000402 BR 25  
2248 007332  
2249 007332 062706 000004 15: ADD #4,SP ;/ADD #4 TO STACK POINTER.  
2250 007336 005077 172034 25: CLR #ASR ;CLEAR CLOCK'S CSR.  
2251 007342 013777 001410 172036 MOV VECT2P,#VECT2 ;2 RESTORE VECTOR.

::**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**  
\*TEST 47 \*TEST THAT ST1 WILL CAUSE AN INTERRUPT  
\***XXXXXXXXXXXXXXXXXXXXXXXXXXXX**  
†ST47: SCOPE

2252  
2253  
2254  
2255  
2256 007350 000004  
2257  
2258  
2259 007352 012746 000340 MOV #340,-(SP) ;PUT PRIORITY ON STACK.  
2260 007356 012746 007364 MOV #645,-(SP) ;PUT RETURN ADDRESS ON STACK.  
2261 007362 000002 RTI ;DO AN RTI, PUTS PRIORITY IN CPU.

645: CLR #ASR ;2 CLEAR CSR  
MOV #15,#VECT2 ;2 SET UP INTR. VECTOR.  
BIS #BIT14!BIT8,#ASR ;2 INTR ENABLE AND ST1.

2262 007364  
2263 007364 005077 172006  
2264 007370 012777 007436 172010  
2265 007376 052777 040400 171772  
2266  
2267 007404 012746 000000 MOV #0,-(SP) ;PUT PRIORITY ON STACK.  
2268 007410 012746 007416 MOV #655,-(SP) ;PUT RETURN ADDRESS ON STACK  
2269 007414 000002 RTI ;DO AN RTI, PUTS PRIORITY IN CPU.

655: NOP ;2 INTR. FROM HERE.

2270 007416  
2271 007416 000240  
2272  
2273 007420 012746 000340 MOV #340,-(SP) ;PUT PRIORITY ON STACK.  
2274 007424 012746 007432 MOV #665,-(SP) ;PUT RETURN ADDRESS ON STACK  
2275 007430 000002 RTI ;DO AN RTI, PUTS PRIORITY IN CPU.

665:

::**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**>>> ERROR <<<**XXXXXXXXXXXXXXXXXXXXXXXXXXXX**

2280 007432 104007 ERROR 7 ;2 CLOCK FAILED TO INTR. ON ST1.  
2281 007434 000402 BR 25

15: ADD #4,SP ;/ADD #4 TO STACK POINTER.  
25: CLR #ASR ;2 CLEAR CSR.  
MOV VECT2P,#VECT2 ;2 RESTORE INTR. VECTOR.

2282 007436  
2283 007436 062706 000004  
2284 007442 005077 171730  
2285 007446 013777 001410 171732

.SBTTL \*  
.SBTTL \*PHASE 5 ADVANCED TESTING  
.SBTTL \*

2286  
2287  
2288  
2289  
2290

```

2291
2292
2293
2294 007454 000004
2295
2296 007456 005077 171714 CLR QASR ; START WITH CSR CLEAR.
2297 007462 005277 171710 INC QASR ; SET GO BIT.
2298 007466 052777 001000 171702 BIS #BIT9,QASR ; GENERATE THE 1ST ST2 PULSE.
2299 007474 052777 001000 171674 BIS #BIT9,QASR ; GENERATE 2ND ST2 PULSE.
2300 ; THIS SHOULD CAUSE FOR BIT TO SET.
2301 007502 032777 010000 171666 BIT #BIT12,QASR ; DID FOR BIT SET?
2302 007510 001007 BNE IS ; YES-THEN NEXT TEST.
2303
2304 007512 017737 171660 001126 MOV QASR,$BDDAT ; RECORD CSR.
2305 007520 012737 110001 001124 MOV #BIT15!BIT12!BIT0,$GDDAT ; RECORD S/B.
2306
:: *****
:: *TEST 50 *TEST THAT THE "FOR" BIT WILL SET ON 2 ST2'S
:: *****
↑ST50: SCOPE
; *****
; ERROR <<< *****
2309 007526 104001 ERROR 1 ; ERROR-"FOR" BIT FAILED TO SET ON
2310 ; ON TWO SUCCESSIVE ST2 PULSES.
2311
; *****
; ERROR <<< *****
2314 007530
2315
2316
2317
2318 007530 000004
2319
2320
2321 007532 005077 171640 CLR QASR ; START WITH THE CSR CLEAR.
2322 007536 005277 171634 INC QASR ; SET GO BIT.
2323 007542 052777 000400 171626 BIS #BIT08,QASR ; GENERATE AN ST1.
2324 007550 052777 000400 171620 BIS #BIT08,QASR ; GENERATE ANOTHER ST1.
2325 ; AT THIS POINT THE "FOR" BIT SHOULD HAVE SET
2326 007556 032777 010000 171612 BIT #BIT12,QASR ; DID THE FOR BIT SET?
2327 007564 001007 BNE TST52
2328 007566 017737 171604 001126 MOV QASR,$BDDAT ; RECORD CSR.
2329 007574 012737 012001 001124 MOV #BIT10!BIT12!BIT0,$GDDAT ; RECORD S B.
2330
:: *****
:: *TEST 51 *TEST THAT THE "FOR" BIT WILL SET ON 2 ST1'S
:: *****
↑ST51: SCOPE
; *****
; ERROR <<< *****
2333 007602 104001 ERROR 1 ; ERROR- "FOR" BIT FAILED TO SET ON
2334 ; TWO SUCCESSIVE ST1 PULSES.
2335
; *****
; ERROR <<< *****
2338
2339
2340
2341
2342
2343 007604 000004
2344 007606 012737 000002 001160
↑ST52: SCOPE
MOV #2,$TIMES ; DO 2 ITERATIONS

```

```

2345
2346 007614 005077 171556          CLR      QASR          ; START WITH CSR CLEAR.
2347 007620 012777 177777 171552    MOV      #-1,QABF     ; PRELOAD BUFFER REG.
2348 007626 052777 000013 171542    BIS      #13,QASR     ; START CLOCK MODE 1.1MHZ.
2349 007634 005000          CLR      R0          ; NOW DO A SHORT DELAY.
2350 007636 105200          INCB    R0          ; DURING THIS DELAY, THE CLOCK WILL
2351 007640 001376          BNE     1$          ; HAVE OVERFLOWED TWICE-
2352                                     ; THIS SHOULD CAUSE THE FOR BIT TO SET.
2353
2354 007642 032777 010000 171526    BIT      #BIT12,QASR ; DID FOR SET?
2355 007650 001007          BNE     2$          ; YES-NEXT TEST.
2356
2357 007652 017737 171520 001126    MOV      QASR,$BDDAT ; NO-RECORD THE CSR.
2358 007660 012737 010213 001124    MOV      #010213,$GDDAT ; RECORD S/B.
2359
:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ >> ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$

2362 007666 104001          ERROR   1          ; ERROR FOR BIT FAILED TO SET ON
2363                                     ; TWO SUCCESSIVE OVERFLOWS.
2364
:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ >> ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$

2367 007670
2368 2$:
2369 :: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$
2370 :: *TEST 53 *TEST THAT FOR BIT WILL CLEAR IF GO BIT IS SET
2371 :: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$
2372 †ST53: SCOPE
2373 007672 012777 000001 171476    MOV      #BIT0,QASR   ; CLEAR CSR SET GO BIT.
2374 007700 052777 001000 171470    BIS      #BIT9,QASR   ; SET 1ST ST2 PULSE.
2375 007706 052777 001000 171462    BIS      #BIT9,QASR   ; GENERATE 2ND ST2 PULSE.
2376                                     ; FOR BIT SETS HERE.
2377 007714 042777 100001 171454    BIC      #BIT0!BIT15,QASR ; CLEAR GO BIT AND ST2 FLAG.
2378
2379 007722 052777 000001 171446    BIS      #BIT0,QASR   ; SET THE "GO" BIT AGAIN -
2380                                     ; SHOULD CLEAR FOR BIT.
2381
2382 007730 017737 171442 001126    MOV      QASR,$BDDAT ; READ THE CSR.
2383
2384 007736 012737 100001 001124    MOV      #100001,$GDDAT ; RECORD WHAT CSR S/B.
2385 007744 032737 010000 001126    BIT      #BIT12,$BDDAT ; DID FOR BIT CLEAR?
2386 007752 001401          BEQ     1$          ; YES NEXT TEST.
2387
:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ >> ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$

2390 007754 104001          ERROR   1          ; FOR BIT FAILED TO CLEAR
2391                                     ; WHEN "GO" BIT WAS SET.
2392
:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ >> ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$

2395 007756 005077 171414          1$: CLR      QASR          ; CLEAR THE CSR.
2396
2397 :: $$$$$$$$$$$$$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$
2398 :: *TEST 54 *TEST THAT WE CAN DISABLE THE INTERNAL OSC

```

```

2399
2400 007762 000004
2401 007764 012737 000005 001160
2402 007772 005077 171400
2403 007776 005077 171376
2404 010002 005037 001124
2405
2406 010006 012777 004000 171362
2407 010014 052777 000011 171354
2408 010022 005000
2409 010024 105200
2410 010026 001376
2411
2412 010030 017746 171342
2413 010034 052777 004007 171334
2414
2415
2416
2417
2418 010042 052777 001000 171326
2419 010050 012746 000010
2420 010054 052777 000400 171314
2421 010062 005316
2422 010064 001373
2423 010066 005726
2424 010070 017737 171304 001126
2425
2426 010076 012677 171274
2427 010102 005737 001126
2428 010106 001401
2429

::*****
↑ST54: SCOPE
MOV #5,STIMES ;:DO 5 ITERATIONS
CLR QASR ;:CLEAR THE CSR
CLR QABR ;:CLEAR THE PRESET BUFFER
CLR $GDUAT ;:CLEAR EXPED.
MOV #BIT11,QASR ;:DISABLE THE INTERNAL OSC.
BIS #BIT3!BIT0,QASR ;:START CLOCK:RATE 1MHZ.
CLR RO
INC RO ;:DELAY A SHORT TIME.
BNE IS
MOV QASR, -(6 ;:/-RDCLK-
BIS #4007,QASP ;:/SAVE CSR
;:/SET MODE 3,DIS INTR OSC,NO RATE
;:/THIS MUST BE DONE IN
;:/ORDER TO XFERR COUNTER
;:/TO BUFFER ON ST2.
BIS #BIT9,QASR ;:/GENERATE ON ST2 PULSE
MOV #8, -(SP) ;:/NOW GENERATE
BIS #BIT8,QASR ;:/EIGHT ST1 PULSES
DEC (SP)
BNE 64$
TST (SP)+
MOV QABR,$BDDAT ;:/RESET STACK
;:/READ THE PRESET BUFFER.
;:/PREVIOUS COUNTER
;:/CONTENTS ARE IN $BDDAT.
MOV (6)+,QASR ;:/RESTORE CSR
TST $BDDAT
BEQ 2$ ;:NO - GOOD - NEXT TEST.

:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$
2432 010110 104011 ERROR 11 ;:CLOCK DISABLE INTERNAL
2433 ;:OSC. DID NOT WORK.
2434

:: $$$$$$$$$$$$$$$$$$$$$$$$$$$$ ERROR <<< $$$$$$$$$$$$$$$$$$$$$$$$$$$$
2437 010112 005077 171260 2$: CLR QASR ;:CLEAR THE CSR.
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450 010150 052777 000400 171220 1$: BIS #BIT8,QASP ;:2 GENERATE 1 MAINTENANCE OSC.
2451 ;:NOTE: AT 1MHZ, IT TAKES 10
2452 ;:MAINT. OSC TO EQUAL 1 COUNT

```

```

253 010156 005200 INC RD ;DO 20 MAINTENANCE OSC.
254 010160 001373 BNE RS
255
256 010162 017746 171210 MOV JASR, -(6) ; /-RDCLK-
257 010166 052777 004007 171202 BIS #4007, JASR ; /AVE CSR
258 ; /SET MODE 3, DIS INTR OSC NO RATE
259 ; /THIS MUST BE DONE IN
260 ; /ORDER TO XFERR COUNTER
261 ; /TO BUFFER ON ST2.
262 010174 052777 001000 171174 BIS #BIT9, JASR ; /GENERATE ON ST2 PULSE
263 010202 012746 000010 MOV #8, -(SP) ; /NOW GENERATE
264 010206 052777 000400 171162 64$: BIS #BIT8, JASR
265 010214 005316 DEC (SP) ; /EIGHT ST1 PULSES
266 010216 001373 BNE 64$
267 010220 005726 TST (SP)+
268 010222 017737 171152 001126 MOV JABR, SBDDAT ; /RESET STACK
269 ; /READ THE PRESET BUFFER,
270 ; /PREVIOUS COUNTER
271 010230 012677 171142 MOV (6)+, JASR ; /CONTENTS ARE IN SBDDAT.
272 010234 005737 001126 TST SBDDAT ; /RESTORE CSR
273 010240 001001 BNE 2$ ; YES - NEXT TEST.

```

::: \$ > ERROR <<< \$

```

274 010242 104011 ERROR 11 ;ERROR COULD NOT COUNT USING
275 ;MAINTENANCE OSC.

```

::: \$ ERROR <<< \$

```

276 010244 005077 171126 2$: CLR JASR ;CLEAR THE CSR.

