

The microfiche card displays a grid of 100 frames of data. The frames are arranged in 10 rows and 10 columns. Each frame contains a different set of data, likely related to the 'DATA RELIABILITY' and 'MD-11-DZTAE-C' project. The data is presented in various formats, including tables, charts, and text blocks. The frames are very small and densely packed, typical of microfiche technology.

100 100 100 100  
100 100 100 100  
100 100 100 100  
100 100 100 100

B01

EOF1DZRRDESEG411

00010000

770610

PDP10 411

RKHDR1DZTRECSEQ

00010000

770610

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZTAE-C-D  
PRODUCT NAME: TALL DATA RELIABILITY  
DATE CREATED: 16 MARCH 77  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHOR: JIM LACEY

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1973,1977 BY DIGITAL EQUIPMEN CORPORATION

## CONTENTS

1. ABSTRACT
2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 STORAGE
  - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURE
4. STARTING PROCEDURE
  - 4.1 CONTROL SWITCH SETTINGS
  - 4.2 STARTING ADDRESS
  - 4.3 PROGRAM & OPERATOR ACTION
5. OPERATING PROCEDURE
  - 5.1 OPERATIONAL SWITCH SETTINGS
  - 5.2 SUBROUTINE ABSTRACTS
6. ERRORS
  - 6.1 ERROR TYPES
  - 6.2 DATA ERRORS
  - 6.3 ERROR RECOVERY
7. RESTRICTIONS
8. MISCELLANEOUS
  - 8.1 EXECUTION TIME
  - 8.2 STACK POINTER
  - 8.3 END OF TEST
  - 8.4 DRIVE COMPATIBILITY
  - 8.5 DATA FORMAT
  - 8.6 TEST TYPEOUT
9. PROGRAM DESCRIPTION
  - 9.1 FORMAT PASS
  - 9.2 READ ONLY PASS
  - 9.3 WRITE ONLY PASS

1. ABSTRACT

THIS PROGRAM COLLECTS STATISTICAL INFORMATION PERTAINING TO THE DATA RELIABILITY OF THE TAI1/TUGO WHEN RUN FOR EXTENDED PERIODS OF TIME. IT USES A NUMBER OF DIFFERENT PARAMETERS CONTROLLING THE DATA PATTERNS, THE NUMBER OF BYTES PER BLOCK (RECORD) AND THE NUMBER OF BLOCKS PER FILE.

2. REQUIREMENTS2.1 EQUIPMENT

PDP-11 COMPUTER WITH OR WITHOUT HARDWARE SWITCH REGISTER WITH CONSOLE TELETYPE, AND A TAI1 CASSETTE

2.2 STORAGE

THIS PROGRAM REQUIRES APPROX. 4K STORAGE.

2.3 PRELIMINARY PROGRAMS

MAINDEC-11-DZTAA  
 MAINDEC-11-DZTAB  
 MAINDEC-11-DZTAC  
 MAINDEC-11-DZTAD

3. LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES OR A CASSETTE TAPE.

4. STARTING PROCEDURE4.1 CONTROL SWITCH SETTINGS

SEE 5.1.

4.2 STARTING ADDRESSES

2100 NORMAL STARTING ADDRESS  
 2104 SELECT DRIVE (S) BEFORE STARTING TEST  
 2110 SELECT DRIVE(S) AND ADDRESSES BEFORE STARTING TEST

## 4.3 PROGRAM &amp; OPERATOR ACTION

1. LOAD PROGRAM INTO MEMORY (SEE SECTION 3.)
2. LOAD A WRITE ENABLED CASSETTE IN BOTH DRIVES
3. LOAD ADDRESS 200.
4. SET SWITCHES (SEE SECTION 4.1)
5. PRESS START.

\*\*\* NOTE: IF USING THE SOFTWARE SWITCH REGISTER THE PROGRAM WILL TYPE "SWR=XXXXXX NEW=" AFTER TYPING THE NAME OF THE PROGRAM.

## 4.3.1 DRIVE SELECTION

STARTING THE PROGRAM AT 200 WILL RESULT IN AUTOMATIC SELECTION OF DRIVES "A" AND "B" TO BE TESTED.

STARTING THE PROGRAM AT 204 OR 210 ALLOWS THE OPERATOR TO SELECT THE DRIVE(S) TO BE TESTED.

THE PROGRAM WILL TYPE "DRIVE(S)?".

EITHER OR BOTH DRIVES CAN BE SELECTED BY TYPING "A" AND/OR "B" FOLLOWED BY A CARRIAGE RETURN.

## 4.3.1.1 DRIVE SELECTION EXAMPLES

DRIVE(S)? A,B	:DRIVES A AND B SELECTED
DRIVE(S)? AB	:DRIVES A AND B SELECTED
DRIVE(S)? B,A	:DRIVES B AND A SELECTED
DRIVE(S)? B	:DRIVE B SELECTED
DRIVE(S)? SB?	:DRIVE S IS IMPOSSIBLE
DRIVE(S)?	:ASK FOR DRIVE

## 4.3.2 ADDRESS SELECTION

STARTING THE PROGRAM AT 210 ALLOWS THE OPERATOR TO CHANGE THE "CONTROL AND STATUS" AND "DATA BUFFER" REGISTER ADDRESSES, THE VECTOR ADDRESS AND THE PRIORITY LEVEL.

THE PROGRAM WILL ASK FOR THE DRIVES TO BE TESTED AS PER 4.3.1. AFTER THE DRIVES HAVE BEEN SELECTED IT WILL ASK FOR:

1. BUS ADDRESS OF THE CONTROL AND STATUS REGISTER (TACS)
2. VECTOR ADDRESS
3. PRIORITY LEVEL

AND THE OPERATOR MUST RESPOND WITH THE DESIRED PARAMETER OR A CARRIAGE RETURN (WHICH IMPLIES LEAVE AS IS). WHEN ALL PARAMETERS HAVE BEEN DEFINED THE PROGRAM WILL TYPE THEM BACK OUT AND ASK IF THEY ARE OK AT WHICH TIME THE OPERATOR RESPONDES WITH A "Y" OR A "CARRIAGE RETURN" FOR "YES" ANYTHING ELSE IS A "NO".

## 4.3.2.1 ADDRESS SELECTION EXAMPLES

```
DRIVES(S) A
TACS? 177500
VECTOR? 260
PRIORITY? 6
TACS=177500 TADB=177502 VECTOR=000260 PRIORITY=000300
OK?
```

```
DRIVES(S) A,B
TACS? 470
VECTOR?
PRIORITY?
TACS=177470 TADB=177472 VECTOR=000260 PRIORITY=000300
OK?
```

GO1

# H01

## 5. OPERATING PROCEDURE

### 5.1 OPERATIONAL SWITCH SETTINGS

IF THE DIAGNOSTIC IS RUN ON A CPU WITHOUT A SWITCH REGISTER THEN A SOFTWARE SWITCH REGISTER IS USED WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER. IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST OR IF ONE DOES AND IT CONTAINS ALL ONES (177777) THEN THE SOFTWARE SWITCH REGISTER (LOC. 176) IS USED.

#### CONTROL:

THIS PROGRAM ALSO SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOC. 176) FROM THE TTY. THIS CAN BE ACCOMPLISHED BY DOING THE FOLLOWING:

- 1) TYPE CONTROL G <↑G>: THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE PROGRAM.
- 2) THE MACHINE WILL THEN TYPE: SWR=XXXXXXNEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)
- 3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:
  - A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED)  
IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.
  - B) IF A CONTROL U <↑U> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 2.

WITH SW<15:10>=0 THE PROGRAM WILL PRINT OUT ON ERRORS AND CONTINUE IN TEST.  
THE SWITCH SETTINGS ARE:

SW<15>=1...HALT ON ERROR  
SW<14>=1...LOOP ON TEST  
SW<13>=1...INHIBIT ERROR TYPEOUTS  
SW<10>=1...RING BELL ON ERROR  
SW<09>=1...HALT AFTER NEXT "END-OF-TEST" TYPEOUT  
SW<08>=1...AT NEXT "END-OF-TAPE" (EOT) GO TO "END-OF-TEST"  
SW<07>=1...PERFORM PASS AS PER SWR<1:0>  
SWR<1:0>=00=FORMAT  
SWR<1:0>=01=READ ONLY  
SWR<1:0>=10=WRITE ONLY  
SWR<1:0>=11=READ ONLY



## 5.2 SUBROUTINE ABSTRACTS

## 5.2.1 SCOPE

THIS SUBROUTINE CALL (VIA AN IOT INSTRUCTION) IS PLACED AT AN OPTIMUM POSITION IN THE INSTRUCTION SECTION OF THE "FORMAT", "READONLY" AND "WRITEONLY" CODE.

IF SWR<14>=1 THE PROGRAM WILL LOOP THROUGH A SPECIFIC SEQUENCE DEPENDING ON THE TYPE OF PASS BEING PERFORMED.  
 \*\*\* THIS ROUTINE SUPPORTS THE S/W SWITCH REG FUNCTIONS

## 5.2.1.1 FORMAT PASS SCOPE LOOP

1. SETUP FOR A WRITE
2. WRITE
3. BACKSPACE BLOCK GAP
4. SETUP FOR A READ
5. READ
6. REPEAT STEPS 1-5 UNTIL "EOT"

## 5.2.1.2 READONLY PASS SCOPE LOOP

1. SETUP FOR READ
2. READ
3. CHECK FOR SYNC & DATA ERROR
4. BACK SPACE BLOCK GAP
5. REPEAT STEPS 2-4 INDEFINITELY

## 5.2.1.3 WRITEONLY PASS SCOPE LOOP

1. SETUP FOR WRITE
2. WRITE
3. REPEAT STEP 2 UNTIL "EOT"

## 5.2.2 TRAPCATCHER

A ".+2" - "HALT" SEQUENCE IS REPEATED FROM LOC. 0 TO LOC. 776 TO CATCH ANY UNEXPECTED TRAPS. THUS, ANY UNEXPECTED TRAPS WILL HALT AT THE DEVICE TRAP VECTOR +2.

5.2.3 ERROR  
---

THIS SUBROUTINE CALL (VIA A EMT INSTRUCTION) IS USED TO REPORT ALL ERRORS. (REFER TO 6.)

5.2.4 TRAP  
----

A NUMBER OF SUBROUTINES ARE CALLED BY THE TRAP INSTRUCTION. FOLLOWING IS THE CALLS USED AND THE STARTING ADDRESS OF THE ROUTINE.

## 5.2.4.1 TYPE (\$TYPE)

TYPE AN ASCIZ STRING ON THE TTY

## 5.2.4.2 RDCHR(\$RDCHR)

READ A SINGLE ASCII CHARACTER FROM THE TTY

## 5.2.4.3 RDLIN(\$RDLIN)

READ AN ASCII STRING FROM THE TTY

5.2.6 THE FOLLOWING SUBROUTINES ARE CALLED BY A 'JSR'.  
-----

5.2.6.1 A2OCT

THIS ROUTINE CHANGES AN ASCII STRING TO AN OCTAL NUMBER.

5.2.6.2 ASKDRV

THIS ROUTINE IS USED TO ASK THE OPERATOR WHICH DRIVE(S)  
ARE TO BE TESTED

5.2.6.3 ASKADR

THIS ROUTINE IS USED TO INPUT THE ADDRESSES OF THE "TACS"  
AND THE VECTOR TO USE.

5.2.6.4 TYPERR

THIS ROUTINE IS USED TO TYPE OUT THE "ERROR" DATA

5.2.6.5 CSRERR

THIS ROUTINE IS USED WHEN AN ERROR  
IS DETECTED. IT WILL EXAMINE THE  
"CONTROL AND STATUS" REGISTER TO  
DETERMINE THE TYPE OF ERROR AND  
TAKE THE APPROPRIATE ACTION.

5.2.6.6 SETUPW

THIS ROUTINE IS USED TO SETUP THE  
PARAMETER BLOCK AND THE WRITE  
BUFFER BEFORE STARTING A "WRITE" FUNCTION

5.2.6.7 FILL

USE TO FILL THE WRITE BUFFER  
WITH A DATA PATTERN.

5.2.6.8 SETUPR

THIS ROUTINE IS USED TO SETUP THE  
PARAMETER BLOCK BEFORE DOING A  
"READ" FUNCTION.

5.2.6.9 SYNCK

THIS ROUTINE IS CALLED AFTER  
PERFORMING A "READ".  
IT CHECKS THE FIRST FOUR BYTES  
OF THE DATA TO INSURE THAT THEY  
CONTAIN THE PROPER FILE AND  
BLOCK NUMBERS.

## 5.2.6.10 DATCMP

THIS ROUTINE IS USED TO CHECK THE DATA IN THE READ BUFFER TO INSURE IT IS CORRECT.

## 5.2.6.11 CNTSFT

THIS ROUTINE IS USED TO COUNT SOFT DATA ERRORS

## 5.2.6.12 CNTHRD

THIS ROUTINE IS USED TO COUNT HARD DATA ERRORS

## 5.2.6.13 SAVRGI OR SAVREG

ROUTINE TO SAVE ALL THE REGISTERS

## 5.2.6.14 CASSETTE PRIMITIVE

THIS IS THE CASSETTE DRIVER

## 5.2.6.15 CASINT

CASSETTE INTERRUPT HANDLER

## 5.2.6.16 DBCD

CHANGES A DOUBLE LENGTH BINARY NUMBER TO A DECIMAL ASCIZ STRING

## 5.2.6.17 SBCD

CHANGES A SINGLE LENGTH BINARY NUMBER TO A DECIMAL ASCIZ STRING.

## 5.2.6.18 SUPRES

TYPES A DECIMAL ASCII STRING SUPPRESSING LEADING ZEROS.

## 5.2.6.19 TYPF

USED TO TYPE THE ASCIZ STRING IMMEDIATELY FOLLOWING THE CALL.

## 5.2.6.20 TPDRV

TYPES THE DRIVE TO BE TESTED

## 5.2.6.21 EXAM

THIS ROUTINE EXAMINES THE SELECTED DRIVES TO INSURE THEY ARE AVAILABLE FOR TESTING.

## 5.2.6.22 \$B2OCT

TYPES AN OCTAL NUMBER

6. ERRORS  
-----

THERE ARE A NUMBER OF ERRORS THAT CAN OCCUR IN THIS PROGRAM. WHEN AN ERROR IS ENCOUNTERED THE CALL TO THE ERROR(ERROR) ROUTINE IS MADE AND IF SW<13> IS NOT SET AN ERROR MESSAGE PERTAINING TO THE ERROR WILL BE TYPED. EACH ERROR TYPE OUT WILL CONTAIN THE FOLLOWING:

1. AN ERROR MESSAGE
2. A DATA HEADER
3. A DATA STRING

REFER TO THE LISTING UNDER \$ERRTB FOR THE DIFFERENT ERRORS THAT CAN OCCUR.

6.1 ERROR TYPES  
-----

THE ERRORS THAT OCCUR IN THIS PROGRAM FALL INTO THREE (3) CATEGORIES DEFINED AND EXPLAINED AS FOLLOWS:

## 6.1.1 PRETEST ERROR

THESE ERRORS WILL BE DETECTED BEFORE TRYING TO TEST THE DATA RELIABILITY OF THE TAIL/TU60.

## 6.1.2 NON-FATAL ERROR

THESE ERRORS WILL BE DUE TO "CRC" OR "DATA" FAILURES WHICH WILL BE REPORTED AS THEY OCCUR. AFTER REPORTING THE ERROR THE PROGRAM WILL CONTINUE TESTING.

## 6.1.3 FATAL ERROR

THIS TYPE OF ERROR WILL BE THE RESULT OF ANY KIND OF ERROR THAT CAUSES THE PROGRAM TO LOSE TRACK OF THE TAPE POSITION, OR THE MAXIMUM NUMBER OF DATA ERRORS HAVE OCCURRED.

THIS ERROR WILL BE REPORTED WHEN IT OCCURS. THEN THE PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

6.2 DATA ERRORS  
-----

THERE ARE TWO TYPES OF DATA ERRORS THAT CAN OCCUR WHICH ARE DEFINED AND EXPLAINED AS FOLLOWS:

## 6.2.1 SOFT ERROR

A SOFT ERROR IS BY DEFINITION ANY "CRC" OR "READ DATA" ERROR THAT OCCURS WHILE READING A BLOCK OF DATA. A SOFT ERROR WILL INVOKE A REREAD OF THE BLOCK.

## 6.2.2 HARD ERROR

A HARD ERROR IS DEFINED AS ANY "CRC" OR "READ DATA" ERROR THAT OCCURS ON THE INITIAL READ OF A BLOCK OF DATA AND CAN NOT BE READ CORRECTLY WITHIN THREE (3) RETRYS.

## 6.3 ERROR RECOVERY

## 6.3.1 PRETEST ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THEN DEPENDING ON HOW THE PROGRAM WAS STARTED IT WILL ASK FOR THE DRIVES AND ADDRESSES FOR TESTING OR RETURN TO MONITOR.

## 6.3.2 NON-FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED AND THE PROGRAM WILL CONTINUE IN TEST.

## 6.3.3 FATAL ERROR

WHEN THIS TYPE OF ERROR OCCURS IT WILL BE REPORTED. THE PROGRAM WILL ABORT THE TEST AND GO TO THE "END-OF-TEST" TYPEOUT.

## 7. RESTRICTIONS

BEFORE STARTING THE PROGRAM THE OPERATOR MUST INSURE THAT A CASSETTE IS LOADED AND WRITE ENABLED IN THE DRIVE(S) TO BE TESTED.

## 8. MISCELLANEOUS

## 8.1 EXECUTION TIME

TESTING THE TA11/TU60 TO SPECIFICATION TAKES APPROXIMATELY 2 HOURS 30 MINUTES WITH EACH DRIVE TAKING 75 MINUTES.

## 8.2 STACK POINTER

STACK IS INITIALLY SET TO 1100.

### 8.3 END OF TEST

-----

WITH ALL SWITCHES ON A "0" THE END OF TEST TYPEOUT WILL OCCUR WHEN THE PROGRAM COMPLETES 18 TAPE PASSES FROM "BOT" TO "EOT" ON THE DRIVE UNDER TEST OR A FATAL ERROR OCCURS.

#### 8.3.1 EXAMPLE OF AN END-OF TEST TYPEOUT

```
*** END-OF-TEST ***
SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1471488
BYTES WRITTEN=369000
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16
```

### 8.4 DRIVE COMPATIBILITY

-----

THE COMPATIBILITY BETWEEN DRIVES CAN BE TESTED BY DOING A "FORMAT" PASS ON ONE DRIVE AND THEN READING IT ON ANOTHER DRIVE.

#### 8.4.1 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #1

-----

THIS EXAMPLE FORMATS ON DRIVE A AND READS FROM DRIVE B

##### 8.4.1.1 "FORMAT" DRIVE "A"

1. PLACE A WRITE ENABLED TAPE IN DRIVE "A"
2. INSURE DRIVE "B" IS EMPTY
3. LOAD ADDRESS 200
4. SET SW09 AND SW08 TO "1" ALL OTHERS TO "0"
5. PRESS START
6. PROGRAM WILL PERFORM A "FORMAT" PASS ON DRIVE "A". TYPE "END-OF-TEST" STATISTICS AND HALT

##### 8.4.1.2 "READ" DRIVE "B"

1. REMOVE THE TAPE FROM DRIVE "A" AND PLACE IT IN DRIVE "B"
2. LOAD ADDRESS 200
3. SET SW09, SW08, SW07 AND SW00 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS ON DRIVE "B", TYPE "END-OF-TEST" STATISTICS AND HALT.

## 8.4.2 DRIVE COMPATIBILITY PROCEDURE EXAMPLE #2

THIS EXAMPLE READS KNOWN GOOD TAPE(S)

### 8.4.2.1 "READ"

1. PLACE THE KNOWN GOOD TAPE(S) IN THE DRIVE(S) TO BE TESTED. (NOTE: IT MIGHT BE WISE TO HAVE THE TAPE(S) WRITE LOCKED.)
2. LOAD ADDRESS 200
3. SET SW08, SW07 AND SW00 TO A "1" ALL OTHERS TO A "0"
4. PRESS START
5. PROGRAM WILL PERFORM A "READONLY" PASS AND TYPE "END-OF-TEST" STATISTICS ON DRIVE(S) TO BE TESTED.

### 8.5 DATA FORMAT

THE DATA FORMAT USED IN THIS PROGRAM WILL RESULT IN APPROXIMATELY ELEVEN (11) FILES OF 8192 BYTES TO BE WRITTEN ON TAPE.

#### 8.5.1 FILE STRUCTURE

EACH FILE WILL CONSIST OF SIXTEEN (16) BLOCKS OF DATA, WITH EACH BLOCK CONTAINING AN UNIQUE DATA PATTERN.

#### 8.5.2 BLOCK STRUCTURE

EACH BLOCK WILL HAVE AN "ID" CODE AS THE FIRST FOUR (4) BYTES OF DATA. THIS "ID" WILL BE THE FILE NUMBER AND BLOCK NUMBER AND THEIR COMPLEMENTS. THE DATA FOLLOWING THE "ID" IS A PATTERN THAT REPEATS ITSELF EVERY EIGHT (8) BYTES FOR THE LENGTH OF THE BLOCK.

#### 8.5.3 BLOCK SIZE AND PATTERN

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
0	1024	314 063 314 063 146 231 146 231
1	512	314 103 063 146 231 063 146



8.5.3 (CONT.)

BLOCK NUMBER	BLOCK SIZE	BLOCK PATTERN
2	1024	001 002 004 010 020 040 100 200
3	256	177 277 337 357 367 372 375 376
4	1024	252 1125 2125 1125 2125 1255 2555 1255
5	128	000 111 222 333 044 155 266 377
6	1024	000 044 111 222 333 444 555 666 777

8.5.3 (CONT.)

BLOCK NUMBER -----	BLOCK SIZE -----	BLOCK PATTERN -----
7	64	000 222 044 266 111 333 155 377
8	1024	001 003 007 017 037 077 177 377
9	32	376 374 370 360 340 300 200 000
10	128	001 376 002 374 004 373 010 367
11	256	020 357 040 337 100 277 200 177

8.5.3 (CONT.)

<u>BLOCK NUMBER</u>	<u>BLOCK SIZE</u>	<u>BLOCK PATTERN</u>
12	512	000 000 000 000 000 000 000 000
13	1024	377 377 377 377 377 377 377
14	32	000 377 000 377 000 377 000 377
15	128	017 360 207 170 303 074 341 036

8.6 TEST TYPEOUT

THE FOLLOWING EXAMPLE SHOWS A TYPICAL TYPEOUT WHERE BOTH DRIVES WERE TESTED AND NO ERRORS OCCURRED.

MAINDEC-11-DZTAE-A

DRIVE A AND DRIVE B WILL BE TESTED

```

*** FORMAT *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** WRITE *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** FORMAT *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** WRITE *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A
*** READ *** DRIVE A

```

\*\*\* END-OF-TEST \*\*\*

```

SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1476608
BYTES WRITTEN=370209
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16

```

```

*** FORMAT *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** WRITE *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** FORMAT *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** WRITE *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B
*** READ *** DRIVE B

```

\*\*\* END-OF-TEST \*\*\*

```

SOFT ERRORS=0
HARD ERRORS=0
BYTES READ=1504096
BYTES WRITTEN=376868
TAPE PASSES=18
FILES/PASS=12
BLOCKS/FILE=16

```

9. PROGRAM DESCRIPTION  
-----

THIS PROGRAM IS DESIGNED AROUND THREE PRIMARY ROUTINES THAT WILL TRANSFER DATA TO AND/OR FROM THE TA11/TU60 GOING FROM "BOT" TO "EOT" OF THE TAPE. EACH OF THESE ROUTINES MAKE USE OF COMMON SUBROUTINES TO MANIPULATE TAPE MOTION, KEEP TRACK OF TAPE POSITION, SETUP DATA BUFFERS AND CHECK, COUNT AND REPORT ERRORS. THESE ROUTINES ARE DEFINED AND EXPLAINED BELOW.

9.1 FORMAT PASS  
-----

THIS IS A WRITE, BACKSPACE AND READ ROUTINE STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. WRITE A BLOCK OF DATA
2. BACKSPACE A BLOCK GAP
3. READ THE BLOCK
4. CHECK FOR SYNC ERROR
5. CHECK FOR DATA ERROR
6. REPEAT STEPS 1-5 SIXTEEN TIMES
7. WRITE A FILE GAP
8. REPEAT STEPS 1-7 UNTIL "EOT"

9.2 WRITEONLY PASS  
-----

THIS IS A WRITE ONLY ROUTINE. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. WRITE SIXTEEN BLOCKS OF DATA
2. WRITE A FILE GAP
3. REPEAT STEPS 1 & 2 TO "EOT"

9.3 READONLY PASS  
-----

THIS IS A READ ONLY ROUTINE AND REQUIRES THAT A "FORMAT" OR "WRITEONLY" PASS HAS ALREADY BEEN PERFORMED. STARTING AT "BOT" THE FOLLOWING PROCEDURE IS PERFORMED:

1. READ A BLOCK OF DATA
2. CHECK FOR SYNC ERROR
3. CHECK FOR DATA ERROR
4. REPEAT STEPS 1-3 SIXTEEN (16) TIMES
5. SPACE FORWARD FILE GAP
6. REPEAT STEPS 1-5 UNTIL THE LAST BLOCK OF THE LAST FILE HAS BEEN READ.

13	GENERAL INFORMATION
59	OPERATIONAL SWITCH SETTINGS
77	BASIC DEFINITIONS
218	STARTING ADDRESSES
219	TRAP CATCHER
228	STARTING ADDRESS(ES)
234	COMMON TAGS
323	BLOCK SIZE TABLE
344	TABLE OF POINTERS TO THE DIFFERENT PATTERNS
366	DATA PATTERNS
420	PARAMETER BLOCK USED WITH ALL FUNCTIONS
428	ERROR POINTER TABLE
530	START OF TEST
552	INITIALIZE THE COMMON TAGS
586	TYPE PROGRAM NAME
593	GET VALUE FOR SOFTWARE SWITCH REGISTER
612	"FORMAT" ROUTINE
964	"READ ONLY" ROUTINE
1101	"WRITE ONLY" ROUTINE
1187	CHECK "EOTS"
1211	END OF PASS ROUTINE
1307	SCOPE HANDLER ROUTINE
1340	ERROR HANDLER ROUTINE
1382	ERROR TIMEOUT ROUTINE
1487	DETERMINE CONTROL AND STATUS ERROR
1575	ROUTINE TO SETUP FOR A WRITE OPERATION
1597	ROUTINE TO FILL THE BUFFER BEFORE A WRITE
1642	ROUTINE TO SETUP FOR A READ
1660	ROUTINE TO CHECK FOR SYNC PROBLEMS
1699	ROUTINE TO CHECK THE READ DATA
1749	COUNT SOFT DATA ERROR
1766	COUNT HARD ERROR
1780	SAVE AND RESTORE RD-RS ROUTINES
1829	CASSETTE PRIMITIVES ROUTINE
1879	CASSETTE INTERRUPT HANDLER
1924	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
1986	SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
2004	TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS
2027	READ AN OCTAL NUMBER FROM THE TTY
2066	ROUTINE TO TYPE DRIVE
2084	ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2119	ROUTINE TO INPUT CSR, DSR, AND VECTOR ADDRESS AND PRIORITY
2180	ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2210	TYPE ROUTINE
2280	TTY INPUT ROUTINE
2419	BINARY TO OCTAL (ASCII) AND TYPE
2496	TRAP DECODER
2519	TRAP TABLE
2540	POWER DOWN AND UP ROUTINES
2585	DATA TABLE POINTERS AND DATA FORMATS FOR ERRORS
2623	ASCII MESSAGES
2705	READ AND WRITE BUFFER



57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112

.SBTTL OPERATIONAL SWITCH SETTINGS

```

*
* SWITCH USE
* -----
* 15 HALT ON ERROR
* 14 LOOP ON TEST
* 13 INHIBIT ERROR TYPEOUTS
* 10 DING BELL ON ERROR
* 9 HALT AFTER NEXT "END-OF-TEST" TYPE OUT
* 8 AT NEXT "EOT" GOTO "END-OF-TEST"
* 7 PERFORM AS PER SWR<1:0>
*
* 00=FORMAT
* 01=READONLY
* 10=WRITEONLY
* 11=READONLY

```

.SBTTL BASIC DEFINITIONS

```

*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

*MISCELLANEOUS DEFINITIONS
000011 HT= 11 ;;CODE FOR HORIZONTAL TAB
000012 LF= 12 ;;CODE FOR LINE FEED
000015 CR= 15 ;;CODE FOR CARRIAGE RETURN
000200 CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776 PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
177774 STKLMT= 177774 ;;STACK LIMIT REGISTER
177772 PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
177570 DSWR= 177570 ;;HARDWARE SWITCH REGISTER
177570 DDISP= 177570 ;;HARDWARE DISPLAY REGISTER

```

.SBTTL GENERAL PURPOSE REGISTER DEFINITIONS

```

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

```

.SBTTL PRIORITY LEVEL DEFINITIONS

```

000000 PR0= 0 ;;PRIORITY LEVEL 0
000040 PR1= 40 ;;PRIORITY LEVEL 1
000100 PR2= 100 ;;PRIORITY LEVEL 2
000140 PR3= 140 ;;PRIORITY LEVEL 3
000200 PR4= 200 ;;PRIORITY LEVEL 4
000240 PR5= 240 ;;PRIORITY LEVEL 5

```



113 000300 PR6= 300 ;;PRIORITY LEVEL 6  
 114 000340 PR7= 340 ;;PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

115		SW15=	100000
116		SW14=	40000
117	100000	SW13=	20000
118	040000	SW12=	10000
119	020000	SW11=	4000
120	010000	SW10=	2000
121	004000	SW09=	1000
122	002000	SW08=	400
123	001000	SW07=	200
124	000400	SW06=	100
125	000200	SW05=	40
126	000100	SW04=	20
127	000040	SW03=	10
128	000020	SW02=	4
129	000010	SW01=	2
130	000004	SW00=	1
131	000002	.EQUIV	SW09,SW9
132	000001	.EQUIV	SW08,SW8
133		.EQUIV	SW07,SW7
134		.EQUIV	SW06,SW6
135		.EQUIV	SW05,SW5
136		.EQUIV	SW04,SW4
137		.EQUIV	SW03,SW3
138		.EQUIV	SW02,SW2
139		.EQUIV	SW01,SW1
140		.EQUIV	SW00,SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

141		BIT15=	100000
142		BIT14=	40000
143		BIT13=	20000
144		BIT12=	10000
145	100000	BIT11=	4000
146	040000	BIT10=	2000
147	020000	BIT09=	1000
148	010000	BIT08=	400
149	004000	BIT07=	200
150	002000	BIT06=	100
151	001000	BIT05=	40
152	000400	BIT04=	20
153	000200	BIT03=	10
154	000100	BIT02=	4
155	000040	BIT01=	2
156	000020	BIT00=	1
157	000010	.EQUIV	BIT09,BIT9
158	000004	.EQUIV	BIT08,BIT8
159	000002	.EQUIV	BIT07,BIT7
160	000001	.EQUIV	BIT06,BIT6
161		.EQUIV	BIT05,BIT5
162		.EQUIV	BIT04,BIT4
163		.EQUIV	BIT03,BIT3
164		.EQUIV	BIT02,BIT2
165			
166			
167			
168			

169  
170  
171  
172  
173 000004  
174 000010  
175 000014  
176 000014  
177 000014  
178 000020  
179 000024  
180 000030  
181 000034  
182 000060  
183 000064  
184 000240  
185  
186  
187  
188 000000  
189 000002  
190 000004  
191 000006  
192 000010  
193 000012  
194 000014  
195 000016  
196  
197  
198 100000  
199 040000  
200 020000  
201 010000  
202 004000  
203 002000  
204 001000  
205 000400  
206 000200  
207 000100  
208 000040  
209 000020  
210 000010  
211 000004  
212 000002  
213 000001  
214 000016  
215

```
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

:*BASIC "CPU" 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) **SCOPE**
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
;:*****

:*****TA11 FUNCTIONS*****
XWFG= 0 ;:WRITE FILE GAP FUNCTION
XWRITE= 2 ;:WRITE FUNCTION
XREAD= 4 ;:READ FUNCTION
XBSFG= 6 ;:BACK SPACE FILE GAP FUNCTION
XBSBG= 10 ;:BACK SPACE BLOCK GAP FUNCTION
XSFFG= 12 ;:SPACE FWD FILE GAP FUNCTION
XSFBG= 14 ;:SPACE FWD BLOCK GAP FUNCTION
XRWND= 16 ;:REWIND FUNCTION

:*****TA11 BIT ASSIGNMENT*****
ERROR= BIT15
CRCERR= BIT14
LEADER= BIT13
WRTLOCK=BIT12
FGAP= BIT11
TIMERR= BIT10
OFFLINE=BIT09
UNIT= BIT08
TR.REQ= BIT07
INT.EN= BIT06
READY= BIT05
ILBS= BIT04
FUNC2= BIT03
FUNC1= BIT02
FUNCO= BIT01
GO= BIT00
FUNCTION= FUNC2+FUNC1+FUNCO
```

216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230

000000  
  
000174  
000174 000000  
000176 000000  
  
000200 000137 001752  
000204 000137 002004  
000210 000137 002012

```
.SBTTL TRAP CATCHER
      .=0
; *ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
; *SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; *LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      .=174
DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
      JMP @#BEGIN1      ;; JUMP TO STARTING ADDRESS OF PROGRAM
      JMP @#BEGIN2      ; SELECT DRIVE(S) BEFORE STARTING TEST
      JMP @#BEGIN3      ; SELECT DRIVE(S) AND ADDRESSES BEFORE TESTING
; ;*****
```