```

```

::: *****
::: *TEST 56 *TEST THE CLOCK'S 1MHZ DIVIDER
::: *
::: *IN THIS TEST WE WILL CHECK OUT PART OF THE DIVIDER CHAIN LOGIC.
::: *THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS, THIS TEST CHECKS
::: *THAT 100,000 MAIN OSC PULSES GIVES US 10000. COUNTS AT 1MHZ RATE.
::: *****
↑ST56: SCOPE

```

```

277 010250 000004 000005 001160 MOV #5, $TIMES ;:DO 5 ITERATIONS
278 ; /-DIVCH-
279 010260 012700 000012 MOV #10, RD
280 010264 005077 171106 CLR JASR
281 010270 005077 171104 CLR JABR
282 010274 052777 004000 171074 BIS #BIT11, JASP ; /DISABLE INTERNAL OSC.
283 010302 052777 000017 171066 BIS #7:10, JASR ; /SET GO, RATE: 1MHZ., MODE 3.
284 010310 012701 023420 1$: MOV #10000, R1 ; /DO THAT MANY TIMES.
285 010314 052777 000400 171054 2$: BIS #BIT8, JASR ; /GENERATE AN OSC PULSE.
286 010322 005301 DEC R1

```



```

2507 010324 001373      BNE      2$
2508 010326 005300      DEC      R0
2509 010330 001367      BNE      1$
2510 010332 052777 001000 171036      BIS      #BIT9,2ASR      : ST2
2511 010340 012700 000010      MOV      #8, R0
2512 010344 052777 000400 171024 3$:      BIS      #BIT8,2ASR
2513 010352 005300      DEC      R0
2514 010354 001373      BNE      3$
2515
2516 010356 017737 171016 001126      MOV      2ABR,$BDDAT      : READ COUNT.
2517 010364 012737 023420 001424      MOV      #1000,$STMP0      : EXPECT THESE MANY COUNTS.
2518 010372 023737 001424 001126      CMP      $STMP0,$BDDAT      : DID WE GET THEM??
2519 010400 001401      BEQ
2520 010402 104011      TST57      :
2521      ERROR      11      : ERROR 100,000 OSC PULSES
2522      : DID NOT GENERATE 1000.
2523      : COUNTS AT RATE .MHZ

```

```

:*****
:TEST 57      *TEST THE CLOCK'S 100KHZ DIVIDER
:
:*IN THIS TEST WE WILL CHECK OUT PART OF THE DIVIDER CHAIN LOGIC.
:*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS, THIS TEST CHECKS
:*THAT 100,000 MAIN OSC PULSES GIVES US 1000. COUNTS AT 100KHZ RATE.
:*****

```

```

2532 010404 000004      TST57: SCOPE
2533 010406 012737 000005 001160      MOV      #5,$TIMES      : DO 5 ITERATIONS
2534
2535      : -DIVCH-
2536
2537 010414 012700 000012      MOV      #10, R0
2538 010420 005077 170752      CLR      2ASR
2539 010424 005077 170750      CLR      2ABR
2540 010430 052777 004000 170740      BIS      #BIT11,2ASR      : DISABLE INTERNAL OSC.
2541 010436 052777 000027 170732      BIS      #7!20,2ASR      : SET GO.RATE: 100KHZ..MODE 3.
2542
2543 010444 012701 023420      MOV      #10000, R1      : DO THAT MANY TIMES.
2544 010450 052777 000400 170720 1$:      BIS      #BIT8,2ASR      : GENERATE AN OSC PULSE.
2545 010456 005301      DEC      R1
2546 010460 001373      BNE      2$
2547 010462 005300      DEC      R0
2548 010464 001367      BNE      1$
2549 010466 052777 001000 170702      BIS      #BIT9,2ASR      : ST2
2550 010474 012700 000010      MOV      #8, R0
2551 010500 052777 000400 170670 3$:      BIS      #BIT8,2ASR
2552 010506 005300      DEC      R0
2553 010510 001373      BNE      3$
2554
2555 010512 017737 170662 001126      MOV      2ABR,$BDDAT      : READ COUNT.
2556 010520 012737 001750 001424      MOV      #1000,$STMP0      : EXPECT THESE MANY COUNTS.
2557 010526 023737 001424 001126      CMP      $STMP0,$BDDAT      : DID WE GET THEM??
2558 010534 001401      BEQ
2559 010536 104011      TST60      :
2560      ERROR      11      : ERROR 100,000 OSC PULSES
2561      : DID NOT GENERATE 1000.

```

: COUNTS AT RATE 100KHZ

```

2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572 010540 000004
2573 010542 012737 000005 001160
2574
2575
2576 010550 012700 000012
2577 010554 005077 170616
2578 010560 005077 170614
2579 010564 052777 004000 170604
2580 010572 052777 000037 170576
2581
2582 010600 012701 023420 170564 15:
2583 010604 052777 000400 170564 25:
2584 010612 005301
2585 010614 001373
2586 010616 005300
2587 010620 001367
2588 010622 052777 001000 170546
2589 010630 012700 000010
2590 010634 052777 000400 170534 35:
2591 010642 005300
2592 010644 001373
2593
2594 010646 017737 170526 001126
2595 010654 012737 000144 001424
2596 010662 023737 001424 001126
2597 010670 001401
2598 010672 104011
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612 010674 000004
2613 010676 012737 000005 001160
2614

```

```

*****
*TEST 60 *TEST THE CLOCK'S 100KHZ DIVIDER
*
*IN THIS TEST WE WILL CHECK OUT PART OF THE DIVIDER CHAIN LOGIC.
*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS, THIS TEST CHECKS
*THAT 100,000 MAIN OSC PULSES GIVES US 100. COUNTS AT 100KHZ RATE.
*****
TST60: SCOPE
MOV #5, $TIMES ;;DO 5 ITERATIONS
; -DIVCH-
MOV #10, $R0
CLR $ASR
CLR $ABR
BIS #BIT11, $ASR ; DISABLE INTERNAL OSC.
BIS #7!30, $ASR ; SET GO, RATE: 100KHZ., MODE 3.
MOV #10000, $R1 ; DO THAT MANY TIMES.
BIS #BIT8, $ASR ; GENERATE AN OSC PULSE.
DEC $R1
BNE $Z$
DEC $R0
BNE $Z$
BIS #BIT9, $ASR ; ST2
MOV #8, $R0
BIS #9!18, $ASR
DEC $R0
BNE $Z$
MOV $ABR, $BDDAT ; READ COUNT.
MOV #100, $TMPD ; EXPECT THESE MANY COUNTS.
CMP $TMPD, $BDDAT ; DID WE GET THEM?
BEQ TST61 ;;
ERROR 11 ; ERROR 100,000 OSC PULSES
; DID NOT GENERATE 100.
; COUNTS AT RATE 100KHZ

```

```

*****
*TEST 61 *TEST THE CLOCK'S 1KHZ DIVIDER
*
*IN THIS TEST WE WILL CHECK OUT PART OF THE DIVIDER CHAIN LOGIC.
*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS, THIS TEST CHECKS
*THAT 100,000 MAIN OSC PULSES GIVES US 10. COUNTS AT 1KHZ RATE.
*****
TST61: SCOPE
MOV #5, $TIMES ;;DO 5 ITERATIONS

```

```

2615
2616 010704 012700 000012      MOV      #10.,R0
2617 010710 005077 170462      CLR      @ASR
2618 010714 005077 170460      CLR      @ABR
2619 010720 052777 004000      BIS      @BIT11,@ASP      ; DISABLE INTERNAL OSC.
2620 010726 052777 000047 170450      BIS      @7!40,@ASR      ; SET GC.RATE: 1KHZ.,MODE 3.
2621
2622 010734 012701 023420      ;$:      MOV      #10000.,R1      ; DO THAT MANY TIMES.
2623 010740 052777 000400 170430 2$:      BIS      @BIT8,@ASR      ; GENERATE AN OSC PULSE
2624 010746 005301          DEC      R1
2625 010750 001373          BNE     2$
2626 010752 005300          DEC      R0
2627 010754 001367          BNE     1$
2628 010756 052777 001000 170412      BIS      @BIT9,@ASR      ; 5*2
2629 010764 012700 000010      MOV      @8.,R0
2630 010770 052777 000400 170400 3$:      BIS      @BIT8,@ASR
2631 010776 005300          DEC      R0
2632 011000 001373          BNE     3$
2633
2634 011002 017737 170372 001126      MOV      @ABR,@BDDAT      ; READ COUNT.
2635 011010 012737 000012 001424      MOV      #10.,$TMP0      ; EXPECT THESE MANY COUNTS.
2636 011016 023737 001424 001126      CMP      $TMP0,@BDDAT      ; DID WE GET THEM??
2637 011024 001401          BEQ     TST62
2638 011026 104011          ERROR   11                ; ERROR 100,000 OSC PULSES
2639                                     ; DID NOT GENERATE IC.
2640                                     ; COUNTS AT RATE 1KHZ
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651 011030 000004          ;*****
2652 011032 012737 000005 001160      ;*TEST 62      *TEST THE CLOCK'S 100HZ DIVIDER
2653
2654                                     ;*
2655                                     ;*IN THIS TEST WE WILL CHECK OUT PART OF THE DIVIDER CHAIN LOGIC.
2656                                     ;*THERE ARE SEVERAL TESTS THAT ARE USED TO DO THIS THIS TEST CHECKS
2657                                     ;*THAT 100,000 MAIN OSC PULSES GIVES US 1 COUNTS AT 100HZ RATE.
2658                                     ;*****
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2651 SCOPE
2652 MOV      #5,$TIMES      ;;DO 5 ITERATIONS
2653
2654                                     ; -DIVCH-
2655
2656 011040 012700 000012      MOV      #10.,R0
2657 011044 005077 170326      CLR      @ASR
2658 011050 005077 170324      CLR      @ABR
2659 011054 052777 004000 170314      BIS      @BIT11,@ASP      ; /DISABLE INTERNAL OSC.
2660 011062 052777 000057 170306      BIS      @7!50,@ASR      ; /SET GC.RATE: 100HZ.,MODE 3.
2661
2662 011070 012701 023420      ;$:      MOV      #10000.,R1      ; DO THAT MANY TIMES.
2663 011074 052777 000400 170274 2$:      BIS      @BIT8,@ASR      ; GENERATE AN OSC PULSE.
2664 011102 005301          DEC      R1
2665 011104 001373          BNE     2$
2666 011106 005300          DEC      R0
2667 011110 001367          BNE     1$
2668 011112 052777 001000 170256      BIS      @BIT9,@ASR      ; 5*2
2669 011120 012700 000010      MOV      @8.,R0

```

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000008  
000009  
000010  
000011  
000012  
000013  
000014  
000015  
000016  
000017  
000018  
000019  
000020  
000021  
000022  
000023  
000024  
000025  
000026  
000027  
000028  
000029  
000030  
000031  
000032  
000033  
000034  
000035  
000036  
000037  
000038  
000039  
000040  
000041  
000042  
000043  
000044  
000045  
000046  
000047  
000048  
000049  
000050  
000051  
000052  
000053  
000054  
000055  
000056  
000057  
000058  
000059  
000060  
000061  
000062  
000063  
000064  
000065  
000066  
000067  
000068  
000069  
000070  
000071  
000072  
000073  
000074  
000075  
000076  
000077  
000078  
000079  
000080  
000081  
000082  
000083  
000084  
000085  
000086  
000087  
000088  
000089  
000090  
000091  
000092  
000093  
000094  
000095  
000096  
000097  
000098  
000099  
000100  
000101  
000102  
000103  
000104  
000105  
000106  
000107  
000108  
000109  
000110  
000111  
000112  
000113  
000114  
000115  
000116  
000117  
000118  
000119  
000120  
000121  
000122  
000123  
000124  
000125  
000126  
000127  
000128  
000129  
000130  
000131  
000132  
000133  
000134  
000135  
000136  
000137  
000138  
000139  
000140  
000141  
000142  
000143  
000144  
000145  
000146  
000147  
000148  
000149  
000150  
000151  
000152  
000153  
000154  
000155  
000156  
000157  
000158  
000159  
000160  
000161  
000162  
000163  
000164  
000165  
000166  
000167  
000168  
000169  
000170  
000171  
000172  
000173  
000174  
000175  
000176  
000177  
000178  
000179  
000180  
000181  
000182  
000183  
000184  
000185  
000186  
000187  
000188  
000189  
000190  
000191  
000192  
000193  
000194  
000195  
000196  
000197  
000198  
000199  
000200

052777 000400 170244 35:  
005300  
001373  
  
017737 170236 001126  
012737 000001 001424  
023737 001424 001126  
001401  
104011

BIS #BIT8, DADR  
DEC RO  
BNE 35  
  
MOV DADR, \$BDDAT  
MOV #1, \$TMP0  
CMP \$TMP0, \$BDDAT  
BEQ TST63  
ERROR 11

:: READ COUNT.  
:: EXPECT THESE MANY COUNTS.  
:: DID WE GET THEM?  
:  
:  
:: ERROR 100,000 OSC PULSES  
:  
:  
:: DID NOT GENERATE 1  
:  
:  
:: COUNTS AT RATE 100HZ

::\*\*\*\*\*  
\*TEST 63 \*TEST THE CLOCK'S MODE 2 OPERATION

\*IN THIS TEST WE'LL CHECK MODE 2 OPERATION  
\*MODE 2: EXTERNAL EVENTS TIMING MODE  
\*SETTING THE GO BIT CAUSES THE COUNTER TO BEGIN COUNTING FROM  
\*ZERO AND TO FREE-RUN UNTIL THE GO BIT IS WRITTEN  
\*TO A ZERO THE COUNTER WILL CONTINUE COUNTING AFTER  
\*OVERFLOW. AN EXTERNAL PULSE FROM SCHMITZ TRIGGER 2  
\*(WHEN ST2 GO ENABLE IS A "0") CAUSES DATA TO  
\*TRANSFER FROM THE COUNTER TO THE BUFFER/PRESET REG.  
\*WHILE THE COUNTER CONTINUES TO RUN.  
\*  
\*TO TEST THIS MODE, WE'LL DISABLE THE INTERNAL OSC AND USE  
\*MAINTENANCE OSC PULSES AS WELL AS A MAINTENANCE  
\*ST2.

::\*\*\*\*\*  
TST63: SCOPE

MOV #20, \$TIMES ;: DO 20 ITERATIONS  
  
CLR DADR ;: CLEAR THE CSR.  
CLR DADR ;: CLEAR THE PRESET REG.  
MOV #60, DADR ;: SET RATE ST1  
BIS #5, DADR ;: START CLOCK MODE 2.  
  
MOV #2, RO ;: SET TO GIVE 2 ST1 PULSES.  
BIS #BIT8, DADR ;: GENERATE AN ST1 PULSE  
INC RO  
BNE 25 ;: IF NOT DONE 2 TIMES, LOOP.

35:  
BIS #BIT9, DADR ;: HERE'S THE BIGGIE! AN ST2 HAS BEEN GENERATED  
MOV #2, \$GDDAT ;: THE PRESET BUFFER SHOULD BE 2.  
MOV DADR, \$BDDAT ;: READ THE PRESET BUFFER.  
CMP \$BDDAT, \$GDDAT ;: DID A COUNTER TO PRESET BUFFER OCCUR?  
BEQ 45 ;: YES - NEXT SUBTEST.

:: \$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$ ERROR << \$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$\$



IN THIS TEST WE'LL CHECK MODE 3 OPERATION.  
MODE 3 IS JUST LIKE MODE 2 EXCEPT THAT THE  
REG IS ZEROED AFTER AN ST2.

2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828

011442 000004  
011444 012737 000020 0C1160  
011452 005077 167720  
011456 005077 167716  
011462 012777 000060 167706  
011470 052777 000007 167700  
011476 012700 177776  
011502 052777 000400 167666  
011510 005200  
011512 001373  
011514  
011514 052777 001000 167654  
C11522 012737 000002 001124  
011530 017737 167644 001126  
011536 023737 001126 001124  
C11544 001402  
011546 104005  
011550 000440  
011552 005037 001124  
011556 017746 167614  
011562 052777 000005 167606  
011570 042777 100000 167600  
011576 052777 001000 167572  
011604 017737 167570 0C1126  
011612 052677 167560  
011616 005737 001126  
C11622 001402  
C11624 104005

\*\*\*\*\*  
TST64: SCOPE  
MOV #20,\$TIMES ;;DO 20 ITERATIONS  
CLR JASR ;CLEAR THE CSR.  
CLR JABR ;CLEAR THE BUFFER REG  
MOV #60,JASR ;SET RATE: ST1  
BIS #7,JASR ;START CLOCK: MODE 3  
15: MOV #-2,RO ;SET TO GIVE 2 ST1 PULSE  
25: BIS #BIT8,JASR ;GENERATE AN ST1 PULSE  
INC RO  
BNE 25 ;IF NOT DONE 2 TIMES, LOOP.  
35: BIS #BIT9,JASR ;HERE'S THE BIGGIE! AN ST2 HAS BEEN GENERATED  
MOV #2,\$GDDAT ;THE PRESET BUFFER SHOULD BE 2.  
MOV JABR,\$BDDAT ;READ THE PRESET BUFFER.  
CMP \$BDDAT,\$GDDAT ;DID A COUNTER TO PRESET BUFFER OCCUR?  
BEQ 45 ;YES - NEXT SUBTEST.  
:: \*\*\*\*\* ERROR <<< \*\*\*\*\*  
2804 011546 104005 ERROR 5 ;A COUNTER TO PRESET BUFFER DID NOT  
2805 ;HAPPEN PROPERLY.  
2806  
:: \*\*\*\*\* ERROR <<< \*\*\*\*\*  
2809 011550 000440 BR TST65 ;;  
2810  
2811 011552 005037 001124 45: CLR \$GDDAT ;EXPECT ZERO BACK FROM COUNT REG.  
2812 ; -RDCLK1-  
2813 011556 017746 167614 MOV JASR, -(SP) ;SAVE CSR CONTENTS.  
2814 011562 052777 000005 167606 BIS #5,JASR ;SET TO MODE 2,GO  
2815 011570 042777 100000 167600 BIC #BIT15,JASR ;CLR ST FLAG.  
2816 011576 052777 001000 167572 BIS #BIT9,JASR ;GENERATE ST2 PULSE.  
2817 011604 017737 167570 0C1126 MOV JABR,\$BDDAT ;READ COUNT REG.  
2818 011612 052677 167560 BIS (SP)+,JASR ;RESTORE CSR.  
2819 011616 005737 001126 TST \$BDDAT ;PREVIOUS CONTENTS OF COUNT REG  
2820 ; IN \$BDDAT.  
2821 C11622 001402 BEQ 55 ;IF SC - NEXT TEST.  
2822  
:: \*\*\*\*\* ERROR <<< \*\*\*\*\*  
2825 011624 104005 ERROR 5 ;THE CLOCK FORGOT TO ZERO THE COUNT  
2826 ;REG. AFTER AN ST2 OCCURRED ON  
2827 ;A MODE 3 COUNT.  
2828  
:: \*\*\*\*\* ERROR <<< \*\*\*\*\*



2885 012150 023737 001124 001126  
2886 012156 001401  
2887

CMP \$GDDAT,\$BDDAT ;OK  
BEQ TST66 ;:

:::\*\*\*\*\*  
:::\*\*\*\*\* ERROR <<<\*\*\*\*\*

2890 012160 104001  
2891

ERROR 1 ;ST1 AND/OR ST2 FLAG FAILED TO SET

:::\*\*\*\*\* ERROR <<<\*\*\*\*\*

2894  
2895  
2896  
2897  
2898  
2899  
2900  
2901

:::\*\*\*\*\*  
: TEST 66 \*DWARF TEST OF OVERFLOW OUT,ST1 IN AND OUT,AND ST2 IN.  
: IN THIS TEST,WE'LL TEST OVERFLOW OUT,ST1 IN,ST1 OUT, AND  
: ST2 IN.  
:\*\*\*\*\*

2902 012162 000004  
2903 012164 012737 000010 001160  
2904 012172 012737 012344 0011C  
2905 012200 012737 012344 001110  
2906 012206 005737 001450  
2907 012212 001505  
2908 012214 104401 012222  
2909 012220 000426

TST66: SCOPE  
MOV #10,\$TIMES ;:DO 10 ITERATIONS  
MOV #15,\$LPADR  
MOV #15,\$IPERF  
TST DWARF ;DWARF TESTS SELECTED?  
BEQ TST67 ;:  
TYPE 65\$ ;:TYPE ASCIZ STRING  
BR 64\$ ;:GET OVER THE ASCIZ  
65\$: .ASCIZ <200><?>"DWARF: ALL SWITCHES OFF,S2-2 AND S2-3 ON"  
64\$:

2910  
2911 012276  
2912 012276 104401 012304  
2913 012302 000416  
2914

66\$: .ASCIZ <200>"PANEL: SAME AS LAST TEST"<?>  
67\$:

2915 012340  
2916 012340 004737 014744  
2917 012344 005077 167026  
2918 01235C 005737 001450  
2919 012354 001424  
2920 012356 012777 177777 167014  
2921 012364 012777 000011 167004  
2922 012372 000240  
2923 012374 000240  
2924 01237C 000240

1\$: JSR PC,ANY2  
CLR @ASR  
TST DWARF  
BEQ TST67 ;:  
MOV #-1,@ABR ;:PRESET FOR OVERFLOW.  
MOV #11,@ASR ;:SET 1MHZ.GO.  
NOP  
NOP  
NOP

2925 012400 012737 102210 001124  
2926 012406 017737 166764 001126  
2927 012414 023737 001124 001126  
2928 012422 001401  
2929

MOV #102210,\$GDDAT ;:EXPECT ST1,ST2 FLAGS SET.  
MOV @ASR,\$BDDAT ;:READ CSR.  
CMP \$GDDAT,\$BDDAT  
BEQ TST67 ;:

:::\*\*\*\*\* ERROR <<<\*\*\*\*\*

2932 012424 104001  
2933

ERROR 1 ;ST1 AND/OR ST2 FLAG(S) FAIL TO SET.

:::\*\*\*\*\* ERROR <<<\*\*\*\*\*

2936  
2937  
2938

:::\*\*\*\*\*  
: TEST 67 \*IF ENABLED,CHECK THRESHOLD ST1 FROM TESTOR



```

2939 012426 000004
2940 012430 012737 000002 001160      MOV      #2,$TIMES          ;;DO 2 ITERATIONS
2941 012436 005737 001444          TST      EX$              ;;OPERATING IN TESTOR MODE?
2942 012442 001002          BNE      4$              ;;YES DO THIS TEST.
2943 012444 000137 013644          JMP      ENDF            ;;NO-END PASS
2944 012450 005737 001450          TST      DWARF           ;;DWARF MODE??
2945 012454 001452          BEQ      40$            ;;QUESTION ALLREADY BEEN ASKED?
2946 012456 005737 001452          TST      ASK             ;;QUESTION ALLREADY BEEN ASKED?
2947 012462 001032          BNE      35$            ;;QUESTION ALLREADY BEEN ASKED?
2948 012464 104401 012472          TYPE     ,65$           ;;TYPE ASCIZ STRING
2949 012470 000421          BR       64$           ;;GET OVER THE ASCIZ
2950 012534 104410          .ASCIZ  '<200> #15 VOLT SUPPLY TO DWARF' (Y OR N) #
2951 012536 012637 001452          RDCHR          (SP)+,ASK
2952 012542 042737 000240 001452          MOV      #240,ASK       ;;STRIP PARITY AND LOWER CASE.
2953 012550 123727 001452 000131          CMPB    ASK,#'Y         ;;DID HE ANSWER YES?
2954 012556 001411          BEQ      40$
2955 012560 123727 001452 000116          CMPB    ASK,#'N
2956 012566 001403          BEQ      39$
2957 012570 005037 001452          CLR     ASK
2958 012574 000725          BR       4$
2959 012576 000137 013644          JMP      ENDF
2960 012602 104401 012610          TYPE     ,67$           ;;TYPE ASCIZ STRING
2961 012606 000421          BR       66$           ;;GET OVER THE ASCIZ
2962 012652 104401 012660          .ASCIZ  '<200> #PANEL: ST1 +ST2 POTS OUT AND CCW#
2963 012652 000424          TYPE     ,69$           ;;TYPE ASCIZ STRING
2964 012656 000424          BR       68$           ;;GET OVER THE ASCIZ
2965 012730 005077 166442          .ASCIZ  '<200> #DWARF: S2-7 AND S2-8 ON, ALL OTHERS OFF#
2966 012734 004737 014744          CLR     JASR           ;;CLEAR CSR
2967 012740 005077 166432          JSR    PC,ANY2
2968 012744 004737 014704          CLR     JASR
2969 012750 017737 166422 001126          JSR    PC,ANYKEY
2970 012756 012737 000000 001124          MOV     JASR,$BDDAT    ;;READ CSR
2971 012764 032737 102000 001126          MOV     #0,$GDDAT
2972 012772 001401          BIT     #BIT15:BIT10,$BDDAT ;;DID ANY FLAG SET?
2973 012774 104002          BEQ     TST70          ;;
2974 012774 104002          ERROR 2              ;;ST1 OR ST2 THRESHOLD LEVEL ERROR
2975 012774 104002              ;;FLAGS SHOULD NOT HAVE SET!
2976 *****
2977 TEST 70          *ST1,ST2 THRESHOLD TEST #2,POTS CW
2978 *****
2979 $T70:  SCOPE
2980 MOV     #1$, $LPERR
2981 MOV     #1$, $LPADR
2982
2983
2984
2985
2986
2987
2988
2989 012776 000004
2990 013000 012737 013050 001110
2991 013006 012737 013050 001106
2992

```

2993 013014 104401 013022  
 2994 013020 000413  
 2995  
 2996 013050  
 2997 013050 005077 166322  
 2998 013054 004737 014744  
 2999 013060 005077 166312  
 3000 013064 004737 014704  
 3001 013070 017737 166302 001126  
 3002 013076 032737 102000 001126  
 3003 013104 001401  
 3004 013106 104002  
 3005  
 3006  
 3007  
 3008  
 3009 013110 000004  
 3010 013112 C12737 013200 001110  
 3011 013120 012737 013200 001106  
 3012 013126 104401 013134  
 3013 013132 000422  
 3014  
 3015 013200  
 3016 013200 005077 166172  
 3017 013204 004737 014744  
 3018 013210 005077 166162  
 3019 013214 004737 014704  
 3020 013220 017737 166152 001126  
 3021 013226 012737 102000 001124  
 3022 013234 042737 075777 001126  
 3023 013242 023737 001124 001126  
 3024 013250 001401  
 3025 013252 104002  
 3026  
 3027  
 3028  
 3029  
 3030 013254 000004  
 3031 013256 012737 000010 001150  
 3032  
 3033  
 3034 013264 005737 001444  
 3035 013270 001002  
 3036 013272 000137 013644  
 3037 013276 005737 001450  
 3038 013302 001373  
 3039 013304 012737 013514 001110  
 3040 013312 012737 013514 001106  
 3041 013320 104401 013326  
 3042 013324 000416  
 3043  
 3044 013362  
 3045 013362 104401 013370  
 3046 013366 000423

```

TYPE 65$          ;;TYPE ASCIZ STRING
BR 64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <200>#PANEL: TURN POTS CW#
64$:
1$: CLR 2ASR
   JSR PC,ANY2
   CLR 2ASH
   JSR PC,ANYKEY
   MOV 2ASR,$BDDAT
   BIT 15:BIT10,$BDDAT      ;DID ANY FLAG SET?
   BEQ TST71                ;;
   ERROR 2                  ;ST1 OR ST2 THRESHOLD ERROR.

*****
;*TEST 71 *ST1,ST2 THRESHOLD TEST #3 MID RANGE
*****
↑ST71: SCOPE
      MOV #1$,$LPERR
      MOV #1$,$LPADR
      TYPE 65$          ;;TYPE ASCIZ STRING
      BR 64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <200>#PANEL: SET ST1,ST2 POTS MID-RANGE.#
64$:
1$: CLR 2ASR
   JSR PC,ANY2
   CLR 2ASR
   JSR PC,ANYKEY
   MOV 2ASR,$BDDAT
   MOV 15:BIT10,$GDDAT
   BIC 075777,$BDDAT
   CMP $GDDAT,$BDDAT      ;AT MID RANGE THEY BOTH SHOLD SET.
   BEQ TST72                ;;
   ERROR 2                  ;ST1 OR ST2 FAILED TO SET.

*****
;*TEST 72 *TEST CLOCK REPEATABILITY IF ON TESTOR
*****
↑ST72: SCOPE
      MOV #10,$TIMES      ;;DO 10 ITERATIONS

      TST EXS             ;TESTOR MODE EXABLED??
      BNE 10$
      JMP ENDP            ;NO REPORT END PASS.
10$: TST DWARF
      BNE 9$
      MOV #1$,$LPERR
      MOV #1$,$LPADR
      TYPE 65$          ;;TYPE ASCIZ STRING
      BR 64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <200>#PANEL: ST1 POT OUT AND CW#
64$:
      TYPE 67$          ;;TYPE ASCIZ STRING
      BR 66$           ;;GET OVER THE ASCIZ

```

```

3047      ;:67S: .ASCIZ <200 # ST2 POT IN AND SLOPE OUT - #
3048 013436      66S:
3049 013436 104401 013444      TYPE 69S      ;;TYPE ASCIZ STRING
3050 013442 000422      BR 68S      ;;GET OVER THE ASCIZ
3051      ;:69S: .ASCIZ <200 #DWARF: S2 ALL SWITCHES OFF,S2-6 ON#
3052 013510      68S:
3053 013510 004737 014744      JSR PC,ANY2
3054 013514 012777 020016 165654 1S: MOV #BIT13!16,2ASR ;SET 1MHZ MODE 3,ST2 GO ENABLE. TEST CLOCK
3055 013522 005077 165672      CLR 2TSCLC ;CLEAR STATUS REG.
3056 013526 012777 100000 165666      MOV #100000,2TSCLO ;PRESET COUNT REG.
3057 013534 012777 000013 165656      MOV #13,2TSCLC ;SET 1MHZ MODE 1,GO
3058 013542 105777 165652 2S: TSTB 2TSCLC ;WAIT FOR CLOCK OVERFLOW.
3059 013546 100375      BPL 2S
3060 013550 042777 100000 165620      BIC #BIT15,2ASR
3061
3062 013556 042777 000200 165634 3S: BIC #200,2TSCLC ;CLEAR OVERFLOW FLAG.
3063 013564 105777 165630      TSTB 2TSCLC ;WAIT FOR NEXT OVERFLOW.
3064 013570 100375      BPL 3S
3065
3066 013572 005077 165600      CLR 2ASR ;STOP CLOCK.
3067 013576 005077 165616      CLR 2TSCLC
3068 013602 017737 165572 001126      MOV 2ABR,$BDDAT ;READ RESULTS
3069 013610 012737 100000 001124      MOV #100000,$GDDAT ;S/B COUNT
3070 013616 013700 001124      MOV $GDDAT,RO
3071 013622 163700 001126      SUB $BDDAT,RO
3072 013626 100001      BPL 4S ;+DIF.
3073 013630 005100      COM RO ;OTHERWISE MAKE IT
3074 013632 020027 000007 4S: CMP RO,#7 ;SHOULD NOT VARY MORE THAN 7 COUNTS.
3075 013636 003401      BLE TS+73 ;:
3076
3077 013640 104010      ERROR 10 ;CLOCK REPEATABILITY ERROR.
3078
3079      ;:*****
3080      ;:TEST 73 END OF TESTS
3081      ;:*****
3082 013642 000004      †ST73: SCOPE
3083      ;:
3084      ;:NOW WE'LL DETERMINE IF WE ARE ALLOWED TO AUTO-SIZE.
3085      ;:IF SO, WE'LL FIND OUT IF THERE ARE OTHER CLOCKS OUT THERE
3086      ;:TO TEST.
3087      ;:
3088      ;:
3089      ENDP:
3090 013644 000005      RESET
3091
3092
3093 013646 105737 001215      TSTB $ENVM ;SEE IF APT WILL LET UP AUTO-SIZE.
3094 013652 100547      BMI 4S ;NO - EXIT.
3095
3096
3097 013654 023737 001440 001442      CMP MDEVCT,TSTCNT ;TESTED MAX. UNITS?
3098 013662 001543      BEQ 4S ;YES EXIT.
3099 013664 006337 001432      ASL ROTATE ;POINT NEXT UNIT.
3100 013670 005237 001440      INC MDEVCT