.SBTTL COMMON TAGS

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

231  
232  
233  
234  
235  
236  
237 001100  
238 001100  
239 001100 000000  
240 001102 000  
241 001103 000  
242 001104 000000  
243 001106 000000  
244 001110 000000  
245 001112 000000  
246 001114 000  
247 001115 001  
248 001116 000000  
249 001120 000000  
250 001122 000000  
251 001124 000000  
252 001126 000000  
253 001130 000000  
254 001132 000000  
255 001134 000  
256 001135 000  
257 001136 000000  
258 001140 177570  
259 001142 177570  
260 001144 177560  
261 001146 177562  
262 001150 177564  
263 001152 177566  
264 001154 000  
265 001155 002  
266 001156 012  
267 001157 000  
268 001160 000000  
269  
270 001162 000000  
271 001164 000000  
272 001166 000000  
273 001170 000000  
274 001172 000000  
275 001174 000000  
276 001176 000000  
277 001200 000000  
278 001202 077  
279 001203 015  
280 001204 000012  
281  
282 001206 000000  
283 001210 000000  
284 001212 000000  
285 001214 000000 000000  
286 001220 000000 000000

. =1100  
SCMTAG: .WORD 0  
SPASS: .WORD 0  
STSTNM: .BYTE 00  
SERFLG: .BYTE 00  
SICNT: .WORD 00  
SLPADR: .WORD 00  
SLPERR: .WORD 00  
SERTTL: .WORD 00  
SITEMB: .BYTE 0  
SERMAX: .BYTE 1  
SERRPC: .WORD 0  
SGDADR: .WORD 0  
SBDADR: .WORD 0  
SGDAT: .WORD 0  
SBDAT: .WORD 0  
SAUTOB: .BYTE 0  
SINTAG: .BYTE 0  
SWR: .WORD DSWR  
DISPLAY: .WORD DDISP  
STKS: 177560  
STKB: 177562  
STPS: 177564  
STPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TPFLG: .BYTE 0  
\$REGAD: .WORD 0  
\$REG0: .WORD 0  
\$REG1: .WORD 0  
\$REG2: .WORD 0  
\$REG3: .WORD 0  
\$TMP0: .WORD 0  
\$TMP1: .WORD 0  
\$TMP2: .WORD 0  
\$TMP3: .WORD 0  
\$QUES: .ASCII ??  
\$CRLF: .ASCII <15>  
\$LF: .ASCII <12>  
\$OFTNM: .WORD 0  
\$HARDNM: .WORD 0  
\$EOTS: .WORD 0  
RBYTTL: .WORD 0.0  
WBYTTL: .WORD 0.0

: START OF COMMON TAGS  
: CONTAINS PASS COUNT  
: 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 BYTE  
: 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=YES)  
: CONTAINS THE ADDRESS FROM  
: WHICH (\$REG0) WAS OBTAINED  
: CONTAINS ((\$REGAD)+0)  
: CONTAINS ((\$REGAD)+2)  
: CONTAINS ((\$REGAD)+4)  
: CONTAINS ((\$REGAD)+6)  
: USER DEFINED  
: USER DEFINED  
: USER DEFINED  
: USER DEFINED  
: QUESTION MARK  
: CARRIAGE RETURN  
: LINE FEED  
: \*\*\*\*\*  
: NUMBER OF SOFT ERRORS  
: NUMBER OF HARD ERRORS  
: NUMBER OF TAPE PASSES  
: NUMBER OF BYTES READ  
: NUMBER OF BYTES WRITTEN

287	001224	000000		RDRYS: .WORD	0	: KEEPS COUNT OF REREADS
288	001226	000000		WRTRYS: .WORD	0	: KEEPS COUNT OF REWRITES
289	001230	000000	000000	BYTNUM: .WORD	0,0	: THE NUMBER OF BYTES "READ" OR "WRITTEN : DURING AN OPERATION
290						
291	001234	000017		LASTBK: .WORD	15.	: WILL CONTAIN THE # OF THE LAST BLOCK : AFTER A "FORMAT" OR "WRITE" PASS
292						
293	001236	000011		LASTFL: .WORD	9.	: WILL CONTAIN THE # OF THE LAST FILE : AFTER A "FORMAT" OR "WRITE" PASS
294						
295						
296						
297	001240	177500		TACSL: 177500		: ADDRESS OF TACS
298	001242	177502		TADBL: 177502		: ADDRESS OF TADB
299	001244	000260	000262	TAVEC: 260,262		: TAI1 VECTOR ADDRESS
300	001250	000300		TAPRIO: 300		: TAI1 BR LEVEL 6
301	001252	000000	000000	DRVKEY: 0,0		
302	001256	001252		DRVPNT: DRVKEY		
303	001260	000003		MAXRDS: .WORD	3	: MAX REREADS BEFORE CALLING IT A HARD ERROR
304	001262	000003		MAXERR: .WORD	3	: MAX HARD ERRORS ALLOWED
305	001264	000022		MAXEOT: .WORD	18.	: NUMBER OF TAPE PASSES BEFORE END-OF-TEST
306	001266	000000		PASCNT: .WORD	0	: COUNT # OF TAPE PASSES
307	001270	001274	000000	PSCNTL: .WORD	FORPAS,0	: CONTROLS THE TYPE OF PASS
308	001274	000001		FORPAS: .WORD	1	: 1 FORMAT PASS
309	001276	000003		RD1PAS: .WORD	3	: 3 READONLY PASSES
310	001300	000001		WRTPAS: .WORD	1	: 1 WRITEONLY PASS
311	001302	000004		RD2PAS: .WORD	4	: 4 READONLY PASSES
312	001304	000		FILE: .BYTE	0	: FILE NUMBER
313	001305	377			377	: 1'S COMPLEMENT ON FILE NUMBER
314	001306	000		BLOCK: .BYTE	0	: BLOCK NUMBER
315	001307	377			377	: 1'S COMPLEMENT OF BLOCK NUMBER
316	001310	000020		FILESZ: .WORD	16.	: NUMBER OF BLOCKS PER FILE

3200  
3201  
3202  
3203  
3204  
3205  
3206  
3207  
3208  
3209  
3210  
3211  
3212  
3213  
3214  
3215  
3216  
3217  
3218  
3219  
3220  
3221  
3222  
3223  
3224  
3225  
3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234  
3235  
3236  
3237  
3238  
3239  
3240  
3241  
3242  
3243  
3244  
3245  
3246  
3247  
3248  
3249  
3250  
3251  
3252  
3253  
3254  
3255  
3256  
3257  
3258  
3259  
3260

001312 002000  
001314 001000  
001316 002000  
001320 000400  
001322 002000  
001324 000200  
001326 002000  
001330 000100  
001332 002000  
001334 000040  
001336 000200  
001340 000400  
001342 001000  
001344 002000  
001346 000040  
001350 000200  
  
001352 001412  
001354 001422  
001356 001432  
001360 001442  
001362 001452  
001364 001462  
001366 001472  
001370 001502  
001372 001512  
001374 001522  
001376 001532  
001400 001542  
001402 001552  
001404 001562  
001406 001572  
001410 001602

////////////////////////////////////

.SBTTL            BLOCK SIZE TABLE

BLKSZ:	.WORD	1024.	:BLOCK	0
	.WORD	512.	:BLOCK	1
	.WORD	1024.	:BLOCK	2
	.WORD	256.	:BLOCK	3
	.WORD	1024.	:BLOCK	4
	.WORD	128.	:BLOCK	5
	.WORD	1024.	:BLOCK	6
	.WORD	64.	:BLOCK	7
	.WORD	1024.	:BLOCK	8
	.WORD	32.	:BLOCK	9
	.WORD	128.	:BLOCK	10
	.WORD	256.	:BLOCK	11
	.WORD	512.	:BLOCK	12
	.WORD	1024.	:BLOCK	13
	.WORD	32.	:BLOCK	14
	.WORD	128.	:BLOCK	15

////////////////////////////////////

.SBTTL            TABLE OF POINTERS TO THE DIFFERENT PATTERNS

PATS:	PAT0
	PAT1
	PAT2
	PAT3
	PAT4
	PAT5
	PAT6
	PAT7
	PAT8
	PAT9
	PAT10
	PAT11
	PAT12
	PAT13
	PAT14
	PAT15

# E03

TAII DATA RELIABILITY MAINDEC-11-DZTRAE-C  
 DZTRAE.P11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 9  
 TABLE OF POINTERS TO THE DIFFERENT PATTERNS

				: //	
		.SBTTL		DATA PATTERNS	
360					
361					
362					
363					
364					
365	001412	314	063	314	PAT0: .BYTE 314,063,314,063,146,231,146,231
366	001415	063	146	231	
367	001420	146	231		
368	001422	314	231	063	PAT1: .BYTE 314,231,063,146,314,231,063,146
369	001425	146	314	231	
370	001430	063	146		
371	001432	001	002	004	PAT2: .BYTE 001,002,004,010,020,040,100,200
372	001435	010	020	040	
373	001440	100	200		
374	001442	177	277	337	PAT3: .BYTE 177,277,337,357,367,373,375,376
375	001445	357	367	373	
376	001450	375	376		
377	001452	252	125	252	PAT4: .BYTE 252,125,252,125,252,125,252,125
378	001455	125	252	125	
379	001460	252	125		
380	001462	000	111	222	PAT5: .BYTE 000,111,222,333,044,155,266,377
381	001465	333	044	155	
382	001470	266	377		
383	001472	000	044	111	PAT6: .BYTE 000,044,111,155,222,266,333,377
384	001475	155	222	266	
385	001500	333	377		
386	001502	000	222	044	PAT7: .BYTE 000,222,044,266,111,333,155,377
387	001505	266	111	333	
388	001510	155	377		
389	001512	001	003	007	PAT8: .BYTE 001,003,007,017,037,077,177,377
390	001515	017	037	077	
391	001520	177	377		
392	001522	376	374	370	PAT9: .BYTE 376,374,370,360,340,300,200,000
393	001525	360	340	300	
394	001530	200	000		
395	001532	001	376	002	PAT10: .BYTE 001,376,002,375,004,373,010,367
396	001535	375	004	373	
397	001540	010	367		
398	001542	020	357	040	PAT11: .BYTE 020,357,040,337,100,277,200,177
399	001545	337	100	277	
400	001550	200	177		
401	001552	000	000	000	PAT12: .BYTE 000,000,000,000,000,000,000,000
402	001555	000	000	000	
403	001560	000	000		
404	001562	377	377	377	PAT13: .BYTE 377,377,377,377,377,377,377,377
405	001565	377	377	377	
406	001570	377	377		
407	001572	000	377	000	PAT14: .BYTE 000,377,000,377,000,000,377,377
408	001575	377	000	000	
409	001600	377	377		
410	001602	017	360	207	PAT15: .BYTE 017,360,207,170,303,074,341,036
411	001605	170	303	074	
412	001610	341	036		

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C  
DZTREC.P11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 10  
DATA PATTERNS

414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424

001612 000  
001613 000  
001614 001612  
001616 013464  
001620 000000

;//

.SBTTL PARAMETER BLOCK USED WITH ALL FUNCTIONS

PARMBK: .BYTE 0 ;USED FOR STATUS/ERROR  
.BYTE 0 ;DRIVE # (DRIVE A=0, B=1)  
.WORD PARMBK ;POINTS TO STATUS/ERROR BYTE  
.WORD BUFFER ;FIRST ADDRESS OF DATA BUFFER  
.WORD 0 ;USED FOR BYTE COUNT.



.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

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

001622

\$ERRTB:

\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

001622

ITEMSO: ;ITEMS 001-002

;NOTE: ALL NUMBERS WILL BE TYPED AS 6 DIGIT OCTAL NUMBERS  
; UNLESS OTHERWISE NOTED

;ITEM 1  
EM1 ;DATA ERROR  
DH1 ;PC FILE BLOCK BYTE GDDAT BDDAT GDADR BDADR  
DT1 ;\$ERRPC \$REGO \$REG2 BYTNUM \$GDDAT \$BDDAT \$GDADR \$BDADR  
DF1 ;FILE,BLOCK AND BYTE ARE TYPED IN DECIMAL

;ITEM 2  
EM2 ;SYNC ERROR  
DH2 ; EXPT'D EXPT'D RCV'D RCV'D  
;PC FILE BLOCK FILE BLOCK  
DT2 ;\$ERRPC \$REGO \$REG2 \$TMPO \$TMP2  
DF2 ;ALL NUMBERS EXCEPT \$ERRPC ARE TYPED IN DECIMAL

\*\*\*\*\*  
\*\*\*\*\*  
\*\*\*\*\*

001642

ITEMS1: ;ITEMS 101-107

EM101 ;DRIVE IS OFF-LINE  
DH101 ;PC FILE BLOCK FUNCTION  
DT101 ;\$ERRPC \$REGO \$REG2 \$TMPO  
0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
; \$TMPO WILL BE TYPED AS A FUNCTION NAME

EM102 ;DRIVE IS WRITE-LOCK  
DH101 ;PC FILE BLOCK FUNCTION  
DT101 ;\$ERRPC \$REGO \$REG2 \$TMPO  
0 ;FILE AND BLOCK WILL BE TYPED IN DECIMAL  
; \$TMPO WILL BE TYPED AS A FUNCTION NAME

481	001662	012340	EM103	:CLEAR LEADER ERROR
482	001664	012750	DH101	:PC FILE BLOCK FUNCTION
483	001666	012026	DT101	:SERRPC \$REG0 \$REG2 \$TMPO
484	001670	000000	0	:FILE AND BLOCK WILL BE TYPED IN DECIMAL
485				:\$TMPO WILL BE TYPED AS A FUNCTION NAME
486				
487	001672	012363	EM104	:FILE GAP ERROR
488	001674	012750	DH101	:PC FILE BLOCK FUNCTION
489	001676	012026	DT101	:SERRPC \$REG0 \$REG2 \$TMPO
490	001700	000000	0	:FILE AND BLOCK WILL BE TYPED IN DECIMAL
491				:\$TMPO WILL BE TYPED AS A FUNCTION NAME
492				
493	001702	012402	EM105	:TIMING ERROR
494	001704	012750	DH101	:PC FILE BLOCK FUNCTION
495	001706	012026	DT101	:SERRPC \$REG0 \$REG2 \$TMPO
496	001710	000000	0	:FILE AND BLOCK WILL BE TYPED IN DECIMAL
497				:\$TMPO WILL BE TYPED AS A FUNCTION NAME
498				
499	001712	012417	EM106	:BLOCK CHECK ERROR
500	001714	012750	DH101	:PC FILE BLOCK FUNCTION
501	001716	012026	DT101	:SERRPC \$REG0 \$REG2 \$TMPO
502	001720	000000	0	:FILE AND BLOCK WILL BE TYPED IN DECIMAL
503				:\$TMPO WILL BE TYPED AS A FUNCTION NAME
504				
505	001722	012441	EM107	:UNKNOWN INTERRUPT
506	001724	012750	DH101	:PC FILE BLOCK FUNCTION
507	001726	012026	DT101	:SERRPC \$REG0 \$REG2 \$TMPO
508	001730	000000	0	:FILE AND BLOCK WILL BE TYPED IN DECIMAL
509				:\$TMPO WILL BE TYPED AS A FUNCTION NAME
510				
511				:*****
512				:*****
513				:*****
514				:*****
515	001732		ITEMS2:	:ITEMS 201-202
516				
517	001732	012463	EM201	:TA11 FAILED TO RESPOND
518	001734	013011	DH201	:PC TACS
519	001736	012040	DT201	:SERRPC TACS
520	001740	000000	0	:BOTH NUMBERS ARE TYPED AS OCTAL NUMBERS
521				
522	001742	012512	EM202	:NO DRIVES AVAILABLE
523	001744	013026	DH202	:PC
524	001746	012046	DT202	:SERRPC
525	001750	000000	0	:
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

```

:////////////////////
:////////////////////
:*****
:BEGIN1 IS FOR NORMAL START
:BEGIN2 IS FOR DRIVE SELECTION
:BEGIN3 IS FOR DRIVE & ADDRESS SELECTION
:;*****
537 001752 005005 BEGIN1: CLR R5 ;NORMAL START
538 001754 012737 041101 001252 MOV #AB, @DRVKEY
539 001762 122737 000005 000041 CMPB #5, @41 ;CASSETTE DDP?
540 001770 001012 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
541 001772 022737 000260 001244 CMP #260, @TAVEC ;STANDARD VECTOR?
542 002000 001006 BNE BGNCMN ;GO BEGIN COMMON CODE IF NO
543 002002 000403 BR BEGIN3 ;GET DRIVES AND ADDRESSES
544 002004 012705 000001 BEGIN2: MOV #1, R5 ;ASK FOR DRIVES FLAG
545 002010 000402 BR BGNCMN ;BEGIN COMMON CODE
546 002012 012705 000002 BEGIN3: MOV #2, R5 ;ASK FOR DRIVES AND ADDRESSES
547 002016 005067 175754 BGNCMN: CLR PS
548 .SBTTL INITIALIZE THE COMMON TAGS
549 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
550 002022 012706 001100 MOV $CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
551 002026 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
552 002030 022706 001140 CMP $SWR, R6 ;;DONE?
553 002034 001374 BNE -6 ;;LOOP BACK IF NO
554 002036 012706 001100 MOV #STACK, SP ;;SETUP THE STACK POINTER
555 ;;INITIALIZE A FEW VECTORS
556 002042 012737 004760 000020 MOV $$SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
557 002050 012737 000340 000022 MOV #340, @IOTVEC+2 ;;LEVEL 7
558 002056 012737 005060 000030 MOV $ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
559 002064 012737 000340 000032 MOV #340, @EMTVEC+2 ;;LEVEL 7
560 002072 012737 011522 000034 MOV $STRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
561 002100 012737 000340 000036 MOV #340, @TRAPVEC+2;LEVEL 7
562 002106 012737 011606 000024 MOV $SPWRDN, @PWRVEC ;;POWER FAILURE VECTOR
563 002114 012737 000340 000026 MOV #340, @PWRVEC+2 ;;LEVEL 7
564 002122 016767 002314 002304 MOV $ENDCT, $EOPCT ;;SETUP END-OF-PROGRAM COUNTER
565 002130 012767 002130 176750 MOV #, $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
566 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
567 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
568 002136 013746 000004 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
569 002142 012737 002176 000004 MOV #64$, @ERRVEC ;;SET UP ERROR VECTOR
570 002150 012767 177570 176762 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
571 002156 012767 177570 176756 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
572 002164 022777 177777 176746 CMP #-1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
573 002172 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
574 ;;AND THE HARDWARE SWR IS NOT = -1
575 002174 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
576 002176 012716 002204 64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN
577 002202 000002 RTI
578 002204 012767 000176 176726 65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
579 002212 012767 000174 176722 MOV #DISPREG, DISPLAY
580 002220 012637 000004 66$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
581 .SBTTL TYPE PROGRAM NAME
582