```

```

3101 013674 062737 000004 001376 ADD #4,ASR ;YES, ADD TO BASE ADDR.
3102 013702 013746 000004 MOV ERRVEC, (6 ;SAVE CONTENTS OF LOC 4.
3103 013706 012737 014154 000004 MOV #15,ERRVEC ;SET UP IN CASE NO MORE CLOCKS.
3104 013714 005777 165456 TST QASR ;TIME OUT HERE IF NO MORE CLOCKS.
3105 013720 005737 001202 TST $PASS ;IF HERE, ANOTHER CLOCK FOUND.
3106 013724 001003 BNE 3$ ;IS THIS 1ST PASS?
3107 013726 053737 001432 001434 BIS ROTATE,UTEST ;NO-GET OUT.
3108 013734 104401 013742 3$: TYPE 65$ ;:TYPE ASCIZ STRING
3109 013740 000405 BR 64$ ;:GET OVER THE ASCIZ
3110 013754 313746 001204 64$: .ASCIZ '15 (12) UNIT #' ;:
3111 013754 104402 MOV $DEVCT, -(SP) ;:SAVE $DEVCT FOR TYPEOUT
3112 013760 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3113 013762 104401 013770 TYPE 67$ ;:TYPE ASCIZ STRING
3114 013766 000405 BR 66$ ;:GET OVER THE ASCIZ
3115 014002 66$: .ASCIZ # ADDR= # ;:
3116 014002 162737 000004 001376 SUB #4,ASR ;:SAVE ASR FOR TYPEOUT
3117 014010 013746 001376 MOV ASR, -(SP) ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3118 014014 104402 TYPOC ;:
3119 014016 062737 000004 001376 ADD #4,ASR ;:TYPE ASCIZ STRING
3120 014024 104401 014032 TYPE 69$ ;:GET OVER THE ASCIZ
3121 014030 000406 BR 68$ ;:
3122 014046 68$: .ASCIZ # VECTOR= # ;:
3123 014046 013746 001402 MOV VECT1, -(SP) ;:SAVE VECT1 FOR TYPEOUT
3124 014052 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3125 014054 104401 014062 TYPE 71$ ;:TYPE ASCIZ STRING
3126 014060 000406 BR 70$ ;:GET OVER THE ASCIZ
3127 014076 70$: .ASCIZ " COMPLETED " ;:
3128 014076 005237 001204 INC $DEVCT ;:TYPE ASCIZ STRING
3129 014102 104401 014110 TYPE 73$ ;:GET OVER THE ASCIZ
3130 014106 000410 BR 72$ ;:
3131 014130 72$: .ASCIZ " TESTING UNIT #" ;:
3132 014130 013746 001204 MOV $DEVCT, -(SP) ;:SAVE $DEVCT FOR TYPEOUT
3133 014134 104402 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
3134 014136 012637 000004 MOV (6)+,ERRVEC ;:RESET LOC 4.
3135 014142 062737 000010 001402 ADD #10,VECT1 ;:UPDATE VECTOR ADDR.
3136 014150 000137 002312 JMP LOOP ;:TEST NEW UNIT.
3137 014154 1$: ADD #4,SP ;:RESTORE /ADD #4 TO STACK POINTER.
3138 014154 062706 000004 MOV (6)+,ERRVEC ;:RESTORE LOC 4
3139 014160 012637 000004 SUB #4,ASR
3140 014164 162737 000004 001376
3141 014172 4$:

```

```

3155 014172 104401 014200          TYPE      755          ;;TYPE ASCIZ STRING
3156 014176 000405          BR        745          ;;GET OVER THE ASCIZ
3157          ;;755: .ASCIZ  (15)12 "UNIT"
3158 014212          MOV      $DEVCT,- SP)  ;;SAVE $DEVCT FOR TYPEOUT
3159 014212 013746 001204          TYPOC     ;;GO TYPE--OCTAL ASCII ALL DIGITS
3160 014216 104402          TYPE      775          ;;TYPE ASCIZ STRING
3161 014220 104401 014226          BR        765          ;;GET OVER THE ASCIZ
3162 014224 000405          ;;775: .ASCIZ  # ADDR= #
3163          ;;765:
3164 014240          MOV      ASR,- SP)    ;;SAVE ASR FOR TYPEOUT
3165 014240 013746 001376          TYPOC     ;;GO TYPE--OCTAL ASCII ALL DIGITS
3166 014244 104402          TYPE      795          ;;TYPE ASCIZ STRING
3167 014246 104401 014254          BR        785          ;;GET OVER THE ASCIZ
3168 014252 000406          ;;795: .ASCIZ  #: VECTOR= #
3169          ;;785:
3170 014270          MOV      VECT1,-(SP)  ;;SAVE VECT1 FOR TYPEOUT
3171 014270 013746 001402          TYPOC     ;;GO TYPE--OCTAL ASCII ALL DIGITS
3172 014274 104402          TYPE      815          ;;TYPE ASCIZ STRING
3173 014276 104401 014304          BR        805          ;;GET OVER THE ASCIZ
3174 014302 000412          ;;815: .ASCIZ  " TEST COMPLETED "
3175          ;;805:
3176 014330          25:  MOV      $BASE,ASR
3177          MOV      $VECT1,VECT1
3178 014330 013737 001250 001376          MOV      $DEVCT,LCNT
3179 014336 013737 001244 001402          INC      LCNT
3180 014344 013737 001204 001446          MOV      #0,$DEVCT
3181 014352 005237 001446          CLR      MDEVCT      ;BEGIN TESTING 1ST UNIT.
3182 014356 012737 000000 001204          MOV      #1,ROTATE   ;POINT TO IT.
3183          .SBTTL  END OF PASS ROUTINE
3184 014364 005037 001440          ;;*****
3185 014370 012737 000001 001432          ;;INCREMENT THE PASS NUMBER ($PASS)
3186          ;;IF THERES A MONITOR GO TO IT
3187          ;;IF THERE ISN'T JUMP TO LOOP
3188          $EOP:
3189          NOP
3190          CLR      $STNM      ;;ZERO THE TEST NUMBER
3191          CLR      $TIMES     ;;ZERO THE NUMBER OF ITERATIONS
3192          INC      $PASS      ;;INCREMENT THE PASS NUMBER
3193          BIC      #10000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
3194          DEC      (PC)+      ;;LOOP?
3195          $EOPCT: .WORD 1
3196          BGT      $DOAGN     ;;YES
3197          MOV      (PC)+,2(PC)+ ;;RESTORE COUNTER
3198          $SENDCT: .WORD 1
3199          MOV      $DOAGN
3200 014414 042737 100000 001202          ;;TYPE ASCIZ STRING
3201 014422 005327          BR        655          ;;GET OVER THE ASCIZ
3202 014424 000001          TYPE      655
3203 014426 003122          BR        645
3204 014430 012737          ;;
3205 014432 000001          ;;
3206 014434 014424          ;;
3207 014436 104401 014444          ;;
3208 014442 000406          ;;

```

```

3209
3210 014460
3211 014460 013746 001202
3212 014464 104402
3213 014466 104401 014474
3214 014472 000411
3215
3216 014516
3217 014516 013746 001436
3218 014522 104402
3219 014524 104401 014532
3220 014530 000407
3221
3222 014550
3223 014550 013746 001446
3224 014554 104402
3225 014556 104401 014564
3226 014562 000411
3227
3228 014606
3229 014606 104401 014614
3230 014612 000415
3231
3232 014646
3233 014646 013746 001434
3234 014652 104405
3235 014654 013700 000042
3236 014660 001405
3237 014662 000005
3238 014664 004710
3239 014666 000240
3240 014670 000240
3241 014672 000240
3242 014674
3243 014674 000137
3244 014676 002312
3245 014700 377 377 000
3246 014704
3247
3248
3249
3250
3251
3252
3253 014704 105777 164236
3254 014710 104401 014716
3255 014714 000413
3256
3257 014744
3258 014744
3259 014744 104401 014752
3260 014750 000417
3261
3262 015010

::655: .ASCIZ .15<<12>>#ENDPASS #
645:
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE 675 ;;TYPE ASCIZ STRING
BR 665 ;;GET OVER THE ASCIZ
::675: .ASCIZ # OCTAL ERRORS #
665:
MOV ERCNT,-(SP) ;;SAVE ERCNT FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE 695 ;;TYPE ASCIZ STRING
BR 685 ;;GET OVER THE ASCIZ
::695: .ASCIZ #; THERE ARE #
685:
MOV LCNT,-(SP) ;;SAVE LCNT FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE 715 ;;TYPE ASCIZ STRING
BR 705 ;;GET OVER THE ASCIZ
::715: .ASCIZ # (OCTAL) UNITS.#
705:
TYPE 735 ;;TYPE ASCIZ STRING
BR 725 ;;GET OVER THE ASCIZ
::735: .ASCIZ <200>#THE GOOD UNITS (L TO R) #
725:
MOV UTEST,-(SP) ;;SAVE UTEST FOR TYPEOUT
TYPBN ;;GO TYPE--BINARY ASCII
$GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
BEQ $DOA9N ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
$DOA9N: JMP @PC)+ ;;RETURN
$RTNAD: .WORD LOOP
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
.EVEN

;
; THIS ROUTINE TYPES LAST MESSAGE AND WAITS FOR AN OPERATOR
; RESPONSE.
;
ANYKEY: TSTB @STKB ;;CLEAR TTY READY FLAG.
TYPE 655 ;;TYPE ASCIZ STRING
BR 645 ;;GET OVER THE ASCIZ
::655: .ASCIZ <200><7> #SWITCH ST1 3 TIMES#
645:
ANY2: TYPE 655 ;;TYPE ASCIZ STRING
BR 645 ;;GET OVER THE ASCIZ
::655: .ASCIZ <200><7> #TYPE ANY KEY WHEN DONE...# 7
645:

```

3263  
3264  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278  
3279  
3280  
3281  
3282  
3283  
3284  
3285  
3286  
3287  
3288  
3289  
3290  
3291  
3292  
3293  
3294  
3295  
3296  
3297  
3298  
3299  
3300  
3301  
3302  
3303  
3304  
3305  
3306  
3307  
3308  
3309  
3310  
3311  
3312  
3313  
3314  
3315  
3316  
3317  
3318  
3319  
3320

015010 105777 164130  
015014 100375  
015016 105777 164124  
015022 104401 015030  
015026 000401  
  
015032  
015032 000207

IS: TSTB 2STKS ;WAIT FOR OPERATOR.  
BPL 18  
TSTB 2STKB ;CLEAR TTY READY FLAG.  
TYPE 678 ;TYPE ASCII STRING  
BR 668 ;GET OVER THE ASCII  
;678: .ASCIIZ 200 ##  
668: RTS PC

.SBTTL ;I/O SIGNAL TEST #1 ST1 IN AND ST2 OUT IN AND OUT

SWITCH PACK S2 MUST BE SET UP AS FOLLOWS:

- SWITCH 1 - OFF
- 2 - ON
- 3 - OFF
- 4 - OFF
- 5 - ON
- 6 - ON
- 7 - NOT USED

THIS SELECTS TTL THRESHOLDS AND POSITIVE SLOPE FOR SCHMITT TRIGGER 1.

PLEASE REMOVE ANY PREVIOUS JUMPER.

JUMPER THE FOLLOWING PINS TOGETHER:

J1 - S5 (ST2 OUT) TO J1 - VV (ST1-IN)

LOAD AND START AT LOCATION 210  
END PASSES OCCUR IMMEDIATELY AND ARE NOT REPORTED  
ERRORS ARE REPORTED AS IN THE REGULAR LOGIC TEST AND  
THEIR PRINTOUT MAY BE INHIBITED

015034 104407  
015036 005077 164334  
015042 005077 164332  
015046 012777 000061 164322  
015054 052777 001000 164314  
015062 012777 000005 164306  
015070 052777 001000 164300  
015076 027727 164276 000001  
015104 001753  
015106 104000  
015110 000751

IOTST1: CKSWR ;CHECK THE SWR  
IS: CLR 2ASR ;CLEAR THE CSR  
CLR 2ABR ;CLEAR THE BUFFER REG.  
MOV #61, 2ASR ;RATE ST1, MODE 0, GO.  
BIS #BIT9, 2ASR ;GENERATE A MAINTENANCE ST2.  
MOV #5, 2ASR ;NOW SET TO READ COUNT REG  
BIS #BIT9, 2ASR ;FORCE COUNT -> BUFFER REG.  
CMP 2ABR, #1 ;DID COUNT REG ADVANCE ONCE?  
BEQ IOTST1 ;YES - LOOP.  
ERROR ;ST2 OUT TO ST1 IN FAILED.  
BR IOTST1

.SBTTL ;I/O SIGNAL TEST #2 CLOCK OVFLOW OUT TEST.

015112 104407  
015114 005077  
015120 012777  
015126 012777  
015134 052777  
015142 000240  
015144 000240  
015146 005777  
015152 104200  
015158 000755

164256  
177777  
000063  
000400  
164252  
164242  
164234  
164224

IOTST2:  
:SBR

: SWITCH PACK 52 MUST BE SET UP AS FOLLOWS:  
SWITCH 1 - OFF  
2 - OFF  
3 - OFF  
4 - ON  
5 - ON  
6 - OFF  
7 - NOT USED

: THIS SELECTS TTL THRESHOLDS AND POSITIVE SLOPE FOR  
: SCHMITT TRIGGER 2.  
: PLEASE REMOVE ANY PREVIOUS JUMPER.  
: JUMPER THE FOLLOWING PINS TOGETHER:  
: J1 - RR (CLK OV) TO J1 - TT ST2-IN  
: LOAD AND START AT LOCATION 214.  
: END PASSES OCCUR IMMEDIATELY AND ARE NOT REPORTED.  
: ERRORS ARE REPORTED AS IN TH REGULAR LOGIC TEST AND  
: THEIR PRINTOUT MAY BE INHIBITED.

: CKSWR : CHECK THE SWR.  
: CLR : CLEAR THE CSR.  
: MOV #1, JABR : PRELOAD PRESET BUFFER.  
: MOV #63, JASR : RATE ST1, MODE 1, GO.  
: BIS #BIT8, JASR : GENERATE A MAIN. ST1.  
: NOP  
: NOP  
: IST JASR : DID OVERFLOW SET ST2 FLAG  
: BMI IOTST2 : YES - LOOP  
: ERROR : CLK OV OUT TO ST2 IN FAILED.  
: BR IOTST2 : LOOP

: I O SIGNAL TEST #3 ST1 OUT AND ST2 IN

: SWITCH PACK 52 MUST BE SET UP AS FOLLOWS:  
SWITCH 1 - OFF  
2 - OFF  
3 - OFF  
4 - ON  
5 - ON  
6 - ON  
7 - NOT USED

: THIS SELECTS TTL THRESHOLD AND POSITIVE SLOPE FOR  
: SCHMITT TRIGGER 2.



```

: PLEASE REMOVE ANY PREVIOUS JUMPERS.
: JUMPER THE FOLLOWING PINS TOGETHER:
: J1 - UU (ST1 OUT) TO J1 - TT (ST2-IN)
: LOAD AND START AT LOCATION 220
: END PASSES OCCUR IMMEDIATELY AND ARE NOT REPORTED
: ERRORS ARE REPORTED AS IN THE REGULAR LOGIC TEST AND
: THEIR PRINTOUT MAY BE INHIBITED

```

```

015160 104407 IOTST3: CKSWR ; CHECK THE SWR
015162 012777 000001 164206 MOV #1, DADR ; SET GO BIT.
015170 052777 000400 164200 BIS #BIT8, DADR ; GENERATE A MAIN, ST1.
015176 005777 164174 TST DADR ; DID ST2 FLAG SET?
015202 100401 BMI IS ;
015204 104000 ERROR ; ST1 OUT TO ST2 IN FAILED

015206 032777 010000 164162 IS: BIT #BIT12, DADR ; DID "FOR" BIT SET?
015214 001761 BEQ IOTST3 ; NO - GOOD!
015216 104000 ERROR ; "FOR" BIT SET ON ONLY 1 ST2.
015220 000757 BR IOTST3 ; LOOP

```

```

.SBTTL
.SBTTL *SYSMAC ROUTINES
.SBTTL

```

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

```

: *****
: THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
: OCTAL (ASCII) NUMBER AND TYPE IT.
: $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
: CALL:
:   MOV     NUM, -(SP)      ; NUMBER TO BE TYPED
:   TYPOS   ; CALL FOR TYPEOUT
:   .BYTE  N                ; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
:   .BYTE  M                ; M=1 OR 0
:                               ; 1=TYPE LEADING ZEROS
:                               ; 0=SUPPRESS LEADING ZEROS
: $TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
: $TYPOS OR $TYPOC
: CALL:
:   MOV     NUM, -(SP)      ; NUMBER TO BE TYPED
:   TYPON   ; CALL FOR TYPEOUT
: $TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
: CALL:
:   MOV     NUM, -(SP)      ; NUMBER TO BE TYPED
:   TYPOC   ; CALL FOR TYPEOUT

```

```

015222 017646 000000 STYPOS: MOV 2(SP), -(SP) ; PICKUP THE MODE
015226 116637 000001 015445 MOVB 1 SP, $OFILL ; LOAD ZERO FILL SWITCH

```

```

3425 015234 112637 015447      MOVB      SP + $OCN DE+.    ;; NUMBER OF DIGITS TO TYPE
3426 015240 062716 000002      ADD       #2, SP        ;; ADJUST RETURN ADDRESS
3427 015244 000406      BR        $TYPEON      ;;
3428 015246 112737 000001 015445 $TYPEON: MOVB     #1, $OFILL      ;; SET THE ZERO FILL SWITCH
3429 015254 112737 000006 015447      MOVB     #6, $OMODE+1   ;; SET FOR SIX DIGITS
3430 015262 112737 000005 015444 $TYPEON: MOVB     #5, $OCNT    ;; SET THE ITERATION COUNT
3431 015270 010346      MOV      R3, -(SP)     ;; SAVE R3
3432 015272 010446      MOV      R4, -(SP)     ;; SAVE R4
3433 015274 010546      MOV      R5, -(SP)     ;; SAVE R5
3434 015276 113704 015447      MOVB     $OMODE+1, R4   ;; GET THE NUMBER OF DIGITS TO TYPE
3435 015302 005404      NEG      R4            ;; SUBTRACT IT FOR MAX. ALLOWED
3436 015304 062704 000006      ADD      #6, R4        ;; SAVE IT FOR USE
3437 015310 110437 015446      MOVB     R4, $OMODE     ;; GET THE ZERO FILL SWITCH
3438 015314 113704 015445      MOVB     $OFILL, R4     ;; GET THE ZERO FILL SWITCH
3439 015320 016605 000012      MOV      12(SP), R5    ;; PICKUP THE INPUT NUMBER
3440 015324 005003      CLR      R3            ;; CLEAR THE OUTPUT WORD
3441 015326 006105      1$:      ROL      R5        ;; ROTATE MSB INTO "C"
3442 015330 000404      BR        $S           ;; GO DO MSB
3443 015332 006105      2$:      ROL      R5        ;; FORM THIS DIGIT
3444 015334 006105      ROL      R5
3445 015336 006105      ROL      R5
3446 015340 010503      MOV      R5, R3
3447 015342 006103      3$:      ROL      R3            ;; GET LSB OF THIS DIGIT
3448 015344 105337 015446      DECB     $OMODE        ;; TYPE THIS DIGIT?
3449 015350 100016      BPL      #S           ;; BR IF NO
3450 015352 042703 177770      BIC      #177770, R3   ;; GET RID OF JUNK
3451 015356 001002      BNE      #S           ;; TEST FOR 0
3452 015360 005704      TST     R4            ;; SUPPRESS THIS 0?
3453 015362 001403      BEQ     #S           ;; BR IF YES
3454 015364 005204      4$:      INC      R4            ;; DON'T SUPPRESS ANYMORE 0'S
3455 015366 052703 000060      BIS     #'0, R3       ;; MAKE THIS DIGIT ASCII
3456 015372 052703 000040      5$:      BIS     #' ', R3     ;; MAKE ASCII IF NOT ALREADY
3457 015376 110337 015442      MOVB     R3, #S       ;; SAVE FOR TYPING
3458 015402 104401 015442      TYPE     #S           ;; GO TYPE THIS DIGIT
3459 015406 105337 015444      7$:      DECB     $OCNT      ;; COUNT BY 1
3460 015412 003347      BGT     #S           ;; BR IF MORE TO DO
3461 015414 002402      BLT     #S           ;; BR IF DONE
3462 015416 005204      INC     R4            ;; INSURE LAST DIGIT ISN'T A BLANK
3463 015420 000744      BR        #S           ;; GO DO THE LAST DIGIT
3464 015422 012605      6$:      MOV     (SP)+, R5     ;; RESTORE R5
3465 015424 012604      MOV     (SP)+, R4     ;; RESTORE R4
3466 015426 012603      MOV     (SP)+, R3     ;; RESTORE R3
3467 015430 016666 000002 000004      MOV     2(SP), #SP    ;; SET THE STACK FOR RETURNING
3468 015436 012616      MOV     (SP)+, (SP)
3469 015440 000002      RTI
3470 015442      8$:      .BYTE   0            ;; RETURN
3471 015443      .BYTE   0            ;; STORAGE FOR ASCII DIGIT
3472 015444      .BYTE   0            ;; TERMINATOR FOR TYPE ROUTINE
3473 015445      .BYTE   0            ;; OCTAL DIGIT COUNTER
3474 015446      .WORD   0            ;; ZERO FILL SWITCH
3475      .SBTTL  BINARY TO ASCII AND TYPE ROUTINE

```

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 16 BIT

```

01  
02  
03  
04  
05  
06  
07  
08  
09  
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  
00  
01  
02  
03  
04  
05  
06  
07  
08  
09  
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  
00

```

015450 010146
015452 016601 000006
015456 000261
015460 112737 000060 015522
015466 006101
015470 001406
015472 105537 015522
015476 104401 015522
015502 000241
015504 000765
015506 012601
015510 016666 000002 000004
015516 012615
015520 000002
015522 000 000
    
```

```

: *BINARY-ASCII NUMBER AND TYPE IT
: *CALL
: *   MOV      NUMBER, SP      :: NUMBER TO BE TYPED
: *   TYPBN   :: TYPE IT
STYPBN: MOV      R1, SP      :: SAVE R1 ON THE STACK
        MOV      6, SP, R1  :: SET THE INPUT NUMBER
        SEC      :: SET "C" SO CAN KEEP TRACK OF THE NUMBER OF BITS
        MOVB    #'0, $BIN  :: SET CHARACTER TO AN ASCII "0"
        ROL     R1          :: GET THIS BIT
        BEQ     ZS         :: DONE?
        ADCB    $BIN       :: NO--SET THE CHARACTER EQUAL TO THIS BIT
        TYPE    , $BIN     :: GO TYPE THIS BIT
        CLC      :: CLEAR "C" SO CAN KEEP TRACK OF BITS
        BR      IS        :: GO DO THE NEXT BIT
ZS:     MOV      SP+, R1    :: POP THE STACK INTO R1
        MOV      2, SP, 4, SP :: ADJUST THE STACK
        MOV      SP+, SP
        RTI
$BIN:   .BYTE    0, 0
    
```

.SBTTL ERROR HANDLER ROUTINE

```

: *****
: *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
: *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
: *AND GO TO SER_TYP ON ERROR
: *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
: *SW15=1 HALT ON ERROR
: *SW13=1 INHIBIT ERROR TYPEOUTS
: *SW10=1 BELL ON ERROR
: *SW09=1 LOOP ON ERROR
: *CALL
: *   ERROR    N      :: ERROR=EMT AND N=ERROR ITEM NUMBER
    
```

```

015524 104407
015526 105237 001103
015532 001775
015534 013777 001102 163400
015542 032777 002000 16337C
015550 001402
015552 104401 001164
015556 005237 001112
015562 011637 001116
015566 162737 000002 001116
015574 117737 16331E 001114
015602 032777 020000 16333C
015610 001004
015612 004737 015732
015616 104401 001171
015622
015622 122737 000001 001214
    
```

```

$ERROR: CKSWR      :: TEST FOR CHANGE IN SOFT-SWR
        INCB    $ERFLG    :: SET THE ERROR FLAG
        BEQ     ZS        :: DON'T LET THE FLAG GO TO ZERO
        MOV     $STNM, $DISPLAY :: DISPLAY TEST NUMBER AND ERROR FLAG
        BIT     @BIT10, $SWR :: BELL ON ERROR?
        BEQ     IS        :: NO - SKIP
        TYPE    , $BELL   :: RING BELL
        INC     $ERITL    :: COUNT THE NUMBER OF ERRORS
        MOV     (SP), $ERRPC :: GET ADDRESS OF ERROR INSTRUCTION
        SUB     #2, $ERRPC
        MOVB   @ $ERRPC, $ITEMB :: STRIP AND SAVE THE ERROR ITEM CODE
        BIT     @BIT13, $SWR :: SKIP TYPEOUT IF SET
        BNE    ZOS        :: SKIP TYPEOUTS
        JSR    PC, $ERR_TYP :: GO TO USER ERROR ROUTINE
        TYPE    , $ERFLF
ZOS:    CMB      @BPTEN, $SERV :: RUNNING IN APT MODE
    
```

```

3533 015630 001007      BNE      2$          ;; NO SKIP APT ERROR REPORT
3534 015632 113737 001114 015644  MOVB    $ITEMB,21$  ;; SET ITEM NUMBER AS ERROR NUMBER
3535 015640 004737 017424      JSR     PC,$AT1$    ;; REPORT FATAL ERROR TO APT
3536 015644      000      21$:  .BITE    0
3537 015645      000      .BITE    0
3538 015646 000777      BR      22$          ;; APT ERROR LOOP
3539 015650 005777 163264  22$:  TST     $SWR      ;; HALT ON ERROR
3540 015654 100002  23$:  BPL     3$          ;; SKIP IF CONTINUE
3541 015656 000000      HALT                    ;; HALT ON ERROR!
3542 015660 104407      CKSWR                    ;; TEST FOR CHANGE IN SOFT-SWR
3543 015662 032777 001000 163250  3$:  BIT     @BIT09,$SWR  ;; LOOP ON ERROR SWITCH SET
3544 015670 001402      BEQ    4$          ;; BR IF NO
3545 015672 013716 001110      MOV    $LPERR,$SP)  ;; FUDGE RETURN FOR LOOPING
3546 015676 005737 001162  4$:  TST     $ESCAPE    ;; CHECK FOR AN ESCAPE ADDRESS
3547 015702 001402      BEQ    5$          ;; BR IF NONE
3548 015704 013716 001162      MOV    $ESCAPE,$SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
3549 015710
3550
3551 015710 005237 001436      INC    ERCNT          ;; UPDATE ERROR COUNT.
3552 015714 001002      BNE    10$          ;; BUT DON'T LET IT OVERFLOW.
3553 015716 005337 001436      DEC    ERCNT          ;; KEEP AT 177777 IF OVERFLOW.
3554 015722
3555 015722 043737 001432 001434  10$:  BIC    ROTATE,UTEST  ;; REMOVE UNIT FROM LIST OF GOOD ONES.
3556 015730 000002      RTI                    ;; EXIT.

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

3558
3559
3560
3561 *****
3562 *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
3563 *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" $ERRTB .
3564 *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
3565
3566 SERrTYP:
3567 TYPE      $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3568 MOV      RO,-($SP)      ;; SAVE RO
3569 CLR      RO              ;; PICKUP THE ITEM INDEX
3570 BISB    @($ITEMB,RO)
3571 BNE     1$
3572 MOV     $ERRPC,$SP      ;; IF ITEM NUMBER IS ZERO, JUST
3573                          ;; TYPE THE PC OF THE ERROR
3574 TYPOC                    ;; SAVE $ERRPC FOR TYPEOUT
3575 BR      6$              ;; ERROR ADDRESS
3576 DEC     RO              ;; GO TYPE--OCTAL ASCII:ALL DIGITS
3577 ASL    RO              ;; GET OUT
3578 ASL    RO              ;; ADJUST THE INDEX SO THAT IT WILL
3579 ASL    RO              ;; WORK FOR THE ERRCR TABLE
3580 ADD     @($ERRTB,RO)    ;; FORM TABLE POINTER
3581 MOV     (RO)+,2$        ;; PICKUP "ERROR MESSAGE" POINTER
3582 BEQ    3$              ;; SKIP TYPEOUT IF NO POINTER
3583 TYPE                    ;; TYPE THE "ERROR MESSAGE"
3584 .WORD   0              ;; "ERROR MESSAGE" POINTER GOES HERE
3585 TYPE    $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3586 MOV     (RO)+,4$        ;; PICKUP "DATA HEADER" POINTER

```