```

```

583                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
584 002224 005227 177777                INC      #-1                ;;FIRST TIME?
585 002230 001036                        BNE      67$                ;;BRANCH IF NO
586 002232 022737 004726 000042        CMP      #$SENDAD,2#42    ;;ACT-11?
587 002240 001432                        BEQ      67$                ;;BRANCH IF YES
588 002242 104401 002300                TYPE     68$                ;;TYPE ASCIZ STRING
589                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
590 002246 005737 000042                TST     2#42                ;;ARE WE RUNNING UNDER XXDP/ACT?
591 002252 001006                        BNE      69$                ;;BRANCH IF YES
592 002254 026727 176660 000176        CMP      SWR,#SWREG        ;;SOFTWARE SWITCH REG SELECTED?
593 002262 001005                        BNE      70$                ;;BRANCH IF NO
594 002264 104405                        GTSWR                                     ;;GET SOFT-SWR SETTINGS
595 002266 000403                        BR      70$
596 002270 112767 000001 176636 69$:  MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
597 002276                                     70$:
598 002276 000413                        BR      67$                ;;GET OVER THE ASCIZ
599                                     ;;68$: .ASCIZ <CRLF>/MAINDEC-11-DZTAE-C/<CRLF>
600 002326                                     67$:
601
602                                     ;;*****
603                                     ;;*****
604
605                                     ;THE CONTENTS OF R5 DETERMINES WHAT WILL BE DONE
606
607                                     R5=2   ASK FOR DRIVE(S) AND ADDRESSES (TACS AND VECTOR)
608                                     R5=1   ASK FOR DRIVE(S)
609                                     R5=0   DON'T ASK FOR ANYTHING
610
611                                     ;;*****
612 002326 010504  BEGINX: MOV     R5,R4                ;COPY R5
613 002330 005305                        DEC     R5                ;ASK FOR DRIVES?
614 002332 002406                        BLT    CHKADR            ;BR IF NO
615 002334 004737 007620                JSR    PC,2#ASKDRV        ;GO GET DRIVES TO BE TESTED
616 002340 005305                        DEC     R5                ;ASK FOR ADDRESSES?
617 002342 002402                        BLT    CHKADR            ;BR IF NO
618 002344 004737 007730                JSR    PC,2#ASKADR        ;GO GET TA11 ADDRESSES
619
620                                     ;;*****
621                                     ;;*****
622
623                                     ;CHECK THAT "TACS" WILL RESPOND TO ADDRESSING
624
625                                     I.   TIMEOUT OCCURRED
626                                     A.  TYPE ERROR MESSAGE
627                                     B.  EXAMINE R4
628                                     1.  R4>0 GOTO BEGINX
629                                     2.  R4=0 EXAMINE (42)
630                                     A.  (42)=0 GOTO BEGINX
631                                     B.  (42)>0 GOTO $ENDAD
632
633                                     II.  TIMEOUT DIDN'T OCCUR
634                                     A.  CONTINUE
635
636 002350 012737 002366 000004  CHKADR: MOV    #1$,2#ERRVEC    ;IN CASE OF TIMEOUTS
637 002356 005000                        CLR    R0                ;USE AS A SWITCH
638 002360 005777 176654                TST    2TACSL            ;SEE IF TA11 RESPONDS

```

K03

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C MACY11 27(1006) 17-MAR-77 14:55 PAGE 15  
 DZTAECP11 17-MAR-77 14:52 GET VALUE FOR SOFTWARE SWITCH REGISTER

639	002364	000402				BR	2\$		;BR IF NO TIMEOUT
640	002366	005200			1\$:	INC	RO		;COME HERE ON TIMEOUT
641	002370	022626				CMP	(SP)+,(SP)+		;CLEANUP THE STACK
642	002372	012737	000006	000004	2\$:	MOV	#ERRVEC+2,@#ERRVEC		;RESTORE TIMEOUT VECTOR
643	002400	005700				TST	RO		;DID A TIMEOUT OCCUR?
644	002402	001412				BEQ	3\$		;BR IF NO
645	002404	104201				ERROR	201		;TA11 FAILED TO RESPOND
646	002406	012705	000002			MOV	#2,R5		;DRIVES & ADDRESSES
647	002412	005704				TST	R4		;OPERATOR INPUTS?
648	002414	001344				BNE	BEGINX		;BR IF YES
649	002416	013700	000042			MOV	@#42,RO		;GET MONITOR RETURN ADDRESS
650	002422	001741				BEQ	BEGINX		;BR IF NO MONITOR
651	002424	000137	004726			JMP	@#SENDAD		;GO TO END
652	002430	012777	007056	176606	3\$:	MOV	#CASINT,@TAVEC		;SETUP CASSETTE INTERRUPT VECTOR
653	002436	005077	176604			CLR	@TAVEC+2		

```

654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677 002442 012700 001252
678 002446 004737 010174
679 002452 000410
680 002454 116010 000001
681 002460 001412
682 002462 004737 010174
683 002466 000407
684 002470 005010
685 002472 000405
686 002474 005200
687 002476 004737 010174
688 002502 000401
689 002504 105010
690 002506 012700 001252
691 002512 010037 001256
692 002516 121060 000001
693 002522 001002
694 002524 105060 000001
695 002530 005710
696 002532 001424
697 002534 112037 001613
698 002540 104401 001203
699 002544 106237 001613
700 002550 004737 007564
701 002554 111037 001613
702 002560 001406
703 002562 104401 013220
704 002566 106237 001613
705 002572 004737 007564
706 002576 104401 013225
707 002602 000412
708 002604 104202
709 002606 012705 000002
    
```

```

*****
*****
;MAKE SURE THE DRIVES IN THE DRIVE TABLE CAN BE TESTED
;I. DESIRED DRIVES CAN NOT BE TESTED
;A. TYPE ERROR MESSAGE
;B. EXAMINE R4
;1. R4>0 GOTO BEGINX
;2. R4=0 EXAMINE (42)
;A. (42)=0 GOTO BEGINX
;B. (42)>0 GOTO $ENDAD
;II. BOTH DRIVES IN THE TABLE BUT ONLY ONE OF THEM CAN BE TESTED
;A. CLEAR BAD DRIVE FROM THE DRIVE TABLE
;B. TYPE THE DRIVE TO BE TESTED
;C. CONTINUE IN PROGRAM
;III. DESIRED DRIVE(S) CAN BE TESTED
;A. TYPE THE DRIVE(S) TO BE TESTED
;B. CONTINUE IN PROGRAM
*****
CHKDRV: MOV #DRVKEY,R0 ;PICKUP ADDRESS OF ASCII DRIVE KEY
;JSR PC,@#EXAM ;GO EXAMINE FIRST DRIVE
;BR 1$ ;OK TO TEST---GO CHECK NEXT
;MOVB 1(R0),(R0) ;REPLACE 1ST WITH 2ND
;BEQ 2$ ;BR IF NO 2ND DRIVE SELECTED
;JSR PC,@#EXAM ;GO EXAMINE DRIVE
;BR 2$ ;OK TO TEST
;CLR (R0) ;CLEAR DRIVE CODES
;BR 2$
1$: INC R0 ;POINT TO 2ND
;JSR PC,@#EXAM ;GO EXAMINE DRIVE
;BR 2$ ;OK TO TEST
;CLRB (R0) ;CLEAR 2ND
;2$: MOV #DRVKEY,R0 ;RESET ADDRESS POINTERS
;MOV R0,@#DRVPNT
;CMPB (R0),1(R0) ;1ST = 2ND?
;BNE 3$ ;BR IF NO
;CLRB 1(R0) ;YES---CLEAR 2ND
;3$: TST (R0) ;ANY DRIVES?
;BEQ 5$ ;BR IF NO
;MOVB (R0)+,@#PARMBK+1 ;SETUP TO TYPE THIS DRIVE
;TYPE ,SCRLF ;TYPE "CR" & "LF"
;ASRB @#PARMBK+1 ;ADJUST FOR TYPING
;JSR PC,@#TPDRV ;GO TYPE DRIVE
;MOVB (R0),@#PARMBK+1 ;GET 2ND
;BEQ 4$ ;BR IF NONE
;TYPE ,MSG9 ;"AND"
;ASRB @#PARMBK+1 ;ADJUST FOR TYPING
;JSR PC,@#TPDRV ;GO TYPE THE 2ND DRIVE
;4$: TYPE ,MSG10 ;"WILL BE TESTED"<CRLF>
;BR $START
;5$: ERROR 202 ;NO DRIVES AVAILABLE
;MOV #2,R5 ;DRIVES & ADDRESS
    
```

M03

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C MACY11 27(1006) 17-MAR-77 14:55 PAGE 17  
DZTREC.P11 17-MAR-77 14:52 GET VALUE FOR SOFTWARE SWITCH REGISTER

710	002612	005704		TST	R4	: OPERATOR INPUTS?
711	002614	001244		BNE	BEGINX	: BR IF YES
712	002616	013700	000042	MOV	@#42, R0	: GET MONITOR RETURN ADDRESS
713	002622	001641		BEQ	BEGINX	: NO MONITOR
714	002624	000137	004726	JMP	@#SENDAD	: GO TO END

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

002630 012700 001114  
002634 012701 000005  
002640 004737 002764  
002644 012700 001206  
002650 012701 000014  
002654 004737 002764  
002660 013701 001256  
002664 112100  
002666 006200  
002670 042700 177776  
002674 110037 001613  
002700 105711  
002702 001002  
002704 012701 001252  
002710 010137 001256  
002714 012737 000011 001236  
002722 012737 000017 001234  
002730 013737 001260 001224  
002736 013737 001262 001226  
002744 012737 001274 001270  
002752 005037 001272  
002756 005037 001266  
002762 000404  
002764 005020  
002766 005301  
002770 003375  
002772 000207

```

////////////////////////////////////
////////////////////////////////////
*****
*****
1. CLEAR THE VARIABLE STORAGE AREA
2. SETUP FOR THE DRIVE THAT IS TO BE TESTED
3. SETUP "LASTFL" AND "LASTBK" INCASE OPERATOR SELECTS "READONLY" PASS
4. SETUP THE MAX. NUMBER OF RETRYS FOR READS AND WRITES
5. SETUP TO START WITH A "FORMAT" PASS
6. CLEAR THE PASS COUNTER
*****
START:  MOV    #$SITEMB,RO           ;CLEAR VARIABLE STORAGE AREA
        MOV    #$BDDAT-$SITEMB/2,R1
        JSR    PC,@#CLEAR           ;CLEAR AREA
        MOV    #SOFTNM,RO
        MOV    #LASTFL-SOFTNM/2,R1
        JSR    PC,@#CLEAR
        MOV    @#DRVPNT,R1          ;GET DRIVE POINTER
        MOVB   (R1)+,RO             ;SETUP THE DRIVE
        ASR    RO
        BIC    #1CBIT00,RO
        MOVB   RO,@#PARMBK+1
        TSTB   (R1)                ;END OF DRIVE TABLE?
        BNE    2$                  ;BR IF NO
        MOV    #DRVKEY,R1           ;RESET DRIVE TABLE POINTER
        MOV    R1,@#DRVPNT         ;RESTORE THE DRIVE POINTER
        MOV    #9,@#LASTFL         ;SET "LAST FILE" & "LAST BLOCK" INCASE
        MOV    #15,@#LASTBK        ;OPERATOR SELECTS "READONLY" PASS
        MOV    @#MAXRDS,@#RDTRYS   ;SETUP MAX. # OF REREAD TRYS
        MOV    @#MAXERR,@#WRTRYS   ;SETUP MAX. # OF REWRITE TRYS
        MOV    #FORPAS,@#PSCNTL    ;DO A FORMAT PASS
        CLR    @#PSCNTL+2
        CLR    @#PASCNT
        BR     LOOPER              ;CLEAR THE PASS COUNT
        CLR    (RO)+               ;GET AROUND CLEAR
        DEC    R1
        BGT    CLEAR
        RTS    PC
CLEAR:

```



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

```

*****
*****
I. CHECK SWR<7>
A. SWR<7>=1 PERFORM AS PER SWR<1:0>
  1. 00=FORMAT PASS
  2. 01=READ ONLY PASS
  3. 10=WRITE ONLY PASS
  4. 11=READ ONLY PASS

B. SWR<7>=0
  1. UPDATE PASCNT
  2. PERFORM AS PER (PSCNTL+2)
    A. FORMAT
    B. READONLY
    C. WRITEONLY
    D. READONLY
*****
LOOPER: MOVB @SWR,@$STMP0 ;IF SWR<7>=1 ; DO PER SWR<1:0>
        BICB @C<BIT07!BIT01!BIT00>,@$STMP0
        BMI 3$ ;BR IF SWR<7>=1
        CMP @PASCNT,@PSCNTL ;IS THE COUNTER AT MAX.?
        BLT 2$ ;BR IF NO
        CLR @PASCNT ;RESET THE COUNTER
        ADD #2,@PSCNTL ;MOVE TO THE NEXT PASS TYPE
        INC @PSCNTL+2
        CMP @PSCNTL,@RD2PAS+2 ;TIME TO RESET PASS TYPE?
        BNE 1$ ;BR IF NO
        MOV #FORPAS,@PSCNTL ;YES---RESET THE PASS CONTROL WORDS
        CLR @PSCNTL+2
        BR 1$ ;GO CHECK THE COUNT
        INC @PASCNT ;COUNT THIS PASS
        MOV @PSCNTL+2,@$STMP0 ;PICKUP THE TYPE OF PASS TO DO NEXT
3$:
        CLR -(SP)
        MOVB @$STMP0,(SP) ;PICKUP THE DISPATCH INDEX
        ASLB (SP) ;POSITION THE INDEX
        ASLB (SP) ; BEFORE USING IT
        ADD (SP)+,PC ;GO TO THE ROUTINE
        JMP @FORMAT
        JMP @READONLY
        JMP @WRITEONLY
        JMP @READONLY
//
//
*****
*****

```



```

959 003214 000137 004412          JMP      @#$EOP          ;GO TO END OF PROGRAM
960  : ////////////////////////////////////////////////////
961  : "WRITE-FILL-GAP"
962  : ////////////////////////////////////////////////////
963  003220          2$:
964  003220 012700 001612          MOV      #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
965  003224 004737 006722          JSR      PC,@#WFG        ;GO TO WFG
966  003230 105737 001612          3$: TSTB     @#PARMBK      ;WAIT ON FLAG
967  003234 001775          BEQ      3$
968  003236 100006          BPL      4$              ;BR IF NO ERROR
969  003240 004037 005536          JSR      RO,@#CSRERR     ;GO TO CSR ERROR CHECK
970  003244 000137 004412          JMP      @#$EOP          ;GO TO END OF PROGRAM
971  003250 000137 004364          JMP      @#CKEOTS
972  : ////////////////////////////////////////////////////
973  : "WRITE" A BLOCK OF DATA
974  : ////////////////////////////////////////////////////
975  003254 012737 003262 001106 4$: MOV      #$$,@#SLPADR     ;SETUP SCOPE LOOP ADDRESS
976  003262 004737 006072          5$: JSR      PC,@#SETUPW    ;GO SETUP FOR A WRITE
977  003266 012700 001612          MOV      #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
978  003272 004737 006726          JSR      PC,@#WRITE      ;GO TO WRITE
979  003276 105737 001612          6$: TSTB     @#PARMBK      ;WAIT ON FLAG
980  003302 001775          BEQ      6$
981  003304 100006          BPL      7$              ;BR IF NO ERROR
982  003306 004037 005536          JSR      RO,@#CSRERR     ;GO TO CSR ERROR CHECK
983  003312 000137 004412          JMP      @#$EOP          ;GO TO END OF PROGRAM
984  003316 000137 004364          JMP      @#CKEOTS
985  : ////////////////////////////////////////////////////
986  : "BACK-SPACE-BLOCK-GAP"
987  : ////////////////////////////////////////////////////
988  003322          7$:
989  003322 012700 001612          MOV      #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
990  003326 004737 006742          JSR      PC,@#BSBG       ;GO TO BSBG
991  003332 105737 001612          8$: TSTB     @#PARMBK      ;WAIT ON FLAG
992  003336 001775          BEQ      8$
993  003340 100004          BPL      9$              ;BR IF NO ERROR
994  003342 004037 005536          JSR      RO,@#CSRERR     ;GO TO CSR ERROR CHECK
995  003346 000137 004412          JMP      @#$EOP          ;GO TO END OF PROGRAM
996  : ////////////////////////////////////////////////////
997  : "READ" A BLOCK OF DATA
998  : ////////////////////////////////////////////////////
999  003352 004737 006220          9$: JSR      PC,@#SETUPR    ;GO SETUP FOR A READ
1000 003356 012700 001612          MOV      #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
1001 003362 004737 006732          JSR      PC,@#READ       ;GO TO READ
1002 003366 105737 001612          10$: TSTB    @#PARMBK      ;WAIT ON FLAG
1003 003372 001775          BEQ      10$
1004 003374 100004          BPL      11$             ;BR IF NO ERROR
1005 003376 004037 005536          JSR      RO,@#CSRERR     ;GO TO CSR ERROR CHECK
1006 003402 000137 004412          JMP      @#$EOP
1007 003406 000004          11$: SCOPE
1008  : ////////////////////////////////////////////////////
1009  : CHECK FOR SYNC ERROR
1010  : ////////////////////////////////////////////////////
1011 003410 004737 006254          JSR      PC,@#SYNCK      ;SYNC ERROR?
1012 003414 000403          BR       12$             ;RETURN HERE IF NO
1013 003416 104002          ERROR   2                ;SYNC ERROR
1014 003420 000137 004412          JMP      @#$EOP