```

3587 016016 001404      BEQ      SS      ;; SKIP TYPEOUT IF 0
3588 016020 104401      TYPE     ;; TYPE THE "DATA HEADER"
3589 016022 000000      .WORD   0      ;; "DATA HEADER" POINTER GOES HERE
3590 016024 104401 001171 4S:      .TYPE  'SCRLF  ;; "CARRIAGE RETURN" & "LINE FEED"
3591 016030 011000      MOV     'RC,RC  ;; PICKUP "DATA TABLE" POINTER
3592 016032 001004      BNE    ~S      ;; GO TYPE THE DATA
3593 016034 012600      6S:    MOV     SP+RC  ;; RESTORE RC
3594 016036 104401 001171  .TYPE  'SCRLF  ;; "CARRIAGE RETURN" & "LINE FEED"
3595 016042 000207      RTS     AC      ;; RETURN
3596 016044
3597 016044 013046      MOV     2(RC)+,-SP  ;; SAVE 2(RC)+ FOR TYPEOUT
3598 016046 104402      TYPEOC  ;; GO TYPE--OCTAL ASCII/ALL DIGITS
3599 016050 005710      TST    RC      ;; IS THERE ANOTHER NUMBER?
3600 016052 001770      BEQ    6S      ;; BR IF NO
3601 016054 104401 016062  .TYPE  'S      ;; TYPE TWO(2) SPACES
3602 016060 000771      BR     7S      ;; LOOP
3603 016062 020040 000      8S:    .ASCIZ  ;; TWO(2) SPACES
3604 016066 016066      .EVEN
3605
3606      .SBTTL SCOPE HANDLER ROUTINE
3607
3608      ;*****
3609      ;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3610      ;AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY%7.C
3611      ;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY%15:08
3612      ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3613      ;*SW14=1 LOOP ON TEST
3614      ;*SW11=1 INHIBIT ITERATIONS
3615      ;*SW09=1 LOOP ON ERROR
3616      ;*SW08=1 LOOP ON TEST IN SWR%7:0
3617      ;*CALL
3618      ;* SCOPE ;;SCOPE=ICT
3619
3620      $SCOPE:
3621      CKSWR      ;; TEST FOR CHANGE IN SOFT-SWR
3622      CKSWR
3623      BIT      #BIT14,2SWR  ;; LOOP ON PRESENT TEST?
3624      BNE      $OVER      ;; YES IF SW14=1
3625      ;*****START OF CODE FOR THE XOR TESTER*****
3626      $XTSTR: BR     6S      ;; IF RUNNING ON THE "XOR" TESTER CHANGE
3627      MOV     2$ERRVEC,-(SP)  ;; THIS INSTRUCTION TO A "NOP" (NOP=24C
3628      MOV     #55,2$ERRVEC  ;; SAVE THE CONTENTS OF THE ERROR VECTOR
3629      TST    2$177060      ;; SET FOR TIMEOUT
3630      MOV     (SP)+,2$ERRVEC  ;; TIME OUT ON XOR?
3631      BR     $$VLAD      ;; RESTORE THE ERROR VECTOR
3632      5S:    CMP     (SP)+,(SP)+  ;; GO TO THE NEXT TEST
3633      MOV     (SP)+,2$ERRVEC  ;; CLEAR THE STACK AFTER A TIME OUT
3634      BR     7S      ;; RESTORE THE ERROR VECTOR
3635      6S:    ;*****END OF CODE FOR THE XOR TESTER*****
3636      BIT      #BIT08,2SWR  ;; LOOP ON SPEC. TEST?
3637      BEQ    2S      ;; BR IF NO
3638      CMPB   2$SWR,$STNM  ;; ON THE RIGHT TEST? SWR%7:0
3639      BEQ    $OVER      ;; BR IF YES
3640      TSTB   $ERFLG      ;; HAS AN ERROR OCCURRED?

```

```

3641 016164 001421      BEQ      3$          ;; BR IF NO
3642 016166 123737 001115 001103  CMPB    $ERMAX,$ERFLG ;; MAX. ERRORS FOR THIS TEST OCCURRED
3643 016174 101015      BPT     3$          ;; BR IF NO
3644 016176 032777 001000 162734  BIT     #BIT09,$SWR   ;; LOOP ON ERROR?
3645 016204 001404      BEQ     4$          ;; BR IF NO
3646 016206 013737 001110 001106 7$: MOV    $LPERR,$LPADR ;; SET LOOP ADDRESS TO LAST SCOPE
3647 016214 000446      BR     $OVER
3648 016216 105037 001103      CLRB   $ERFLG       ;; ZERO THE ERROR FLAG
3649 016222 005037 001160      CLR    $TIMES       ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
3650 016226 000415      BR     1$          ;; ESCAPE TO THE NEXT TEST
3651 016230 032777 004000 162702 3$: BIT     #BIT11,$SWR   ;; INHIBIT ITERATIONS?
3652 016236 001011      BNE    1$          ;; BR IF YES
3653 016240 005737 001202      TST    $PASS        ;; IF FIRST PASS OF PROGRAM
3654 016244 001406      BEQ    1$          ;; INHIBIT ITERATIONS
3655 016246 005237 001104      INC    $ICNT        ;; INCREMENT ITERATION COUNT
3656 016252 023737 001160 001104  CMP    $TIMES,$ICNT  ;; CHECK THE NUMBER OF ITERATIONS MADE
3657 016260 002024      BGE    $OVER        ;; BR IF MORE ITERATION REQUIRED
3658 016262 012737 000001 001104 1$: MOV    #1,$ICNT    ;; REINITIALIZE THE ITERATION COUNTER
3659 016270 013737 016346 001160  MOV    $MXCNT,$TIMES ;; SET NUMBER OF ITERATIONS TO DO
3660 016276 105237 001102  $SVLAD: INCB   $STNM    ;; COUNT TEST NUMBERS
3661 016302 113737 001102 001200  MOVB   $STNM,$TESTN  ;; SET TEST NUMBER IN APT MAILBOX
3662 016310 011637 001106      MOV    (SP),$LPADR  ;; SAVE SCOPE LOOP ADDRESS
3663 016314 011637 001110      MOV    (SP),$LPERR  ;; SAVE ERROR LOOP ADDRESS
3664 016320 005037 001162      CLR    $ESCAPE     ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
3665 016324 112737 000001 001115  MOVB   #1,$ERMAX    ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3666 016332 013777 001102 162602 $OVER: MOV    $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
3667 016340 013716 001106      MOV    $LPADR,(SP) ;; FUDGE RETURN ADDRESS
3668 016344 000002      RTI
3669 016346 003720      $MXCNT: 2000.
3670      .SBTTL TTY INPUT ROUTINE
3671
3672      ;*****
3673      .ENABL  LSB
3674
3675      ;*****
3676      ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3677      ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3678      ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3679      ;*WHEN OPERATING IN TTY FLAG MODE.
3680 016350 022737 000176 001140 $CKSWR: CMP    #SWREG,$SWR   ;; IS THE SOFT-SWR SELECTED?
3681 016356 001074      BNE    15$         ;; BRANCH IF NO
3682 016360 105777 162560      TSTB   $STKS       ;; CHAR THERE?
3683 016364 100071      BPL    15$         ;; IF NO, DON'T WAIT AROUND
3684 016366 117746 162554      MOVB   $STKB,-(SP) ;; SAVE THE CHAR
3685 016372 042716 177600      BIC    #C177,(SP)  ;; STRIP-OFF THE ASCII
3686 016376 022726 000007      CMP    #7,(SP)+    ;; IS IT A CONTROL-G?
3687 016402 001062      BNE    15$         ;; NO, RETURN TO USER
3688 016404 123727 001134 000001  CMPB   $AUTOB,#1   ;; ARE WE RUNNING IN AUTO-MODE?
3689 016412 001456      BEQ    15$         ;; BRANCH IF YES
3690
3691 016414 104401 017075      $GTSWP: TYPE   $CNTLG  ;; ECHO THE CONTROL-G (#G)
3692 016420 104401 017102      TYPE   $MSWR       ;; TYPE CURRENT CONTENTS
3693 016424 013746 000176      MOV    $SWREG,-(SP) ;; SAVE SWREG FOR TYPEOUT
3694 016430 104402      TYPOC  ;; GO TYPE--OCTAL ASCII(ALL DIGITS)

```

```

3695 016432 104401 017113          TYPE      .SMNEW      :: PROMPT FOR NEW SWR
3696 016436 005046                   19$: CLR      - SP      :: CLEAR COUNTER
3697 016440 005046                   CLR      -(SP      :: THE NEW SWR
3698 016442 105777 162476          7$: TSTB   2STKS    :: CHAR THERE?
3699 016446 100375                   BPL      7$      :: IF NOT TRY AGAIN
3700
3701 016450 117746 162472          MOVB    2STKB, SP  :: PICK UP CHAR
3702 016454 042716 177600          BIC     #10177, SP :: MAKE IT 7-BIT ASCII
3703
3704
3705
3706 016460 021627 000025          9$: CMP     (SP), #25  :: IS IT A CONTROL-U?
3707 016464 001005                   BNE     10$      :: BRANCH IF NOT
3708 016466 104401 017070          TYPE    $CNTLL    :: YES ECHO CONTROL-U (1)
3709 016472 062706 000006          20$: ADD    #6, SP  :: IGNORE PREVIOUS INPUT
3710 016476 000757                   BR      19$      :: LET'S TRY IT AGAIN
3711
3712
3713 016500 021627 000015          10$: CMP     (SP), #15  :: IS IT A <CR>?
3714 016504 001022                   BNE     16$      :: BRANCH IF NO
3715 016506 005766 000004          TST     4(SP     :: YES, IS IT THE FIRST CHAR?
3716 016512 001403                   BEQ     11$      :: BRANCH IF YES
3717 016514 016677 000002 162416  MOV     2(SP), 2SWR :: SAVE NEW SWR
3718 016522 062706 000006          11$: ADD    #6, SP  :: CLEAR UP STACK
3719 016526 104401 001171          14$: TYPE    $CRLF   :: ECHO <CR> AND <LF>
3720 016532 123727 001135 000001  CMPB   $INTAG, #1  :: RE-ENABLE TTY KBD INTERRUPTS
3721 016540 001003                   BNE     15$      :: BRANCH IF NOT
3722 016542 012777 000100 162374  MOV     #100, 2STKS :: RE-ENABLE TTY KBD INTERRUPTS
3723 016550 000002                   RTI     :: RETURN
3724 016552 004737 017336          16$: JSR     PC, STYPEC :: ECHO CHAR
3725 016556 021627 000060          CMP     (SP), #60  :: CHAR < 0?
3726 016562 002420                   BLT     18$      :: BRANCH IF YES
3727 016564 021627 000067          CMP     (SP), #67  :: CHAR > 7?
3728 016570 003015                   BGT     18$      :: BRANCH IF YES
3729 016572 042726 000060          BIC     #60, (SP)+ :: STRIP-OFF ASCII
3730 016576 005766 000002          TST     2(SP)     :: IS THIS THE FIRST CHAR
3731 016602 001403                   BEQ     17$      :: BRANCH IF YES
3732 016604 006316                   ASL     (SP)      :: NO, SHIFT PRESENT
3733 016606 006316                   ASL     (SP)      :: CHAR OVER TO MAKE
3734 016610 006316                   ASL     (SP)      :: ROOM FOR NEW ONE.
3735 016612 005266 000002          17$: INC     2(SP)   :: KEEP COUNT OF CHAR
3736 016616 056616 177776          BIS     -2(SP), (SP) :: SET IN NEW CHAR
3737 016622 000707                   BR      7$      :: GET THE NEXT ONE
3738 016624 104401 001170          18$: TYPE    $QUES   :: TYPE ?<CR><LF>
3739 016630 000720                   BR      20$     :: SIMULATE CONTROL-U
3740
3741
3742
3743
3744
3745
3746
3747
3748

```

```

*****
* THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
* CALL:
*      RDCHR      :: INPUT A SINGLE CHARACTER FROM THE TTY
*      RETURN HERE :: CHARACTER IS ON THE STACK
*                :: WITH PARITY BIT STRIPPED OFF

```

```

3749 :
3750 :
3751 016632 011646 $RDCHR: MOV (SP),-(SP) ;; PUSH DOWN THE PC
3752 016634 016666 000004 000002 MOV 4(SP),2(SP) ;; SAVE THE PS
3753 016642 105777 162276 1S: TSTB $TKS ;; WAIT FOR
3754 016646 100375 BPL 1S ;; A CHARACTER
3755 016650 117766 162272 000004 MOVB $TKB,4(SP) ;; READ THE TTY
3756 016656 042766 177600 000004 BIC #C(177),4(SP) ;; GET RID OF JUNK IF ANY
3757 016664 026627 000004 000023 CMP 4(SP),#23 ;; IS IT A CONTROL-'S'
3758 016672 001013 BNE 3S ;; BRANCH IF NO
3759 016674 105777 162244 2S: TSTB $TKS ;; WAIT FOR A CHARACTER
3760 016700 100375 BPL 2S ;; LOOP UNTIL ITS THERE
3761 016702 117746 162240 MOVB $TKB,-(SP) ;; GET CHARACTER
3762 016706 04271F 177600 BIC #C(177),(SP) ;; MAKE IT 7-BIT ASCII
3763 016712 022627 000021 CMP (SP)+,#21 ;; IS IT A CONTROL-'Q'
3764 016716 001366 BNE 2S ;; IF NOT DISCARD IT
3765 016720 000750 BR 1S ;; YES, RESUME
3766 016722 026627 000004 000140 3S: CMP 4(SP),#140 ;; IS IT UPPER CASE?
3767 016730 002407 BLT 4S ;; BRANCH IF YES
3768 016732 026627 000004 000175 CMP 4(SP),#175 ;; IS IT A SPECIAL CHAR?
3769 016740 003003 BGT 4S ;; BRANCH IF YES
3770 016742 042766 000040 000004 BIC #40,4(SP) ;; MAKE IT UPPER CASE
3771 016750 000002 4S: RTI ;; GO BACK TO USER
3772 :*****
3773 :*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3774 :*CALL:
3775 :* RDLIN ;; INPUT A STRING FROM THE TTY
3776 :* RETURN HERE ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3777 :* ;; TERMINATOR WILL BE A BYTE OF ALL C'S
3778 :
3779 016752 010346 $RDLIN: MOV R3, -(SP) ;; SAVE R3
3780 016754 012703 017060 1S: MOV $TTYIN,R3 ;; GET ADDRESS
3781 016760 022703 017070 2S: CMP $TTYIN+8.,R3 ;; BUFFER FULL?
3782 016764 101405 BLOS 4S ;; BR IF YES
3783 016766 104410 RDCHR ;; GO READ ONE CHARACTER FROM THE TTY
3784 016770 112613 MOVB (SP)+,(R3) ;; GET CHARACTER
3785 016772 122713 000177 10S: CMPB #177,(R3) ;; IS IT A RUBOUT
3786 016776 001003 BNE 3S ;; SKIP IF NOT
3787 017000 104401 001170 4S: TYPE $QUES ;; TYPE A '?'
3788 017004 000763 BR 1S ;; CLEAR THE BUFFER AND LOOP
3789 017006 111337 017056 3S: MOVB (R3),9S ;; ECHO THE CHARACTER
3790 017012 104401 017056 TYPE 9S
3791 017016 122723 000015 CMPB #15,(R3)+ ;; CHECK FOR RETURN
3792 017022 001356 BNE 2S ;; LOOP IF NOT RETURN
3793 017024 105063 177777 CLRB -1(R3) ;; CLEAR RETURN (THE 15)
3794 017030 104401 001172 TYPE $LF ;; TYPE A LINE FEED
3795 017034 012603 MOV (SP)+,R3 ;; RESTORE R3
3796 017036 011646 MOV (SP),-(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3797 017040 016666 000004 000002 MOV 4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
3798 017046 012766 017060 000004 MOV $TTYIN,4(SP)
3799 017054 000002 RTI ;; RETURN
3800 017056 000 .BYTE 0 ;; STORAGE FOR ASCII CHAR. TO TYPE
3801 017057 000 .BYTE 0 ;; TERMINATOR
3802 017060 000010 $TTYIN: .BLKB 8. ;; RESERVE 8 BYTES FOR TTY INPUT

```