```

```

915
916
917
918 003424 004737 006356
919 003430 000430
920 003432 104001
921 003434 004037 006550
922 003440 000730
923 003442 004037 006606
924 003446 000402
925 003450 000137 004412
926
927
928
929 003454 005337 001226
930 003460 002414
931 003462 012700 001612
932 003466 004737 006742
933 003472 105737 001612
934 003476 001775
935 003500 100265
936 003502 004037 005536
937 003506 000137 004412
938 003512 013737 001260 001224
939 003520 013737 001262 001226
940
941
942
943 003526 062737
944 003530 001 377
945 003532 001306
946 003534 123737 001310 001306
947 003542 003244
948 003544 062737
949 003546 001 377
950 003550 001304
951 003552 012737
952 003554 000 377
953 003556 001306
954 003560 000617

: ////////////////////////////////////////////////////
: CHECK FOR DATA ERROR IF SOFT REREAD
: ////////////////////////////////////////////////////
12$: JSR PC,@DATCMP ;CHECK THE DATA
BR 15$ ;RETURN HERE IF DATA IS GOOD
ERROR 1 ;DATA ERROR
JSR RO,@CNTSFT ;COUNT SOFT ERROR
BR 7$ ;GO REREAD
JSR RO,@CNTHRD ;COUNT HARD ERROR
BR 13$ ;REWRITE
JMP @SEOP ;TO MANY HARD ERRORS
: ////////////////////////////////////////////////////
: IF HARD ERROR REWRITE
: ////////////////////////////////////////////////////
13$: DEC @WRTRYS ;TRY TO REWRITE THIS BLOCK?
BLT 15$ ;BR IF NO
MOV #PARMBK,RO ;RO=1ST ADDRESS OF PARAMETER BLOCK
JSR PC,@BSBG ;GO TO BSBG
14$: TSTB @PARMBK ;WAIT ON FLAG
BEQ 14$
BPL 4$ ;GO START A WRITE IF NO ERROR
JSR RO,@CSRERR ;GO TO CSR ERROR CHECK
JMP @SEOP
15$: MOV @MAXRDS,@RDTRYS ;RESET THE REREAD COUNT
MOV @MAXERR,@WRTRYS ;RESET THE REWRITE COUNT
: ////////////////////////////////////////////////////
: UPDATE BLOCK # & FILE #
: ////////////////////////////////////////////////////
ADD (PC)+,@(PC)+ ;INCREMENT THE BLOCK NUMBER
.BYTE 1,-1
.WORD BLOCK
CMPB @FILESZ,@BLOCK ;TIME FOR A FILE GAP?
BGT 4$ ;BR IF NO--GO START A WRITE
ADD (PC)+,@(PC)+ ;INCREMENT THE FILE NUMBER
.BYTE 1,-1
.WORD FILE
MOV (PC)+,@(PC)+ ;INITIALIZE THE BLOCK NUMBER
.BYTE 0,377 ;SET BYTE 0 TO 0 AND BYTE 1
.WORD BLOCK ;TO THE 1'S COMP. OF BYTE 0
BR 2$ ;GO WRITE FILE GAP

```

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  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010

```

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

```

.SBTTL "READ ONLY" ROUTINE

```

:*****
:*****

```

: THIS ROUTINE PERFORMS THE FOLLOWING SEQUENCE

1. REWIND TO BOT
2. READ BLOCK OF DATA
3. CHECK FOR SYNC ERROR
4. CHECK FOR DATA ERROR
  - A. DATA ERROR OCCURRED
    - 1.) (RDTRYS)<(MAXRDS)
      - A.) BACK BLOCK GAP
      - B.) GOTO 2.
    - 2.) (RDTRYS)=(MAXRDS)
      - A.) (WRTRYS)<(MAXERR) GOTO 5.
      - B.) (WRTRYS)=(MAXERR) GOTO \$EOP
  - B. NO DATA ERROR --- GOTO 5.
5. UPDATE THE BLOCK NUMBER
6. END OF FILE?
  - A. NO --- GOTO 2.
  - B. YES
    - 1.) UPDATE FILE NUMBER
    - 2.) RESET BLOCK NUMBER TO 0
    - 3.) SPACE FORWARD FILE GAP
    - 4.) GOTO 2.

```

:*****
:*****

```

```

READONLY:
MOV    #READONLY, @#$LPADR      ; SETUP SCOPE LOOP ADDRESS
TYPE   MSG12                    ; "*** READ ***"
JSR    PC, @#TPDRV             ; GO TYPE DRIVE TO BE TESTED
TYPE   $CRLF
MOV    (PC)+, @ (PC)+          ; INITIALIZE THE BLOCK NUMBER
.BYTE  0, 377                  ; SET BYTE 0 TO 0 AND BYTE 1
.WORD  BLOCK                    ; TO THE 1'S COMP. OF BYTE 0
MOV    (PC)+, @ (PC)+          ; INITIALIZE THE FILE NUMBER
.BYTE  0, 377                  ; SET BYTE 0 TO 0 AND BYTE 1
.WORD  FILE                     ; TO THE 1'S COMP. OF BYTE 0

```

```

////////////////////////////////////
"REWIND" TO "BOT"
////////////////////////////////////

```

```

MOV    #PARMBK, RO              ; RO=1ST ADDRESS OF PARAMETER BLOCK
JSR    PC, @#REWIND            ; GO TO REWIND
1$:    TSTB @#PARMBK            ; WAIT ON FLAG
BEQ    1$
BPL    2$                      ; BR IF NO ERROR
JSR    RO, @#CSRERR            ; GO TO CSR ERROR CHECK
JMP    @#$EOP                  ; GO TO END-OF-PROGRAM

```

```

:*****
:*****

```



```

1067
1068
1069 004054
1070 004054 062737
1071 004056 001 377
1072 004060 001306
1073 004062 123737 001310 001306
1074 004070 003267
1075 004072 062737
1076 004074 001 377
1077 004076 001304
1078 004100 012737
1079 004102 000 377
1080 004104 001306
1081
1082
1083
1084 004106 012700 001612
1085 004112 004737 006746
1086 004116 105737 001612
1087 004122 001775
1088 004124 100251
1089 004126 004037 005536
1090 004132 000137 004412
1091

```

```

: UPDATE BLOCK # & FILE #
: ////////////////////////////////////////////////////
10s: ADD (PC)+, @ (PC)+ ; INCREMENT THE BLOCK NUMBER
      .BYTE 1, -1
      .WORD BLOCK
      CMPB @#FILESZ, @#BLOCK ; TIME FOR A FILE GAP?
      BGT 2$ ; BR IF NO
      ADD (PC)+, @ (PC)+ ; INCREMENT THE FILE NUMBER
      .BYTE 1, -1
      .WORD FILE
      MOV (PC)+, @ (PC)+ ; INITIALIZE THE BLOCK NUMBER
      .BYTE 0, 377 ; SET BYTE 0 TO 0 AND BYTE 1
      .WORD BLOCK ; TO THE 1'S COMP. OF BYTE 0
: ////////////////////////////////////////////////////
: "SPACE-FORWARD-FILE-GAP"
: ////////////////////////////////////////////////////
11s: MOV #PARMBK, R0 ; R0=1ST ADDRESS OF PARAMETER BLOCK
      JSR PC, @#SFFG ; GO TO SFFG
      TSTB @#PARMBK ; WAIT ON FLAG
      BEQ 11$
      BPL 2$ ; GO READ
      JSR R0, @#CSRERR ; GO CHECK CSR ERROR
      JMP @#$EOP ; GO TO END-OF-PROGRAM

```





```

1148 004250 000137 004412      JMP      @#SEOP      ;GO TO END-OF-PROGRAM
1149 004254 000137 004364      JMP      @#CKEOTS
1150      ; ////////////////////////////////////////////////////
1151      ; "WRITE" A BLOCK OF DATA
1152      ; ////////////////////////////////////////////////////
1153 004260 004737 006072      4$: JSR      PC,@#SETUPW      ;GO SETUP FOR "WRITE"
1154 004264 012737 004272 001106  MOV      #5$,@#SLPADR      ;SETUP SCOPE LOOP ADDRESS
1155 004272
1156 004272 012700 001612      5$: MOV      #PARMBK,RO      ;RO=1ST ADDRESS OF PARAMETER BLOCK
1157 004276 004737 006726      JSR      PC,@#WRITE      ;GO TO WRITE
1158 004302 105737 001612      6$: TSTB   @#PARMBK      ;WAIT ON FLAG
1159 004306 001775
1160 004310 100006
1161 004312 004037 005536      JSR      RO,@#CSRERR      ;BR IF NO ERROR
1162 004316 000137 004412      JMP      @#SEOP      ;GO TO CSR ERROR CHECK
1163 004322 000137 004364      JMP      @#CKEOTS      ;GO TO END-OF-PROGRAM
1164 004326 000004      7$: SCOPE
1165      ; ////////////////////////////////////////////////////
1166      ; UPDATE BLOCK # & FILE #
1167      ; ////////////////////////////////////////////////////
1168 004330 062737      ADD      (PC)+,@(PC)+      ;INCREMENT THE BLOCK NUMBER
1169 004332      001      377      .BYTE   1,-1
1170 004334 001306      .WORD   BLOCK
1171 004336 123737 001310 001306      CMPB   @#FILESZ,@#BLOCK      ;TIME FOR A FILE GAP?
1172 004344 003345      BGT     4$      ;BR IF NO
1173 004346 062737      ADD      (PC)+,@(PC)+      ;INCREMENT THE FILE NUMBER
1174 004350      001      377      .BYTE   1,-1
1175 004352 001304      .WORD   FILE
1176 004354 012737      MOV      (PC)+,@(PC)+      ;INITIALIZE THE BLOCK NUMBER
1177 004356      000      377      .BYTE   0,377      ;SET BYTE 0 TO 0 AND BYTE 1
1178 004360 001306      .WORD   BLOCK      ;TO THE 1'S COMP. OF BYTE 0
1179 004362 000720      BR      2$      ;START A FILE GAP

```

1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206

////////////////////////////////////  
////////////////////////////////////

.SBTTL CHECK "EOTS"

\*\*\*\*\*  
\*\*\*\*\*

: WHEN A "FORMAT" OR "WRITEONLY" PASS HITS "EOT" OR A  
: "READONLY" PASS READS THE LAST BLOCK OF THE LAST FILE  
: THEY WILL COME HERE

- 1. (EOTS) IS CHECKED
  - A. (EOTS)=(MAXEOT) GOTO \$EOP
  - B. (EOTS)<(MAXEOT) GOTO 2
- 2. SWR<8> = 1?
  - A. YES -- GOTO \$EOP
  - B. NO -- GOTO LOOPER

\*\*\*\*\*

004364	000004		
004366	023737	001212	001264
004374	002006		
004376	032777	000400	174534
004404	001002		
004406	000137	002774	

```

CKEOTS: SCOPE
CMP      @#EOTS,@#MAXEOT      ;MAX. EOTS OCCURRED?
BGE      $EOP                 ;BR IF YES
BIT      #SW08,@SWR          ;IF SWR<08> = 1
BNE      $EOP                 ;GOTO END-OF-TEST
JMP      @#LOOPER            ;NO--LOOP

```

```

1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217 004412
1218 004412 000004
1219 004414 005067 174462
1220 004420 005267 174454
1221 004424 042767 100000 174446
1222 004432 005327
1223 004434 000001
1224 004436 003137
1225 004440 012737
1226 004442 000001
1227 004444 004434
1228 004446 104401 004745
1229 004452 104401 004742
1230 004456 013700 000042
1231 004462 001525
1232 004464 000004
1233
1234
1235
1236 004466 012700 001612
1237 004472 004737 006756
1238 004476 104401 013031
1239
1240 004502 104401 013056
1241 004506 013746 001206
1242 004512 004737 007366
1243 004516 004737 007422
1244
1245 004522 104401 013074
1246 004526 013746 001210
1247 004532 004737 007366
1248 004536 004737 007422
1249
1250 004542 104401 013112
1251 004546 012746 001214
1252 004552 004737 007172
1253 004556 004737 007422
1254
1255 004562 104401 013127
1256 004566 012746 001220
1257 004572 004737 007172
1258 004576 004737 007422
1259
1260 004602 104401 013147
1261 004606 013746 001212
1262 004612 004737 007366

```

```

.SBTTL END OF PASS ROUTINE
;*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*TYPE "END PASS"
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO START
;*IF IT IS DESIRED TO HAVE A BELL INDICATE THE "END OF PASS" LOCATION
;*$ENDMG CAN BE CHANGED TO 7.

$EOP:
SCOPE
CLR $TSTNM ;:ZERO THE TEST NUMBER
INC $PASS ;:INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;:LOOP?
$EOPCT: .WORD 1
BGT $DOAGN ;:YES
MOV (PC)+,@(PC)+ ;:RESTORE COUNTER
$ENDCT: .WORD 1
$EOPCT
TYPE , $ENDMG ;:TYPE "END PASS"
TYPE , $ENULL ;:TYPE A NULL CHARACTER
$GET42: MOV ,@#42,RO ;:GET MONITOR ADDRESS
BEQ $DOAGN ;:BRANCH IF NO MONITOR
SCOPE

;////////////////////////////////////
; TYPE END-OF-TEST STATISTICS
;////////////////////////////////////
MOV #PARMBK,RO ;:RO=PARAMETER BLOCK ADDRESS
JSR PC,@#REWIND ;:START A REWIND
TYPE ,MSG1 ;:<CRLF>"*** END-OF-TEST ***"

TYPE MSG2 ;:<CRLF>"SOFT ERRORS="
MOV ,@#SOFTNM,-(SP) ;:PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,@#$SB2D ;:CHANGE IT TO DECIMAL ASCIZ
JSR PC,@#$SUPRS ;:TYPE WITHOUT LEADING ZEROS

TYPE MSG3 ;:<CRLF>"HARD ERRORS="
MOV ,@#HARDNM,-(SP) ;:PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,@#$SB2D ;:CHANGE IT TO DECIMAL ASCIZ
JSR PC,@#$SUPRS ;:TYPE WITHOUT LEADING ZEROS

TYPE MSG4 ;:<CRLF>"BYTES READ="
MOV ,@#RBYTTL,-(SP) ;:GET ADDRESS OF DOUBLE PRECISION BINARY #
JSR PC,@#$DB2D ;:CHANGE IT TO DECIMAL ASCIZ
JSR PC,@#$SUPRS ;:TYPE IT WITHOUT LEADING ZERO

TYPE MSG5 ;:<CRLF>"BYTES WRITTEN="
MOV ,@#WBYTTL,-(SP) ;:GET ADDRESS OF DOUBLE PRECISION BINARY #
JSR PC,@#$DB2D ;:CHANGE IT TO DECIMAL ASCIZ
JSR PC,@#$SUPRS ;:TYPE IT WITHOUT LEADING ZERO

TYPE MSG6 ;:<CRLF>"TAPE PASSES="
MOV ,@#EOTS,-(SP) ;:PICKUP SINGLE PRECISION BINARY NUMBER
JSR PC,@#$SB2D ;:CHANGE IT TO DECIMAL ASCIZ

```

```

1263 004616 004737 007422 JSR PC, @#$$SUPRS ;TYPE WITHOUT LEADING ZEROS
1264
1265 004622 104401 013165 TYPE MSG7 ;<CRLF>"FILES/PASS="
1266 004626 013700 001236 MOV @#LASTFL, RO ;PICKUP THE LAST FILE NUMBER
1267 004632 105200 INCB RO ;ADD 1 TO MAKE IT # OF FILES
1268 004634 010046 MOV RO, -(SP) ;PUT IT ON THE STACK
1269 004636 004737 007366 JSR PC, @#$$SB2D ;CHANGE IT TO DECIMAL ASCIZ
1270 004642 004737 007422 JSR PC, @#$$SUPRS ;TYPE IT WITHOUT LEADING ZEROS
1271
1272 004646 104401 013202 TYPE MSG8 ;<CRLF>"BLOCKS/FILE="
1273 004652 013746 001310 MOV @#FILESZ, -(SP) ;PICKUP SINGLE PRECISION BINARY NUMBER
1274 004656 004737 007366 JSR PC, @#$$SB2D ;CHANGE IT TO DECIMAL ASCIZ
1275 004662 004737 007422 JSR PC, @#$$SUPRS ;TYPE WITHOUT LEADING ZEROS
1276
1277 004666 104401 001204 TYPE ,SLF
1278 ; ////////////////////////////////////////////////////////////////////
1279 ; CHECK HALT AT END-OF-TEST SWITCH
1280 ; ////////////////////////////////////////////////////////////////////
1281 004672 032777 001000 174240 BIT #SW09, @SWR ;HALT AT END-OF-TEST?
1282 004700 001406 BEQ 100$ ;BR IF NO
1283 004702 000000 HALT ;YES
1284
1285 004704 022767 000176 174226 CMP #SWREG, SWR ;USING S/W SWITCH REG?
1286 004712 001001 BNE 20$ ;NO- GET OUT
1287 004714 104405 GTSWR ;GET VALUE
1288 004716 20$: ;CONTINUE
1289 004716 100$:
1290 004716 013700 000042 MOV @#42, RO ;; INSURE RO CONTAINS THE MONITORS
1291 004722 001405 BEQ $DOAGN ;; RETURN ADDRESS
1292 004724 000005 RESET ;; CLEAR THE WORLD
1293 004726 004710 SENDAD: JSR PC, (RO) ;; GO TO MONITOR
1294 004730 000240 NOP ;; SAVE ROOM
1295 004732 000240 NOP ;; FOR
1296 004734 000240 NOP ;; ACT11
1297 004736 $DOAGN:
1298 004736 000137 JMP @ (PC)+ ;; RETURN
1299 004740 002630 $RTNAD: .WORD START
1300 004742 377 377 000 $ENULL: .BYTE -1, -1, 0 ;; NULL CHARACTER STRING
1301 004745 015 042412 042116 $ENDMG: .ASCIZ <15><12>/END PASS/
1302 004752 050040 051501 000123

```

1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335

004760  
004760 104406  
004762 032777 040000 174150  
004770 001025  
004772 000416  
004774 013746 000004  
005000 012737 005020 000004  
005006 005737 177060  
005012 012637 000004  
005016 000404  
005020 022626  
005022 012637 000004  
005026 000406  
005030  
005030 105267 174046  
005034 011667 174046  
005040 105067 174037  
005044 016777 174032 174070  
005052 016716 174030  
005056 000002

```
.SBTTL SCOPE HANDLER ROUTINE
;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;CALL
;* SCOPE ;;SCOPE=IOT

$SCOPE:
CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
1$: BIT #BIT14,@SWR ;:LOOP ON PRESENT TEST?
BNE $OVER ;:YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV @#ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #5$,@#ERRVEC ;:SET FOR TIMEOUT
TST @#177060 ;:TIME OUT ON XOR?
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $SVLAD ;:GO TO THE NEXT TEST
5$: CMP (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,@#ERRVEC ;:RESTORE THE ERROR VECTOR
BR $OVER ;:LOOP ON THE PRESENT TEST
6$;*****END OF CODE FOR THE XOR TESTER*****
$SVLAD: INCB $TSTNM ;:COUNT TEST NUMBERS
MOV (SP),$LPADR ;:SAVE SCOPE LOOP ADDRESS
CLRB $ERFLG ;:ZERO THE ERROR FLAG
$OVER: MOV $TSTNM,@DISPLAY ;:DISPLAY TEST NUMBER
MOV $LPADR,(SP) ;:FUDGE RETURN ADDRESS
RTI ;:FIXES PS
```

1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376

005060  
005060 104406  
005062 105267 174015  
005066 001775  
005070 016777 174006 174044  
005076 032777 002000 174034  
005104 001402  
005106 104401 005204  
005112 005267 173774  
005116 011667 173774  
005122 162767 000002 173766  
005130 117767 173762 173756  
005136 032777 020000 173774  
005144 001004  
005146 004767 000036  
005152 104401 001203  
005156  
005156 005777 173756  
005162 100002  
005164 000000  
005166 104406  
005170  
005170 022737 004726 000042  
005176 001001  
005200 000000  
005202  
005202 000002  
005204 177607 000377

```
.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 TYPERR 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
*CALL
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

$ERROR:
7$: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
INCB $ERFLG ;;SET THE ERROR FLAG
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
MOV $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
BIT #BIT10,$SWR ;;BELL ON ERROR?
BEQ 1$ ;;NO - SKIP
TYPE $BELL ;;RING BELL
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
MOV ($SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
SUB #2,$ERRPC
MOVB $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET
BNE 20$ ;;SKIP TYPEOUTS
JSR PC,TYPERR ;;GO TO USER ERROR ROUTINE
TYPE $CRLF

20$:
2$: TST $SWR ;;HALT ON ERROR
BPL 3$ ;;SKIP IF CONTINUE
HALT ;;HALT ON ERROR!
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR

3$: CMP #SENDAD,$#42 ;;ACT-11 AUTO-ACCEPT?
BNE 6$ ;;BRANCH IF NO
HALT ;;YES

6$:
$BELL: RTI ;;RETURN
.ASCIZ <207><377><377> ;;ASCII CODE FOR BELL
```

```

1377
1378
1379 005210 104401 001203
1380 005214 010046
1381 005216 010146
1382 005220 005046
1383 005222 113716 001114
1384 005226 005316
1385 005230 006316
1386 005232 006316
1387 005234 006316
1388 005236 116601 000001
1389 005242 112600
1390 005244 066100 005252
1391 005250 000403
1392 005252 001622
1393 005254 001642
1394 005256 001732
1395
1396 005260 012067 000002
1397 005264 104401
1398 005266 000000
1399 005270 104401 001203
1400 005274 012067 000002
1401 005300 104401
1402 005302 000000
1403 005304 104401 001203
1404 005310 060107
1405 005312 000402
1406 005314 000430
1407 005316 000473
1408
1409
1410 005320 010246
1411 005322 012001
1412 005324 012002
1413 005326 000402
1414 005330 104401 013323
1415 005334 012100
1416 005336 001473
1417 005340 105722
1418 005342 001003
1419 005344 011046
1420 005346 104402
1421 005350 000767
1422 005352 010046
1423 005354 004737 007172
1424 005360 062716 000004
1425 005364 012667 000002
1426 005370 104401
1427 005372 000000
1428 005374 000755

:*****
:THIS ROUTINE WILL TYPEOUT THE ERROR MESSAGES
TYPERR: TYPE $CRLF ;TYPE "CR" & "LF"
MOV R0,-(SP) ;SAVE R0
MOV R1,-(SP) ;SAVE R1
CLR -(SP) ;PICKUP THE ITEM BYTE
MOVB 3#$ITEMB,(SP)
DEC (SP) ;ADJUST THE INDEX SO IT
ASL (SP) ;WILL WORK FOR THE
ASL (SP) ;ERROR TABLE
ASL (SP)
MOVB 1(SP),R1 ;GET THE ERROR TYPE
MOVB (SP)+,R0 ;GET THE ERROR NUMBER
ADD 1$(R1),R0 ;FORM THE TABLE POINTER
BR 2$
1$: ITEMS0
ITEMS1
ITEMS2
2$: MOV (R0)+,3$ ;PICKUP "ERROR MESSAGE" POINTER
TYPE ;TYPE "ERROR MESSAGE"
3$: 0
TYPE $CRLF ;TYPE "CR" & "LF"
MOV (R0)+,4$ ;PICKUP "DATA HEADER" POINTER
TYPE ;TYPE "DATA HEADER"
4$: 0
TYPE $CRLF ;TYPE "CR" & "LF"
ADD R1,PC ;GO TYPE THE DATA
BR ERROR0
BR ERROR1
BR ERROR2

:////////////////////
ERROR0: MOV R2,-(SP) ;SAVE R2
MOV (R0)+,R1 ;PICKUP THE "DATA POINTER"
MOV (R0)+,R2 ;PICKUP "FORMAT" POINTER
BR 2$
1$: TYPE MSG14 ;" "
2$: MOV (R1)+,R0 ;GET ADDRESS OF DATA WORD
BEQ EREXT1 ;GO TO EXIT IF 0
TSTB (R2)+ ;TYPE DECIMAL OR OCTAL?
BNE 3$ ;BR IF DECIMAL
MOV (R0),-(SP) ;;SAVE (R0) FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
BR 1$ ;GO GET NEXT DATA WORD
3$: MOV R0,-(SP)
JSR PC,2#$SDB2D ;CHANGE NUMBER TO ASCII
ADD #4,(SP) ;TYPE 6 DIGITS
MOV (SP)+,4$
TYPE ;CALL TYPE ASCII MESSAGE ROUTINE
WORD 0 ;ADDRESS OF ASCII STRING
BR 1$ ;GO GET NEXT DATA WORD

```

```

1429
1430 005376 011001
1431 005400 013146
1432
1433 005402 104402
1434 005404 012767 000003 000006
1435
1436 005412 104401 013323
1437 005416 005327
1438 005420 000000
1439 005422 001412
1440 005424 012146
1441 005426 004737 007172
1442 005432 062716 000004
1443 005436 012667 000002
1444 005442 104401
1445 005444 000000
1446 005446 000761
1447 005450 013101
1448 005452 016167 005466 000002
1449 005460 104401
1450 005462 000000
1451 005464 000421
1452
1453
1454 005466 012067
1455 005470 012106
1456 005472 012114
1457 005474 012121
1458 005476 012145
1459 005500 012170
1460 005502 012213
1461 005504 012235
1462
1463
1464 005506 011000
1465 005510 005710
1466 005512 001406
1467 005514 013046
1468 005516 104402
1469 005520 104401 013323
1470 005524 000771
1471
1472
1473 005526 012602
1474 005530 012601
1475 005532 012600
1476 005534 000207
1477
1478

```

```

:////////////////////
ERROR1: MOV (R0),R1 ;PICKUP THE "DATA POINTER"
MOV @ (R1)+,-(SP) ;;SAVE @ (R1)+ FOR TYPEOUT
;;SERRPC
TYP0C ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
MOV #3,2$ ;PRINT 2 SPACES 3 TIMES
;AND 2 DECIMAL #'S 2 TIMES
1$: TYPE MSG14 ;"
DEC (PC)+ ;TYPE ANOTHER # ?
2$: .WORD 0 ;BR IF NO
BEQ 4$ ;YES---PICKUP ADDRESS OF #
MOV (R1)+,-(SP) ;GO CHANGE TO DECIMAL ASCIZ
JSR PC,@#$DB2D ;ONLY TYPE THE LAST 6 DIGITS
ADD #4,(SP) ;SAVE ADDRESS OF ASCIZ STRING
MOV (SP)+,3$ ;CALL THE TYPE ROUTINE
TYPE ;POINTER GOES HERE
3$: .WORD 0 ;LOOP
BR 1$ ;PICKUP FUNCTION INDEX
4$: MOV @ (R1)+,R1 ;GET THE FUNCTION MESSAGE
MOV BADFUN(R1),5$ ;AND TYPE IT
TYPE ;MESSAGE POINTER GOES HERE
5$: .WORD 0 ;GO TO EXIT
BR EREXT2
;THIS TABLE CONTAINS THE POINTERS TO THE DIFFERENT ASCIZ MESSAGES
;FOR THE FUNCTION BEING PERFORMED WHEN THE ERROR OCCURRED.
BADFUN: MXWFG ;WRITE-FILE-GAP
MXWRIT ;WRITE
MXREAD ;READ
MXBSFG ;BACK-SPACE-FILE-GAP
MXBSBG ;BACK-SPACE-BLK-GAP
MXSFFG ;SPACE-FWD-FILE-GAP
MXSFBG ;SPACE-FWD-BLK-GAP
MXRWND ;REWIND
:////////////////////
ERROR2: MOV (R0),R0 ;PICKUP THE DATA POINTER
1$: TST (R0) ;ALL DATA TYPED?
BEQ EREXT2 ;BR IF YES
MOV @ (R0)+,-(SP) ;;SAVE @ (R0)+ FOR TYPEOUT
TYP0C ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE MSG14 ;"
BR 1$ ;LOOP
:////////////////////
EREXT1: MOV (SP)+,R2 ;RESTORE R2
EREXT2: MOV (SP)+,R1 ;RESTORE R1
MOV (SP)+,R0 ;RESTORE R0
RTS PC ;RETURN

```



```

1479
1480
1481
1482
1483 005536 113737 001304 001162 CSRERR: MOVB @#FILE,@#$REG0 ;GET THE FILE NUMBER
1484 005544 113737 001306 001166 MOVB @#BLOCK,@#$REG2 ;GET THE BLOCK NUMBER
1485 005552 013746 007170 MOV @#FUNLOC,-(SP) ;GET THE LAST FUNCTION
1486 005556 042716 177761 BIC #CFUNCTION,(SP) ;CLEAR AWAY THE JUNK
1487 005562 011637 001172 MOV (SP),@#$TMP0 ;SAVE THE FUNCTION CODE
1488 005566 113766 001612 000001 MOVB @#PARMBK,1(SP) ;COMBINE THE ST/ER WITH
1489 005574 012601 MOV (SP)+,R1 ;THE FUNCTION
1490 005576 005046 CLR -(SP) ;FIND OUT WHAT CAUSED THE ERROR
1491 005600 032701 001000 BIT #OFFLINE,R1 ;OFF LINE?
1492 005604 001025 BNE 6$ ;BR IF YES
1493 005606 032701 010000 BIT #WRTLOCK,R1 ;WRITE LOCK?
1494 005612 001021 BNE 5$ ;BR IF YES
1495 005614 032701 020000 BIT #LEADER,R1 ;BOT/EOT?
1496 005620 001015 BNE 4$ ;BR IF YES
1497 005622 032701 004000 BIT #FGAP,R1 ;FILE GAP?
1498 005626 001011 BNE 3$ ;BR IF YES
1499 005630 032701 002000 BIT #TIMERR,R1 ;TIMING?
1500 005634 001005 BNE 2$ ;BR IF YES
1501 005636 032701 040000 BIT #CRCERR,R1 ;BLOCK CHECK?
1502 005642 001001 BNE 1$ ;BR IF YES
1503 005644 005216 INC (SP) ; 6 = UNKNOWN
1504 005646 005216 1$: INC (SP) ; 5 = CRCERR
1505 005650 005216 2$: INC (SP) ; 4 = TIMERR
1506 005652 005216 3$: INC (SP) ; 3 = FGAP
1507 005654 005216 4$: INC (SP) ; 2 = LEADER
1508 005656 005216 5$: INC (SP) ; 1 = WRTLOCK
1509 005660 6$:
1510 005660 106316 ASLB (SP) ;POSITION INDEX
1511 005662 062607 ADD (SP)+,PC ;BR TO THE ROUTINE
1512 005664 000406 BR OL.ERR
1513 005666 000407 BR WL.ERR
1514 005670 000410 BR CL.ERR
1515 005672 000462 BR FG.ERR
1516 005674 000463 BR TM.ERR
1517 005676 000464 BR BC.ERR
1518 005700 000472 BR X.ERR
1519
1520 005702 104101 OL.ERR: ERROR 101 ;DRIVE IS OFFLINE
1521 005704 000200 RTS RD
1522
1523 005706 104102 WL.ERR: ERROR 102 ;DRIVE IS WRITE LOCK
1524 005710 000200 RTS RD
1525

```

# F05

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C  
 DZTAECP11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 36  
 DETERMINE CONTROL AND STATUS ERROR