```

3803 017070 052536 005015 000 $CNTLU: .ASCIZ /U<<15><12  ::CONTROL "U"
3804 017075 136 006507 000012 $CNTLG: .ASCIZ /G<<15><12  ::CONTROL "G"
3805 017102 005015 053523 020.22 $MSWR: .ASCIZ <15><12 >SWR =
3806 017110 020075 000
3807 017113 040 047040 053505 $MNEW: .ASCIZ / NEW = /
3808 017120 036440 000040
3809
3810 .SBTTL TYPE ROUTINE
3811
3812 ::*****
3813 ::*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3814 ::*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3815 ::*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3816 ::*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3817 ::*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3818
3819 ::*CALL:
3820 ::*1) USING A TRAP INSTRUCTION
3821 ::* TYPE ,MESADR ::MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3822
3823 ::*OR
3824 ::* TYPE
3825 ::* MESADR
3826
3826 017124 105737 001157 $TYPE: TSTB $TFPLG :: IS THERE A TERMINAL?
3827 017130 100002 BPL 1$ :: BR IF YES
3828 017132 000000 HALT :: HALT HERE IF NO TERMINAL
3829 017134 000430 BR 3$ :: LEAVE
3830 017136 010046 1$: MOV RO,-(SP) :: SAVE RO
3831 017140 017600 000002 MOV #2(SP),RO :: GET ADDRESS OF ASCIZ STRING
3832 017144 122737 000001 001214 CMPB #APTENV,$ENV :: RUNNING IN APT MODE
3833 017152 001011 BNE 62$ :: NO, GO CHECK FOR APT CONSOLE
3834 017154 132737 000100 001215 BITB #APTSPool,$ENVM :: SPOOL MESSAGE TO APT
3835 017162 001405 BEQ 62$ :: NO, GO CHECK FOR CONSOLE
3836 017164 010037 017174 MOV RO,61$ :: SETUP MESSAGE ADDRESS FOR APT
3837 017170 004737 017414 JSR PC,$ATY3 :: SPOOL MESSAGE TO APT
3838 017174 000000 .WORD 0 :: MESSAGE ADDRESS
3839 017176 132737 000040 001215 61$: BITB #APTCsup,$ENVM :: APT CONSOLE SUPPRESSED
3840 017204 001003 BNE 60$ :: YES, SKIP TYPE OUT
3841 017206 112046 2$: MOV (RO)+,-(SP) :: PUSH CHARACTER TO BE TYPED ONTO STACK
3842 017210 001005 BNE 4$ :: BR IF IT ISN'T THE TERMINATOR
3843 017212 005726 TST (SP)+ :: IF TERMINATOR POP IT OFF THE STACK
3844 017214 012600 60$: MOV (SP)+,RO :: RESTORE RO
3845 017216 062716 000002 3$: ADD #2,(SP) :: ADJUST RETURN PC
3846 017222 000002 RTI :: RETURN
3847 017224 122716 000011 4$: CMPB #HT,(SP) :: BRANCH IF <HT>
3848 017230 001430 BEQ 8$
3849 017232 122716 000200 CMPB #CRLF,(SP) :: BRANCH IF NOT <CRLF>
3850 017236 001006 BNE 5$
3851 017240 005726 TST (SP)+ :: POP <CR><LF> EQUIV
3852 017242 104401 TYPE :: TYPE A CR AND LF
3853 017244 001171 $CRLF
3854 017246 105037 017402 CLRB $CHARCNT :: CLEAR CHARACTER COUNT
3855 017252 000755 BR 2$ :: GET NEXT CHARACTER
3856 017254 004737 017336 5$: JSP PC,$TYPEC :: GO TYPE THIS CHARACTER

```

```

3857 017260 123726 001156 6$: CMPB $FILLC, SP + :: IS IT TIME FOR FILLER CHARS.
3858 017264 001350 2$: BNE 2$ :: IF NO GO GET NEXT CHAR.
3859 017266 013746 001154 MOV $NULL, SP :: GET # OF FILLER CHARS. NEEDED
3860 :: AND THE NULL CHAR.
3861 017272 105366 000001 7$: DECB 1(SP) :: DOES A NULL NEED TO BE TYPED?
3862 017276 002770 6$: BLT 6$ :: BR IF NO--GO POP THE NULL OFF OF STACK
3863 017300 004737 017336 JSR PC,$TYPEC :: GO TYPE A NULL
3864 017304 105337 017402 DECB $CHARCNT :: DO NOT COUNT AS A COUNT
3865 017310 000770 BR 7$ :: LOOP
3866
3867
3868 ;HORIZONTAL TAB PROCESSOR
3869 017312 112716 000040 8$: MOVB #' (SP) :: REPLACE TAB WITH SPACE
3870 017316 004737 017336 9$: JSR PC,$TYPEC :: TYPE A SPACE
3871 017322 132737 000000 017402 BITB #',$CHARCNT :: BRANCH IF NOT AT
3872 017330 001372 9$: BNE 9$ :: TAB STOP
3873 017332 005726 TST (SP)+ :: POP SPACE OFF STACK
3874 017334 000724 2$: BR 2$ :: GET NEXT CHARACTER
3875 017336 105777 161606 $TYPEC: TST $STPS :: WAIT UNTIL PRINTER IS READY
3876 017342 100375 BPL $TYPEC
3877 017344 116677 000002 161600 MOVB 2(SP),2$TPB :: LOAD CHAR TO BE TYPED INTO DATA REG.
3878 017352 122766 000015 000002 CMPB $CR,2(SP) :: IS CHARACTER A CARRIAGE RETURN?
3879 017360 001003 1$ BNE 1$ :: BRANCH IF NO
3880 017362 105037 017402 CLRB $CHARCNT :: YES--CLEAR CHARACTER COUNT
3881 017366 000406 BR $TYPEX :: EXIT
3882 017370 122766 000012 000002 1$: CMPB $LF,2(SP) :: IS CHARACTER A LINE FEED?
3883 017376 001402 BEQ $TYPEX :: BRANCH IF YES
3884 017400 105227 (PC)+ COUNT THE CHARACTER
3885 017402 000000 $CHARCNT: .WORD 0 :: CHARACTER COUNT STORAGE
3886 017404 000207 $TYPEX: RTS PC
3887
3888 .SBTTL APT COMMUNICATIONS ROUTINE
3889
3890 *****
3891 017406 112737 000001 017652 $ATY1: MOVB #1,$FFLG :: TO REPORT FATAL ERROR
3892 017414 112737 000001 017650 $ATY3: MOVB #1,$MFLG :: TO TYPE A MESSAGE
3893 017422 000403 BR $ATYC
3894 017424 112737 000001 017652 $ATY4: MOVB #1,$FFLG :: TO ONLY REPORT FATAL ERROR
3895 017432 $ATYC:
3896 017432 010046 MOV R0,-(SP) :: PUSH R0 ON STACK
3897 017434 010146 MOV R1,-(SP) :: PUSH R1 ON STACK
3898 017436 105737 017650 TSTB $MFLG :: SHOULD TYPE A MESSAGE?
3899 017442 001450 5$: BEQ 5$ :: IF NOT: BR
3900 017444 122737 000001 001214 CMPB $APTENV,$ENV :: OPERATING UNDER APT?
3901 017452 001031 3$: BNE 3$ :: IF NOT: BR
3902 017454 132737 000100 001215 BITB $APTSPOOL,$ENVM :: SHOULD SPOOL MESSAGES?
3903 017462 001425 3$: BEQ 3$ :: IF NOT: BR
3904 017464 017600 000004 MOV 24(SP),R0 :: GET MESSAGE ADDR.
3905 017470 062766 000002 000004 ADD #2,4(SP) :: BUMP RETURN ADDR.
3906 017476 005737 001174 1$: TST $MSGTYPE :: SEE IF DONE W/ LAST XMISSION?
3907 017502 001375 1$: BNE 1$ :: IF NOT: WAIT
3908 017504 010037 001210 MOV R0,$MSGADR :: PUT ADDR IN MAILBOX
3909 017510 105720 2$: TSTB (R0)+ :: FIND END OF MESSAGE
3910 017512 001376 BNE 2$

```

```

3911 017514 163700 001210 SUB $MSGAD,RO ::SUB START OF MESSAGE
3912 017520 006200 ASR RO ::GET MESSAGE LNGLH IN WORDS
3913 017522 010037 001212 MOV RO,$MSGLEN ::PUT LENGTH IN MAILBOX
3914 017526 012737 000004 001174 MOV #4,$MSGTYPE ::TELL APT TO TAKE MSG.
3915 017534 000413 BR $S ::
3916 017536 017637 000004 017562 3S: MOV #4(SP),4S ::PUT MSG ADDR IN JSR LINKAGE
3917 017544 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDRESS
3918 017552 013746 177776 MOV 177776,-5P ::PUSH 177776 ON STACK
3919 017556 004737 017124 JSR PC,$TYPE ::CALL TYPE MACRO
3920 017562 000000 4S: .WORD 0
3921 017564 5S:
3922 017564 105737 017652 10S: TSTB $FFLG ::SHOULD REPORT FATAL ERROR?
3923 017570 001416 BEQ 12S ::IF NOT: BR
3924 017572 005737 001214 TST $ENV ::RUNNING UNDER APT?
3925 017576 001413 BEQ 12S ::IF NOT: BR
3926 017600 005737 001174 11S: TST $MSGTYPE ::FINISHED LAST MESSAGE?
3927 017604 001375 BNE 11S ::IF NOT: WAIT
3928 017606 017637 000004 001176 MOV #4(SP),$FATAL ::GET ERROR #
3929 017614 062766 000002 000004 ADD #2,4(SP) ::BUMP RETURN ADDR.
3930 017622 005237 001174 INC $MSGTYPE ::TELL APT TO TAKE ERROR
3931 017626 105037 017652 12S: CLRB $FFLG ::CLEAR FATAL FLAG
3932 017632 105037 017651 CLRB $LFLG ::CLEAR LOG FLAG
3933 017636 105037 017650 CLRB $MFLG ::CLEAR MESSAGE FLAG
3934 017642 012601 MOV (SP)+,R1 ::POP STACK INTO R1
3935 017644 012600 MOV (SP)+,R0 ::POP STACK INTO R0
3936 017646 000207 RTS PC ::RETURN
3937 017650 000 $MFLG: .BYTE 0 ::MESSG. FLAG
3938 017651 000 $LFLG: .BYTE 0 ::LOG FLAG
3939 017652 000 $FFLG: .BYTE 0 ::FATAL FLAG
3940 017654 .EVEN
3941 000200 APTSIZE=200
3942 000001 APTENV=001
3943 000100 APTSPool=100
3944 000040 APTCSUP=040
3945 .SBTTL POWER DOWN AND UP ROUTINES
3946
3947 *****
3948 :POWER DOWN ROUTINE
3949 017654 012737 020014 000024 $PWRDN: MOV #SILLUP,$PWRVEC ::SET FOR FAST UP
3950 017662 012737 000340 000026 MOV #340,$PWRVEC+2 ::PRIO:7
3951 017670 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
3952 017672 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
3953 017674 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
3954 017676 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
3955 017700 010446 MOV R4,-(SP) ::PUSH R4 ON STACK
3956 017702 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
3957 017704 017746 161230 MOV #SWR,-(SP) ::PUSH #SWR ON STACK
3958 017710 010637 020020 MOV SP,$SAVR6 ::SAVE SP
3959 017714 012737 017726 000024 MOV #SPWRUP,$PWRVEC ::SET UP VECTOR
3960 017722 000000 HALT
3961 017724 000776 BR .-2 ::HANG UP
3962
3963 *****
3964 :POWER UP ROUTINE

```

```

3965 017726 012737 020014 000024 $PWRUP: MOV $SILLUP, @PWRVEC ; SET FOR FAST DOWN
3966 017734 013706 020020 MOV $SAVR6, SP ; GET SP
3967 017740 005037 020020 CLR $SAVR6 ; WAIT LOOP FOR THE **
3968 017744 005237 020020 1$: INC $SAVR6 ; WAIT FOR THE INC
3969 017750 001375 BNE 1$ ; OF WORD
3970 017752 012677 161162 MOV (SP)+, @JSWP ; POP STACK INTO @JSWP
3971 017756 012605 MOV (SP)+, R5 ; POP STACK INTO R5
3972 017760 012604 MOV (SP)+, R4 ; POP STACK INTO R4
3973 017762 012603 MOV (SP)+, R3 ; POP STACK INTO R3
3974 017764 012602 MOV (SP)+, R2 ; POP STACK INTO R2
3975 017766 012601 MOV (SP)+, R1 ; POP STACK INTO R1
3976 017770 012600 MOV (SP)+, R0 ; POP STACK INTO R0
3977 017772 012737 017654 000024 MOV $PWRDN, @PWRVEC ; SET UP THE POWER DOWN .E.L.T =
3978 020000 012737 000340 000026 MOV #340, @PWRVEC+2 ; PRI0:7
3979 020006 104401 TYPE ; REPORT THE POWER FAILURE
3980 020010 020022 $PWRMG: .WORD $POWER ; POWER FAIL MESSAGE POINT ER
3981 020012 000002 RTI
3982 020014 000000 $SILLUP: HALT ; THE POWER UP SEQUENCE WAS STARTED
3983 020016 000776 BR -2 ; BEFORE THE POWER DOWN WAS COMPLETE
3984 020020 000000 $SAVR6: 0 ; PUT THE SP HERE
3985 020022 005015 047520 042527 $POWER: .ASCIZ 15 12 "POWER"
3986 020030 000122 .EVEN
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008 020032 011637 020206 020206 IOTRD: MOV (6), TRTO ; GET WHERE WE CAME TO.
4009 020036 162737 000004 020206 SUB #4, TRTO ; FORM READ ADDR.
4010
4011 020044 023727 020206 001500 CMP TRTO, #1000 ; DID TRAP FROM LESS THAN ADDR. 1000?
4012 020052 003402 BLE Z$ ; NO-CONTINUE.
4013
4014 020054 000000 1$: HALT ; A BUSS ERROR TIME OUT TRAP BROUGHT US HERE.
4015 ; ADDRESS CONTAINED IN TRTO.
4016
4017 020056 000776 BR 1$ ; DON'T ALLOW CONTINUE.
4018
    