1526	005712	022737	000000	001172	CL.ERR:	CMP	#XWFG, @#\$TMPD		;FUNCTION "WRITE FILE GAP"?
1527	005720	001014				BNE	2\$		;BR IF NO
1528	005722				1\$:				
1529	005722	162737				SUB	(PC)+, @\$(PC)+		;DECREMENT THE FILE NUMBER
1530	005724	001	377			.BYTE	1, -1		
1531	005726	001304				.WORD	FILE		
1532	005730	113737	001304	001236		MOVB	@#FILE, @#LASTFL		;SAVE LAST FILE NUMBER
1533	005736	113737	001310	001234		MOVB	@#FILESZ, @#LASTBK		;AND LAST BLOCK NUMBER
1534	005744	005337	001234			DEC	@#LASTBK		
1535	005750	000425				BR	3\$		;BR IF NO
1536	005752	022737	000002	001172	2\$:	CMP	#XWRITE, @#\$TMPD		;FUNCTION "WRITE"?
1537	005760	001025				BNE	4\$		;BR IF NO
1538	005762	113737	001304	001236		MOVB	@#FILE, @#LASTFL		;SAVE LAST FILE NUMBER
1539	005770	162737				SUB	(PC)+, @\$(PC)+		;DECREMENT THE BLOCK NUMBER
1540	005772	001	377			.BYTE	1, -1		
1541	005774	001306				.WORD	BLOCK		
1542	005776	113737	001306	001234		MOVB	@#BLOCK, @#LASTBK		;SAVE LAST BLOCK NUMBER
1543	006004	163737	007164	001220		SUB	@#CBCNT, @#WBYTTL		;SUBTRACT RESIDUE COUNT FROM
1544	006012	005637	001222			SBC	@#WBYTTL+2		;THE TOTAL WRITE COUNT
1545	006016	005137	001306			COM	@#BLOCK		;IF NO BLOCKS IN THIS FILE
1546	006022	001737				BEQ	1\$		;BACK TO LAST FILE
1547	006024	022020			3\$:	CMP	(R0)+, (R0)+		;ADJUST FOR RETURN
1548	006026	005237	001212			INC	@#EOTS		;COUNT END-OF-TAPE
1549	006032	000401				BR	5\$		
1550	006034	104103			4\$:	ERROR	103		;"CLEAR LEADER" ERROR
1551	006036	000200			5\$:	RTS	R0		;RETURN
1552									
1553	006040	104104			FG.ERR:	ERROR	104		;"FILE GAP ERROR"
1554	006042	000200				RTS	R0		
1555									
1556	006044	104105			TM.ERR:	ERROR	105		;TIMING ERROR
1557	006046	000200				RTS	R0		
1558									
1559	006050	104106			BC.ERR:	ERROR	106		;BLOCK CHECK ERROR
1560	006052	022737	000004	001172		CMP	#XREAD, @#\$TMPD		;IF FUNCTION ISN'T A READ
1561	006060	001001				BNE	1\$		;TAKE THE "EOP" EXIT
1562	006062	022020				CMP	(R0)+, (R0)+		;ADJUST FOR RETURN
1563	006064	000200			1\$:	RTS	R0		
1564									
1565	006066	104107			X.ERR:	ERROR	107		;UNKNOWN INTERRUPT
1566	006070	000200				RTS	R0		

1567  
 1568  
 1569  
 1570  
 1571  
 1572  
 1573  
 1574  
 1575  
 1576  
 1577  
 1578  
 1579  
 1580  
 1581  
 1582  
 1583  
 1584  
 1585  
 1586  
 1587  
 1588  
 1589  
 1590  
 1591  
 1592  
 1593  
 1594  
 1595  
 1596  
 1597  
 1598  
 1599  
 1600  
 1601  
 1602  
 1603  
 1604  
 1605  
 1606  
 1607  
 1608  
 1609  
 1610  
 1611  
 1612  
 1613  
 1614  
 1615  
 1616  
 1617  
 1618  
 1619  
 1620  
 1621  
 1622

006072 013700 001306  
 006076 042700 177760  
 006102 006300  
 006104 066037 001312 001220  
 006112 005537 001222  
 006116 016037 001312 001620  
 006124 016000 001352  
 006130 004737 006136  
 006134 000207  
 006136  
 006136 010046  
 006140 010146  
 006142 010246  
 006144 010346  
 006146 012701 000200  
 006152 012702 013464  
 006156 013722 001304  
 006162 013722 001306  
 006166 010003  
 006170 010300  
 006172 012022

\*\*\*\*\*  
 \*\*\*\*\*

.SBTTL ROUTINE TO SETUP FOR A WRITE OPERATION

: THIS ROUTINE WILL SETUP THE BYTE COUNT, DETERMINE THE PATTERN TO  
 : BE WRITTEN ON TAPE, ADD THE SIZE OF THIS BLOCK TO THE TOTAL  
 : NUMBER OF BYTES TRANSFERRED AND CALL THE ROUTINE TO FILL THE  
 : WRITE BUFFER.

: CALL  
 : JSR PC, @#SETUPW

SETUPW: MOV @#BLOCK, R0 ; GET THE BLOCK NUMBER  
 BIC #1C15., R0 ; KEEP MAX PATTERN  
 ASL R0 ; \*2  
 ADD BLKSZ(R0), @#WBYTTL ; ADD THE BLOCK SIZE TO THE TOTAL  
 ADC @#WBYTTL+2 ; NUMBER OF BYTES WRITTEN  
 MOV BLKSZ(R0), @#PARMBK+6 ; SET THE BYTE COUNT  
 MOV PATS(R0), R0 ; GET PATTERN POINTER  
 JSR PC, @#FILL ; GO FILL WRITE BUFFER  
 RTS PC ; RETURN

\*\*\*\*\*  
 \*\*\*\*\*

.SBTTL ROUTINE TO FILL THE BUFFER BEFORE A WRITE

: CALL  
 : MOV #PATADR, R0 ; 1ST ADDRESS OF 8 BYTE PATADR  
 : JSR PC, @#FILL  
 : THE BUFFER IS FILLED AS FOLLOWS:

: BYTE# DATA  
 : 0001 FILE NUMBER  
 : 0002 FILE NUMBER COMPLEMENTED  
 : 0003 BLOCK NUMBER  
 : 0004 BLOCK NUMBER COMPLEMENTED  
 : 0005 DATA PATTERN  
 : 0006 DATA PATTERN  
 : \* \* \*  
 : \* \* \*  
 : \* \* \*  
 : 1027. DATA PATTERN  
 : 1028. DATA PATTERN

FILL:  
 MOV R0, -(SP) ;: PUSH R0 ON STACK  
 MOV R1, -(SP) ;: PUSH R1 ON STACK  
 MOV R2, -(SP) ;: PUSH R2 ON STACK  
 MOV R3, -(SP) ;: PUSH R3 ON STACK  
 MOV #1024./8., R1 ;: FILL 1024. BYTES WITH PATTERN  
 MOV #BUFFER, R2 ;: 1ST ADDRESS OF DATA BUFFER  
 MOV @#FILE, (R2)+ ;: FILE NUMBER AND ITS COMPLEMENT  
 MOV @#BLOCK, (R2)+ ;: BLOCK NUMBER AND ITS COMPLEMENT  
 MOV R0, R3 ;: SAVE FIRST ADDRESS OF PATTERN  
 1\$: MOV R3, R0 ;: RESET PATTERN POINTER  
 MOV (R0)+, (R2)+ ;: MOVE THE PATTERN

H05

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C  
DZTAECP11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 38  
ROUTINE TO FILL THE BUFFER BEFORE A WRITE

```

1623 006174 012022      MOV      (R0)+,(R2)+      ; INTO THE BUFFER AREA.
1624 006176 012022      MOV      (R0)+,(R2)+
1625 006200 012022      MOV      (R0)+,(R2)+
1626 006202 005301      DEC      R1
1627 006204 001371      BNE      1$
1628 006206 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
1629 006210 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
1630 006212 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
1631 006214 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
1632 006216 000207      RTS      PC

```

```

1633
1634
1635      ;; *****
1636      ;; *****
1637      .SBTTL      ROUTINE TO SETUP FOR A READ

```

1638 ; THIS ROUTINE WILL SETUP THE BYTE COUNT AND ADD THE SIZE OF THIS  
1639 ; BLOCK TO THE TOTAL NUMBER OF BYTES READ.

```

1640
1641 ; CALL
1642 ;     JSR      PC,SETUPR
1643
1644

```

```

1645 006220 013700 001306      SETUPR: MOV      @#BLOCK,R0      ; GET THE BLOCK NUMBER
1646 006224 042700 177760      BIC      #1C15.,R0      ; KEEP MAX PATTERN
1647 006230 006300      ASL      R0      ; X2
1648 006232 066037 001312 001214      ADD      BLKSZ(R0),@#RBYTTL ; ADD THE BLOCK SIZE TO THE TOTAL
1649 006240 005537 001216      ADC      @#RBYTTL+2      ; NUMBER OF BYTES READ
1650 006244 016037 001312 001620      MOV      BLKSZ(R0),@#PARMBK+6 ; SET THE BYTE COUNT
1651 006252 000207      RTS      PC      ; RETURN

```

```

1652
1653      ;; *****
1654      .SBTTL      ROUTINE TO CHECK FOR SYNC PROBLEMS

```

1655 ; THIS ROUTINE CHECKS THE FIRST FOUR BYTES OF THE DATA  
1656 ; BUFFER TO INSURE THE PROPER BLOCK WAS READ  
1657 ; BYTE 1 = FILE NUMBER  
1658 ; BYTE 2 = FILE NUMBER COMPLEMENT  
1659 ; BYTE 3 = BLOCK NUMBER  
1660 ; BYTE 4 = BLOCK NUMBER COMPLEMENT

```

1661
1662 ; CALL
1663 ;     JSR      PC,@#SYNCK
1664 ;     NORMAL RETURN
1665 ;     ERROR RETURN
1666
1667

```

```

1668
1669
1670 006254 012700 013464      SYNCK:  MOV      #BUFFER,R0      ; ADDRESS OF FIRST WORD
1671 006260 011001      MOV      (R0),R1      ; CHECK FILE # WAS READ OK
1672 006262 000301      SWAB    R1      ; BYTE 1 SHOULD BE THE
1673 006264 062001      ADD      (R0)+,R1      ; COMPLEMENT OF BYTE 0
1674 006266 005101      COM     R1
1675 006270 001031      BNE     2$      ; BR IF FILE NUMBER READ WRONG
1676 006272 011001      MOV      (R0),R1      ; CHECK BLOCK # WAS READ OK
1677 006274 000301      SWAB    R1      ; BYTE 3 SHOULD BE THE
1678 006276 062001      ADD      (R0)+,R1      ; COMPLEMENT OF BYTE 2

```

```

1679 006300 005101 COM R1
1680 006302 001024 BNE 2$ ;BR IF BLOCK NUMBER READ WRONG
1681 006304 023740 001306 CMP 2$BLOCK,-(R0) ;MAKE SURE PROPER BLOCK WAS READ
1682 006310 001003 BNE 1$ ;BR IF AT THE WRONG BLOCK
1683 006312 023740 001304 CMP 2$FILE,-(R0) ;MAKE SURE PROPER FILE
1684 006316 001416 BEQ 2$ ;BR IF IN THE PROPER FILE
1685 006320 113737 001304 001162 1$: MOVB 2$FILE,2$SREG0
1686 006326 113737 001306 001166 MOVB 2$BLOCK,2$SREG2
1687 006334 113737 013464 001172 MOVB 2$BUFFER,2$STMP0
1688 006342 113737 013466 001176 MOVB 2$BUFFER+2,2$STMP2
1689 006350 062716 000002 ADD #2,(SP) ;TAKE ERROR RETURN
1690 006354 000207 2$: RTS PC ;RETURN
1691
1692 ;*****
1693 .SBTTL ROUTINE TO CHECK THE READ DATA
1694
1695 ;CALL
1696 ; JSR PC,DATCMP
1697 ; NORMAL RETURN ;ERROR FLAG=0 AND DATA IS GOOD
1698 ; ERROR1 RETURN ;DATA IS BAD (ERROR FLAG=?)
1699 ; ERROR2 RETURN ;DATA IS GOOD BUT ERROR FLAG=1
1700
1701 006356 013700 001306 DATCMP: MOV 2$BLOCK,R0 ;GET THE BLOCK NUMBER
1702 006362 042700 177760 BIC #15.,R0 ;KEEP MAX PATTERN
1703 006366 006300 ASL R0 ;X2
1704 006370 016001 001312 MOV BLKSZ(R0),R1 ;PICKUP THE BLOCK SIZE
1705 006374 012702 013464 MOV #BUFFER,R2 ;FIRST ADDRESS OF DATA
1706 006400 012703 000004 MOV #4,R3 ;SETUP TO CHECK FILE AND BLOCK
1707 006404 012704 001304 MOV #FILE,R4
1708 006410 005301 1$: DEC R1 ;COUNT THIS BYTE
1709 006412 122224 CMPB (R2)+,(R4)+ ;CHECK IT
1710 006414 001015 BNE 4$ ;BR IF BAD
1711 006416 005303 DEC R3 ;AGAIN?
1712 006420 001373 BNE 1$ ;BR IF YES
1713 006422 012703 000010 2$: MOV #8,R3 ;GET NUMBER OF BYTES/PATTERN
1714 006426 016004 001352 MOV PAT5(R0),R4 ;GET FIRST ADDRESS OF PATTERN
1715 006432 005301 3$: DEC R1 ;LAST BYTE BEEN CHECKED?
1716 006434 002437 BLT 5$ ;BR IF YES
1717 006436 122224 CMPB (R2)+,(R4)+ ;NO--CHECK IT
1718 006440 001003 BNE 4$ ;BR IF BAD
1719 006442 005303 DEC R3 ;END OF PATTERN?
1720 006444 003372 BGT 3$ ;BR IF NO
1721 006446 000765 BR 2$ ;YES
1722 006450 005037 4$: CLR 2$SGDDAT
1723 006454 005037 CLR 2$SBDDAT
1724 006460 114437 001124 MOVB -(R4),2$SGDDAT ;SAVE GOOD DATA
1725 006464 010437 001120 MOV R4,2$SGDADR ;SAVE GOOD ADDRESS
1726 006470 114237 001126 MOVB -(R2),2$SBDDAT ;SAVE BAD DATA
1727 006474 010237 001122 MOV R2,2$SBDADR ;SAVE BAD ADDRESS
1728 006500 005401 NEG R1 ;SAVE THE BYTE NUMBER
1729 006502 066001 001312 ADD BLKSZ(R0),R1
1730 006506 010137 001230 MOV R1,2$BYTNUM
1731 006512 113737 001304 001162 MOVB 2$FILE,2$SREG0
1732 006520 113737 001306 001166 MOVB 2$BLOCK,2$SREG2
1733 006526 062716 000002 ADD #2,(SP) ;SETUP TO TAKE ERROR1 EXIT
1734 006532 000405 BR 6$ ;TAKE ERROR1 EXIT

```

```

1735 006534 105737 001612 5$: TSTB @#PARMBK ;DATA IS GOOD BUT WHAT ABOUT
1736 ;THE ERROR FLAG
1737 006540 100002 BPL 6$ ;BR IF ERROR FLAG=0
1738 006542 062716 000004 6$: ADD #4,(SP) ;SETUP TO TAKE ERROR2 EXIT
1739 006546 000207 RTS PC ;RETURN
1740
1741 ;*****
1742 .SBTTL COUNT SOFT DATA ERROR
1743 ;CALL
1744 ; JSR RD,@#CNTSFT
1745 ; RETURN FOR SOFT RETRY
1746 ; RETURN FOR HARD
1747
1748 CNTSFT: INC @#SOFTNM ;COUNT THIS SOFT ERROR
1749 DEC @#RDTRYS ;REREAD THE RECORD?
1750 006550 005237 001206 BGE 1$ ;BR IF YES
1751 006554 005337 001224 MOV @#MAXRDS,@#RDTRYS ;RESET REREAD COUNT
1752 006560 002011 SUB @#MAXRDS,@#SOFTNM ;DON'T COUNT AS SOFT
1753 006562 013737 001260 001224 DEC @#SOFTNM
1754 006570 163737 001260 001206 TST (RD)+ ;STEP OVER THE REREAD RETURN
1755 006576 005337 001206 1$: RTS RD ;RETURN
1756 006602 005720
1757 006604 000200
1758 ;*****
1759 .SBTTL COUNT HARD ERROR
1760 ;CAL
1761 ; JSR RD,@#CNTHRD
1762 ; RETURN MORE ERRORS ALLOWED
1763 ; RETURN MAX. ERRORS HAVE OCCURRED
1764
1765 CNTHRD: INC @#HARDNM ;COUNT THIS HARD ERROR
1766 006606 005237 001210 001262 CMP @#HARDNM,@#MAXERR ;MAX. HARD ERRORS OCCURRED
1767 006612 023737 001210 001262 BLT 1$ ;BR IF NO
1768 006620 002401 TST (RD)+ ;TAKE FATAL EXIT
1769 006622 005720 1$: RTS RD ;RETURN
1770 006624 000200
1771

```

1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791 006626  
1792 006626 010046  
1793 006630 010146  
1794 006632 010246  
1795 006634 010346  
1796 006636 010446  
1797 006640 010546  
1798 006642 016646 000022  
1799 006646 016646 000022  
1800 006652 016646 000022  
1801 006656 016646 000022  
1802 006662 000002  
1803  
1804  
1805  
1806  
1807 006664  
1808 006664 012666 000022  
1809 006670 012666 000022  
1810 006674 012666 000022  
1811 006700 012666 000022  
1812 006704 012605  
1813 006706 012604  
1814 006710 012603  
1815 006712 012602  
1816 006714 012601  
1817 006716 012600  
1818 006720 000002  
1819  
1820

;;\*\*\*\*\*

.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

;;\*\*\*\*\*

;;\*SAVE RO-R5

;;\*CALL:

;;\* SAVREG

;;\*UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:

;;\*

;;\*TOP---(+16)

;;\* +2---(+18)

;;\* +4---R5

;;\* +6---R4

;;\* +8---R3

;;\*+10---R2

;;\*+12---R1

;;\*+14---R0

\$SAVREG:

MOV	R0,-(SP)	;;	PUSH R0 ON STACK
MOV	R1,-(SP)	;;	PUSH R1 ON STACK
MOV	R2,-(SP)	;;	PUSH R2 ON STACK
MOV	R3,-(SP)	;;	PUSH R3 ON STACK
MOV	R4,-(SP)	;;	PUSH R4 ON STACK
MOV	R5,-(SP)	;;	PUSH R5 ON STACK
MOV	22(SP),-(SP)	;;	SAVE PS OF MAIN FLOW
MOV	22(SP),-(SP)	;;	SAVE PC OF MAIN FLOW
MOV	22(SP),-(SP)	;;	SAVE PS OF CALL
MOV	22(SP),-(SP)	;;	SAVE PC OF CALL
RTI			