```



4073 020210 000000 TRAP: WORD 0 :CONTAINS ADDR. WE TRAPPED OF INTR. FRM.  
4074 .SBTTL TRAP DECODER

4075  
4076  
4077  
4078  
4079  
4080  
4081  
4082  
4083  
4084  
4085  
4086  
4087  
4088  
4089  
4090  
4091  
4092  
4093  
4094  
4095  
4096  
4097  
4098  
4099  
4100  
4101  
4102  
4103  
4104  
4105  
4106  
4107  
4108  
4109  
4110  
4111  
4112  
4113  
4114  
4115  
4116  
4117  
4118  
4119  
4120  
4121  
4122  
4123  
4124  
4125  
4126  
4127

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

020212 010046 \$TRAP: MOV RC - SP) ;;SAVE RD  
020214 01660C 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS  
020220 005740 TST -(RO) ;;BACKUP BY 2  
020222 111000 MOVB (RO),RO ;;GET RIGHT BYTE OF TRAP  
020224 006300 ASL RO ;;POSITION FOR INDEXING  
020226 016000 020246 MOV \$TRAPD(RO,RO) ;;INDEX TO TABLE  
020232 000200 RTS RO ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

020234 011646 \$TRAP2: MOV SP) -(SP) ;;MOVE THE PC DOWN  
020236 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN  
020244 000002 RTI ;;RESTORE THE PSW

.SBTTL TRAP TABLE

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

ROUTINE

\$TRAPD: .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  
\$TYPBN ;;CALL=TYPBN TRAP+5(104405) TYPE BINARY (ASCII) NUMBER  
\$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  
EM1: .ASCIZ <15><12><CLOCK SR FUNCTION ERROR>  
EM2: .ASCIZ 15<12> CLOCK SR DATA ERROR.  
EM3: .ASCIZ <15 12> CLOCK BR DATA ERROR.

Line	Address	Value	Address	Value	Label	Text
1127	020366	051101	020101	051101		
1128	020374	047522	000122			
1129	020400	044500	052116	051105	EM4:	.ASCIZ <200> INTERRUPT ERROR
1130	020406	052522	052120	042440		
1131	020414	051122	051117	000		
1132	020421	015	041412	052517	EM5:	.ASCIZ <15><12> COUNT REG. ERROR
1133	020426	052116	051040	043505		
1134	020434	020056	051105	047522		
1135	020442	000122				
1136	020444	005015	047503	047125	EM11:	.ASCIZ <15><12> COUNT ERROR
1137	020452	020124	051105	047522		
1138	020460	020122	000			
1139	020463	015	041412	052517	EM12:	.ASCIZ <15><12> COUNT FUNCTION ERROR
1140	020470	052116	043040	047125		
1141	020476	052103	047511	020116		
1142	020504	051105	047522	000122		
1143	020512	005015	046103	041517	EM16:	.ASCIZ <15><12> CLOCK INTERRUPT ERROR
1144	020520	020113	047111	042524		
1145	020526	051122	050125	020124		
1146	020534	051105	047522	020122		
1147	020542	000				
1148	020543	015	051012	050105	EM20:	.ASCIZ <15><12> REPEATABILITY ERROR
1149	020550	040505	040524	044502		
1150	020556	044514	054524	042440		
1151	020564	051122	051117	000040		
1152	020572	005015	042101	051104	EM26:	.ASCIZ <15><12> ADDRESSING ERROR
1153	020600	051505	044523	043516		
1154	020606	042440	051122	051117		
1155	020614	000				
1156						
1157	020615	015	042412	051122	DH1:	.ASCIZ <15><12> ERRPC ASR WAS S B
1158	020622	041520	040411	051123		
1159	020630	053411	051501	051411		
1160	020636	041057	000			
1161	020641	015	042412	051122	DH3:	.ASCIZ <15><12> ERRPC ABR WAS S B
1162	020646	041520	040411	051102		
1163	020654	053411	051501	051411		
1164	020662	041057	000			
1165	020665	200	051105	050122	DH4A:	.ASCIZ <200> ERRPC TO FROM ADDR.
1166	020672	020103	020040	047524		
1167	020700	020040	020040	020040		
1168	020706	051106	046517	040440		
1169	020714	042104	027122	000		
1170	020721	015	042412	051122	DH12:	.ASCIZ <15><12> ERRPC ASR
1171	020726	041520	040411	051123		
1172	020734	000011				
1173	020736	005015	051105	050122	DH20:	.ASCIZ <15><12> ERRPC ASR 2ND CNT 1ST NCT
1174	020744	004503	051501	004522		
1175	020752	047062	041504	052116		
1176	020760	030411	052123	041516		
1177	020766	004524	000			
1178	020771	015	042412	051122	DH26:	.ASCIZ <15><12> ERRPC CLOCK ADDR.
1179	020776	041520	041411	047514		
1180	020804	045503	040440	042104		

```

021012 027122 000000
021015 042115 030460
021020 046526 041516
021030 042055
021032 050200 042514 051501
021040 020105 052520 046114
021046 047440 052125 051440
021054 020124 053523 052111
021062 044103 051505 040440
021070 042116 052040 042510
021076 020116 052524 047122
021104 052200 042510 020115
021112 047503 050115 042514
021120 042524 054514 041440
021126 020127 051117 041440
021134 053503 003600 000000

021142 .EVEN
021142 001116 001376 001126 DT1: .WORD $ERRPC,ASR,$BDDAT,$GDDAT,0
021150 001124 000000
021154 001116 001400 001126 DT3: .WORD $ERRPC,ABR,$BDDAT,$GDDAT,0
021162 001124 000000
021166 001116 020206 020210 DT4: .WORD $ERRPC,TRTO,TRFR0,0
021174 000000
021176 001116 001376 000000 DT12: .WORD $ERRPC,ASR,0
021204 001116 001376 001126 DT20: .WORD $ERRPC,ASR,$BDDAT,$GDDAT,0
021212 001124 000000
021216 001116 001376 001126 DT22: .WORD $ERRPC,ASR,$BDDAT,$TMPD,0
021224 001424 000000
021230 001116 001424 000000 DT26: .WORD $ERRPC,$TMPD,0
021236 000000 000000 DFC: .WORD 0,0
000001 .END

```



A3ASE = 170420  
 ACPUOP = 000000  
 ADDW11 = 000000  
 ADDW15 = 000000  
 ADDW5 = 000000  
 ADDW9 = 000000  
 AENV = 000000  
 AMADR3 = 000000  
 AMAMS3 = 000000  
 AMSGTY = 000000  
 AMTYP4 = 000000  
 APRIOR = 000200  
 APTSPO = 000100  
 ATESTN = 000000  
 AVECT2 = 000000  
 BIT02 = 000004  
 BIT06 = 000100  
 BIT1 = 000002  
 BIT13 = 020000  
 BIT3 = 000010  
 BIT7 = 000200  
 CKSWR = 104407  
 DFC = 021236  
 DH26 = 020771  
 DISPRE = 000174  
 DT1 = 021142  
 DT26 = 021230  
 DWARFT = 001454  
 EM12 = 020463  
 EM26 = 020572  
 ENDP = 013644  
 GTSWR = 104406  
 IOTST2 = 015112  
 LF = 000012  
 PIRQ = 177772  
 PR1 = 000040  
 PR5 = 000240  
 PSW = 177776  
 RESVEC = 000010  
 R1 = %000001  
 R5 = %000005  
 STACK = 001100  
 SWREG = 000176  
 SW02 = 000004  
 SW06 = 000100  
 SW1 = 000002  
 SW13 = 020000  
 SW3 = 000010  
 SW7 = 000200  
 TVVEC = 000060  
 TRTC = 020206  
 TSTCNT = 001442  
 TST11 = 003326  
 TST15 = 003716

ABR = 001400  
 ADDW0 = 000000  
 ADDW12 = 000000  
 ADDW2 = 000000  
 ADDW6 = 000000  
 ADEVCT = 000000  
 AFATAL = 000000  
 AMADR4 = 000000  
 AMAMS4 = 000000  
 AMTYP1 = 000000  
 ANYKEY = 014704  
 APTCSU = 000040  
 ASK = 001452  
 AUNIT = 000000  
 BIT0 = 000001  
 BIT03 = 000010  
 BIT07 = 000200  
 BIT10 = 002000  
 BIT14 = 040000  
 BIT4 = 000020  
 BIT8 = 000400  
 CR = 000015  
 DH1 = 020615  
 DH3 = 020641  
 DR = 001414  
 DT12 = 021176  
 DT3 = 021154  
 EMTVEC = 000030  
 EM16 = 020512  
 EM3 = 020352  
 ERCNT = 001436  
 HT = 000011  
 IOTST3 = 015160  
 LOOP = 002312  
 PIRQVE = 000240  
 PR2 = 000100  
 PR6 = 000300  
 PWRVEC = 000024  
 ROTATE = 001432  
 R2 = %000002  
 R6 = %000006  
 START = 001534  
 SW0 = 000001  
 SW03 = 000010  
 SW07 = 000200  
 SW10 = 002000  
 SW14 = 040000  
 SW4 = 000020  
 SW8 = 000400  
 TPVEC = 000064  
 TRTVEC = 000014  
 TSTSTR = 001474  
 TST12 = 003424  
 TST16 = 004006

ADDW1 = 000000  
 ADDW13 = 000000  
 ADDW3 = 000000  
 ADDW7 = 000000  
 ADEVM = 000000  
 AMADR1 = 000600  
 AMAMS1 = 000000  
 AMSGAD = 000000  
 AMTYP2 = 000000  
 ANY2 = 014744  
 APTENV = 000001  
 ASR = 001376  
 AUSWR = 000000  
 BIT00 = 000001  
 BIT04 = 000020  
 BIT08 = 000400  
 BIT11 = 004000  
 BIT15 = 100000  
 BIT5 = 000040  
 BIT9 = 001000  
 CRLF = 000200  
 DH12 = 020721  
 DH4A = 020665  
 DR2 = 001416  
 DT20 = 021204  
 DT4 = 021166  
 EM1 = 020272  
 EM2 = 020324  
 EM4 = 020400  
 ERRVEC = 000004  
 IOTRD = 020032  
 IOTVEC = 000020  
 MDEVCT = 001440  
 PRIOR = 001412  
 PR3 = 000140  
 PR7 = 000340  
 RDCHR = 104410  
 RSTART = 002162  
 R3 = %000003  
 R7 = %000007  
 STKMT = 177774  
 SW00 = 000001  
 SW04 = 000020  
 SW08 = 000400  
 SW11 = 004000  
 SW15 = 100000  
 SW5 = 000040  
 SW9 = 001000  
 TRAPVE = 000034  
 TSCLC = 001420  
 TST1 = 002400  
 TST13 = 003522  
 TST17 = 004076

ADDW2 = 000000  
 ADDW10 = 000000  
 ADDW14 = 000000  
 ADDW4 = 000000  
 ADDW8 = 000000  
 AENV = 000000  
 AMADR2 = 000000  
 AMAMS2 = 000000  
 AMSGLG = 000000  
 AMTYP3 = 000000  
 APASS = 000000  
 APTSIZ = 000200  
 ASWREG = 000000  
 AVECT1 = 000440  
 BIT01 = 000002  
 BIT05 = 000040  
 BIT09 = 001000  
 BIT12 = 010000  
 BIT2 = 000004  
 BIT6 = 000100  
 BPTVEC = 000014  
 DDISP = 177570  
 DH20 = 020736  
 DISPLA = 001142  
 DSWR = 177570  
 DT22 = 021216  
 DWARF = 001450  
 EM11 = 020444  
 EM20 = 020543  
 EM5 = 020421  
 EXS = 001444  
 IOTST1 = 015034  
 LCNT = 001446  
 PC = %000007  
 PRO = 000000  
 PR4 = 000200  
 PS = 177776  
 RDLIN = 104411  
 R0 = %000000  
 R4 = %000004  
 SP = %000006  
 SWR = 001140  
 SW01 = 000002  
 SW05 = 000040  
 SW09 = 001000  
 SW12 = 010000  
 SW2 = 000004  
 SW6 = 000100  
 TBITVE = 000014  
 TRFRO = 020210  
 TSCLD = 001422  
 TST10 = 003230  
 TST14 = 003620  
 TST2 = 002500

TST20	004150	TST21	004232	TST22	004360	TST23	004506
TST24	004602	TST25	004662	TST26	004766	TST27	005034
TST3	002542	TST30	005102	TST31	005212	TST32	005424
TST33	005546	TST34	005610	TST35	005656	TST36	006006
TST37	006136	TST4	002640	TST40	006266	TST41	006416
TST42	006546	TST43	006706	TST44	007050	TST45	007110
TST46	007236	TST47	007350	TST5	002736	TST50	007454
TST51	007530	TST52	007604	TST53	007670	TST54	007762
TST55	010116	TST56	010250	TST57	010404	TST6	003034
TST60	010540	TST61	010674	TST62	011030	TST63	011164
TST64	011442	TST65	011652	TST66	012162	TST67	012426
TST7	003132	TST70	012776	TST71	013110	TST72	013254
TST73	013642	TYPBN =	104405	TYPE =	104401	TYPOC =	104402
TYPON =	104404	TYPOS =	104403	UTEST	001434	VECTP	001404
VECT1	001402	VECT2	001406	VECT2P	001410	WRNMI	021015
WSTART =	001514	SAPTHD	001000	SATYC	017432	SATY1	017406
SATY3	017414	SATY4	017424	SAUTOB	001134	SBASE	001250
SBOADR	001122	SBDDAT	001126	SBELL	001164	SBIN	015522
SCDW1	001254	SCHARC	017402	SCKSWR	016350	SCMTAG	001100
SCM3 =	000000	SCNTLG	017075	SCNTLU	017070	SCPUOP	001222
SCRLF	001171	SDEVCT	001204	SDEVM	001252	SDOAGN	014674
SENDAD	014664	SEDOCT	014432	SENULL	014700	SENV	001214
SENVN	001215	SEOP	014376	SEOPCT	014424	SERFLG	001103
SERMAX	001115	SERROR	015524	SERRPC	001116	SERRTB	001256
SERRTY	015732	SERTTL	001112	SESCAP	001162	SETABL	001214
SETEND	001256	SFATAL	001176	SFFLG	017652	SFILLC	001156
SFILLS	001155	SGOADR	001120	SGODAT	001124	SGET42	014654
SGTSWR	016420	SHD =	000001	SHIBTS	001000	SICNT	001104
SILLUP	020014	SINTAG	001135	SITEMB	001114	SLF	001172
SLFLG	017651	SLPADR	001106	SLPERR	001110	\$MADR1	001226
\$MADR2	001232	\$MADR3	001236	\$MADR4	001242	\$MAIL	001174
\$MAMS1	001224	\$MAMS2	001230	\$MAMS3	001234	\$MAMS4	001240
\$MBADR	001002	\$MFLG	017650	\$MNEW	017113	\$MSGAD	001210
\$MSGLG	001212	\$MSGTY	001174	\$MSWR	017102	\$MTYP1	001225
\$MTYP2	001231	\$MTYP3	001235	\$MTYP4	001241	\$MXCNT	016346
\$NULL	001154	\$NWTST =	000001	\$OCNT	015444	\$OMODE	015446
\$OVER	016332	\$PASS	001202	\$PASTM	001006	\$POWER	020022
\$PWRDN	017654	\$PWARMG	020010	\$PWRUP	017726	\$QUES	001170
\$RDCHR	016632	\$RDLIN	016752	\$RDSZ =	000010	\$RTNAD	014676
\$SAVR6	020020	\$SCOPE	016066	\$SETUP =	000117	\$STUP =	177777
\$SVLAD	016276	\$SVPC =	000254	\$SWR =	167400	\$SWREG	001216
\$SWRMK =	000000	\$STESIN	001200	\$TIMES	001160	\$TKB	001146
\$TKS	001144	\$TMPD	001424	\$TMP1	001426	\$TMP3	001430
\$TN =	000074	\$TPB	001152	\$TPFLG	001157	\$TPS	001150
\$TRAP	020212	\$TRAP2	020234	\$TRP =	000012	\$TRPAD	020246
\$TSTM	001004	\$TSTNM	001102	\$TTYIN	017063	\$TYPBN	015450
\$TYPE	017124	\$TYPEC	017336	\$TYPEX	017404	\$TYPOC	015246
\$TYPON	015262	\$TYPOS	015222	\$UNIT	001206	\$UNIM	001010
\$USWR	001220	\$VECT1	001244	\$VECT2	001246	\$XTSTR	016102
\$SET4 =	000000	\$OFILL	015445	.\$X =	001000	.	= 021242

ERROR DETECTED: C