;;\*RESTORE RO-R5

;;\*CALL:

;;\* RESREG

\$RESREG:

MOV	(SP)+,22(SP)	;;	RESTORE PC OF CALL
MOV	(SP)+,22(SP)	;;	RESTORE PS OF CALL
MOV	(SP)+,22(SP)	;;	RESTORE PC OF MAIN FLOW
MOV	(SP)+,22(SP)	;;	RESTORE PS OF MAIN FLOW
MOV	(SP)+,R5	;;	POP STACK INTO R5
MOV	(SP)+,R4	;;	POP STACK INTO R4
MOV	(SP)+,R3	;;	POP STACK INTO R3
MOV	(SP)+,R2	;;	POP STACK INTO R2
MOV	(SP)+,R1	;;	POP STACK INTO R1
MOV	(SP)+,R0	;;	POP STACK INTO R0
RTI			

```

1821 ;*****
1822
1823 .SBTTL          CASSETTE PRIMITIVES ROUTINE
1824
1825 ;CALLED VIA     JSR  PC,(FUNCTION)
1826 ;WITH RO POINTING TO:
1827
1828 :              .BYTE  0
1829 :              .BYTE  DRIVE # (0=A,1=B)
1830 :              .WORD  POINTER TO STATUS/ERROR BYTE
1831 :              .WORD  BUFFER ADDRESS
1832 :              .WORD  BYTE COUNT
1833
1834 006722 010746 WFG:    MOV    PC, -(SP)          ; WRITE A FILE GAP
1835 006724 000415 FUNTAB: BR    IOCOM
1836 006726 010746 WRITE:  MOV    PC, -(SP)          ; WRITE A BLOCK OF DATA
1837 006730 000413          BR    IOCOM
1838 006732 010746 READ:   MOV    PC, -(SP)          ; READ A BLOCK OF DATA
1839 006734 000411          BR    IOCOM
1840 006736 010746 BSFG:  MOV    PC, -(SP)          ; BACK SPACE A FILE GAP
1841 006740 000407          BR    IOCOM
1842 006742 010746 BSBG:  MOV    PC, -(SP)          ; BACK SPACE A BLOCK GAP
1843 006744 000405          BR    IOCOM
1844 006746 010746 SFFG:  MOV    PC, -(SP)          ; SPACE FORWARD A FILE GAP
1845 006750 000403          BR    IOCOM
1846 006752 010746 SFBG:  MOV    PC, -(SP)          ; SPACE FORWARD A BLOCK GAP
1847 006754 000401          BR    IOCOM
1848 006756 010746 REWIND: MOV   PC, -(SP)          ; REWIND TO BEGINNING OF TAPE
1849 006760 162716 006724 IOCOM:  SUB    #FUNTAB, (SP)          ; PC ON STACK IS USED TO DETERMINE
1850 006764 006216          ASR    (SP)                ; FUNCTION
1851 006766 032777 000040 172244 CHKRDY: BIT    #40, @TACSL          ; READY BIT UP?
1852 006774 001774          BEQ    CHKRDY                ; WAIT ON IT
1853 006776 012667 000166          MOV    (SP)+, FUNLOC          ; PUT FUNC. IN BITS 1-3 OF FUNLOC
1854 007002 010146          MOV    R1, -(SP)
1855 007004 010046          MOV    RO, -(SP)
1856 007006 012701 007162          MOV    #CASPAR, R1          ; PT. TO INT. HANDLER PARAM. BLK.
1857 007012 016011 000002          MOV    2(RO), (R1)          ; PTR. TO ST/ERR WD. INTO P.BLK.
1858 007016 105031          CLRB  @R1+                ; ZERO ST/ERR BYTE
1859 007020 005200          INC   RO
1860 007022 152067 000143 CHANOK: BISB  (RO)+, FUNLOC+1          ; COPY CHAN. NUM. TO FUNC. WORD
1861 007026 005720          TST   (RO)+
1862 007030 012061 000002          MOV   (RO)+, 2(R1)          ; BUFF. ADR.
1863 007034 011021          MOV   (RO), (R1)+          ; BYTE COUNT
1864 007036 005721          TST   (R1)+
1865 007040 152711 000101          BISB  #101, (R1)          ; ADD INT. ENABLE AND GO BITS
1866 :              ; TO THE FUNCTION
1867 007044 011177 172170 DOFUN:  MOV   (R1), @TACSL          ; START THE FUNCTION
1868 007050 012600          MOV   (SP)+, RO
1869 007052 012601          MOV   (SP)+, R1
1870 007054 000207          RTS   PC

```



```

1971 ;*****
1972 ;
1973 ;SBTTL          CASSETTE INTERRUPT HANDLER
1974 ;
1975 ;
1976 CASINT:
1977 007056 010146      MOV      R1,-(SP)          ;;PUSH R1 ON STACK
1978 007060 010246      MOV      R2,-(SP)          ;;PUSH R2 ON STACK
1979 007062 016702 172152  MOV      TACSL,R2
1980 007066 012701 007164  MOV      #CBCNT,R1          ;PARAM. BLOCK
1981 007072 105712      TSTB     (R2)              ;TRANSFER REQUEST?
1982 007074 100411      BMI      TREQ              ;YES
1983 007076 011204      MOV      (R2),R4
1984 007100 000304      SWAB    R4
1985 007102 052704 000001  BIS      #1,R4              ;SET THE DONE BIT
1986 007106 110451      MOVB    R4,@-(R1)          ;LOAD ST/ERR BYTE
1987 007110 105012      CLRB   (R2)              ;TURN OFF INTERS.
1988 007112 012602      MOV     (SP)+,R2          ;;POP STACK INTO R2
1989 007114 012601      MOV     (SP)+,R1          ;;POP STACK INTO R1
1990 007116 000002      RTI
1991 ;
1992 007120 005321      TREQ:   DEC     (R1)+
1993 007122 100412      BMI     RWILBS            ;WHEN COUNT NEG.  DONE
1994 007124 032712 000004  BIT     #4,(R2)          ;CHECK READ OR WRITE FUNC.
1995 007130 001003      BNE     READR
1996 007132 113177 172104  MOVB   @R1+,@TADBL        ;WRITE BYTE
1997 007136 000402      BR      RWDONE
1998 007140 117731 172076  READR: MOVB   @TADBL,@(R1)+ ;READ BYTE
1999 007144 005241      RWDONE: INC    -(R1)       ;BUMP  ADDR.
2000 007146 000402      BR      BACK              ;GET BACK
2001 ;
2002 007150 052712 000020  RWILBS: BIS     #20,(R2)    ;INIT. LAST BYTE SEQ.
2003 007154      BACK:
2004 007154 012602      MOV     (SP)+,R2          ;;POP STACK INTO R2
2005 007156 012601      MOV     (SP)+,R1          ;;POP STACK INTO R1
2006 007160 000002      RTI
2007 ;
2008 ; CASSETTE INTER. HANDLER PARAM. BLOCK
2009 ;
2010 ;
2011 007162 000000      CASPAR: .WORD  0          ;STATUS/ERROR POINTER
2012 007164 000000      CBCNT:  .WORD  0          ;WORKING BYTE COUNT
2013 007166 000000      FUNLOC: .WORD  0          ;WORKING BUFFER ADDRESS
2014 007170 000000      FUNLOC: .WORD  0          ;FUNC. WORD BUILT HERE
2015 ;
2016 ;

```

```

1917 ;*****
1918 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
1919
1920 ;*****
1921 ;*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
1922 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
1923 ;*POSITIVE.
1924 ;*CALL
1925 ;*   MOV   #PNTR, -(SP)   ;; POINTER TO LOW WORD OF BINARY NUMBER
1926 ;*   JSR   PC, @#$DB2D   ;; THE FIRST ADDRESS OF ASCII
1927 ;*   RETURN                ;; IS ON THE STACK
1928
1929
1930
1931 $DB2D: SAVREG                ;; SAVE REGISTERS
1932 MOV     2(SP), R2           ;; PICKUP THE DATA POINTER
1933 MOV     #$DECVL, R0        ;; GET ADDRESS OF "$DECVL" STRING
1934 MOV     R0, 2(SP)          ;; PUT ADDRESS OF ASCII STRING ON STACK
1935 MOV     (R2)+, R1          ;; PICKUP THE BINARY NUMBER
1936 MOV     (R2)+, R2
1937 MOV     #10, 4$           ;; SET UP TO DO 10 CONVERSIONS
1938 MOV     #STNPWR, R4        ;; ADDRESS OF TEN POWER
1939 MOV     #STNPWR+2, R5
1940
1941 1$: CLR     R3                ;; CLEAR PARTIAL
1942 2$: SUB     (R4), R1          ;; SUBTRACT TEN POWER
1943 SBC     R2
1944 SUB     (R5), R2
1945 BLT     3$                 ;; BR IF TEN POWER TO LARGE
1946 INC     R3                ;; ADD 1 TO PARTIAL
1947 BR     2$                 ;; LOOP
1948 3$: ADD     (R4)+, R1        ;; RESTORE SUBTRACTED VALUE
1949 ADC     R2
1950 ADD     (R4)+, R2
1951 CMP     (R5)+, (R5)+        ;; MOVE TO NEXT TEN POWER
1952 BIS     #'0, R3            ;; CHANGE PARTIAL TO ASCII
1953 MOVB   R3, (R0)+          ;; SAVE IT
1954 DEC     (PC)+              ;; DONE?
1955 4$: .WORD 0
1956 BNE     1$                 ;; BR IF NO
1957 CLRB   (R0)+              ;; TERMINATOR
1958 RESREG                ;; RESTORE REGISTERS
1959 RTS     PC                 ;; RETURN
1960 $STNPWR: 145000           ;; 1.OE09
1961 35632
1962 160400                   ;; 1.OE08
1963 2765
1964 113200                   ;; 1.OE07
1965 230
1966 041100                   ;; 1.OE06
1967 17
1968 103240                   ;; 1.OE05
1969 1
1970 23420                    ;; 1.OE04
1971 0
1972 1750                      ;; 1.OE03
1973 0

```

1973 007336 000144  
 1974 007340 000000  
 1975 007342 000012  
 1976 007344 000000  
 1977 007346 000001  
 1978 007350 000000  
 1979 007352 000014

144 ;;1.0E02  
 0  
 12 ;;1.0E01  
 0  
 1 ;;1.0E00  
 0

\$DECVL: .BLKB 12. ;;RESERVE STORAGE FOR ASCIZ STRING  
 .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE

\*\*\*\*\*  
 ;;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN  
 ;;UNSIGNED DECIMAL ASCII NUMBER.

\*CALL  
 ;; \* MOV NUMBER, -(SP) ;;PUT BINARY NUMBER ON THE STACK  
 ;; \* JSR PC, @\$\$SB2D ;;CALL  
 ;; \* RETURN ;;ADDRESS OF THE 1ST ASCII CHAR. IS ON THE STACK

1991 007366 016667 000002 000022  
 1992 007374 012746 007416  
 1993 007400 004737 007172  
 1994 007404 062716 000005  
 1995 007410 012666 000002  
 1996 007414 000207  
 1997 007416 000000 000000

\$SB2D: MOV 2(SP), 1\$ ;;SAVE BINARY NUMBER  
 MOV #1\$, -(SP) ;;SET POINTER  
 JSR PC, @\$\$SB2D ;;CALL DOUBLE LENGTH CONVERT  
 ADD #5, (SP) ;;ONLY ALLOW FIVE CHARACTERS  
 MOV (SP)+, 2(SP) ;;PICKUP POINTER  
 RTS PC ;;RETURN

1\$: .WORD 0,0  
 .SBTTL TYPE NUMERICAL ASCII STRING SUPPRESS LEADING ZEROS

\*\*\*\*\*  
 ;;THIS ROUTINE IS USED TO TYPE AN ASCII NUMBER SUPPRESSING THE  
 ;;LEADING NUMBERS.

\*CALL  
 ;; \* MOV #NUMADR, -(SP) ;;FIRST ADDRESS OF ASCII STRING  
 ;; \* JSR PC, @\$\$SUPRS

2008 007422 010046  
 2009 007424 016600 000004  
 2010 007430 105710  
 2011 007432 001403  
 2012 007434 122720 000060  
 2013 007440 001773  
 2014 007442 005300  
 2015 007444 010067 000002  
 2016 007450 104401  
 2017 007452 000000  
 2018 007454 012600  
 2019 007456 012616  
 2020 007460 000207

\$\$SUPRS: MOV RO, -(SP) ;;SAVE RO  
 MOV 4(SP), RO ;;PICKUP THE POINTER  
 1\$: TSTB (RO) ;;TERMINATEOR?  
 BEQ 2\$ ;;BR IF YES  
 CMPB #'0, (RO)+ ;;IS THIS AN ASCII "0" ?  
 BEQ 1\$ ;;BR IF YES  
 2\$: DEC RO ;;BACKUP BY "1"  
 MOV RO, 3\$ ;;SAVE FOR TYPING  
 TYPE ;;GO TYPE  
 3\$: .WORD 0 ;;ASCII POINTER GOES HERE  
 MOV (SP)+, RO ;;RESTORE RO  
 MOV (SP)+, (SP) ;;RESTORE THE STACK  
 RTS PC ;;RETURN

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

\*\*\*\*\*  
 ;;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND  
 ;;CHANGE IT TO BINARY.

\*CALL:  
 ;; \* RDOCT ;;READ AN OCTAL NUMBER  
 ;; \* RETURN HERE ;;LOW ORDER BITS ARE ON TOP OF THE STACK

2021  
 2022  
 2023  
 2024  
 2025  
 2026  
 2027  
 2028

```

2029          ;*          ;;HIGH ORDER BITS ARE IN $HIOCT
2030
2031 007462 011646          $RDOCT: MOV (SP),-(SP)          ;; PROVIDE SPACE FOR THE
2032 007464 016666 000004 000002 MOV 4(SP),2(SP)          ;; INPUT NUMBER
2033 007472 010046          MOV R0,-(SP)          ;; PUSH R0 ON STACK
2034 007474 010146          MOV R1,-(SP)          ;; PUSH R1 ON STACK
2035 007476 010246          MOV R2,-(SP)          ;; PUSH R2 ON STACK
2036 007500 104410          1$: RDLIN          ;; READ AN ASCII LINE
2037 007502 012600          MOV (SP)+,R0          ;; GET ADDRESS OF 1ST CHARACTER
2038 007504 005001          CLR R1          ;; CLEAR DATA WORD
2039 007506 005002          CLR R2
2040 007510 112046          2$: MOVB (R0)+,-(SP)          ;; PICKUP THIS CHARACTER
2041 007512 001412          BEQ 3$          ;; IF ZERO GET OUT
2042 007514 006301          ASL R1          ;; *2
2043 007516 006102          ROL R2
2044 007520 006301          ASL R1          ;; *4
2045 007522 006102          ROL R2
2046 007524 006301          ASL R1          ;; *8
2047 007526 006102          ROL R2
2048 007530 042716 177770          BIC #1C7,(SP)          ;; STRIP THE ASCII JUNK
2049 007534 062601          ADD (SP)+,R1          ;; ADD IN THIS DIGIT
2050 007536 000764          BR 2$          ;; LOOP
2051 007540 005726          3$: TST (SP)+          ;; CLEAN TERMINATOR FROM STACK
2052 007542 010166 000012          MOV R1,12(SP)          ;; SAVE THE RESULT
2053 007546 010267 000010          MOV R2,$HIOCT
2054 007552 012602          MOV (SP)+,R2          ;; POP STACK INTO R2
2055 007554 012601          MOV (SP)+,R1          ;; POP STACK INTO R1
2056 007556 012600          MOV (SP)+,R0          ;; POP STACK INTO R0
2057 007560 000002          RTI          ;; RETURN
2058 007562 000000          $HIOCT: .WORD 0          ;; HIGH ORDER BITS GO HERE
    
```

2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075

007564 104401 013326  
007570 113746 001613  
007574 042716 177776  
007600 062716 000101  
007604 012667 000006  
007610 104401  
007612 007615  
007614 000207  
007616 000000

```
*****  
:SBTTL ROUTINE TO TYPE DRIVE  
:CALL  
: JSR PC,@#TPDRV  
:  
TPDRV: TYPE MSG15 : " DRIVE "  
MOV B @#PARMBK+1,-(SP) : PICKUP THE DRIVE NUMBER  
BIC #C1,(SP) : STRIP ANY JUNK  
ADD #'A,(SP) : MAKE IT ASCIZ  
MOV (SP)+,1$ : SAVE IT FOR TYPEOUT  
TYPE : TYPE IT  
1$ : RTS PC  
 : .WORD 0
```

```

2076 ;:*****
2077
2078 .SBTTL          ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST
2079
2080 ;CALL
2081 ;           JSR      PC,@ASKDRV
2082 ;           RETURN
2083 ;NOTE: R0 AND R1 ARE DESTROYED
2084 ASKDRV: TYPE     MSGDRV          ;<CRLF>"DRIVE(S)? "
2085           CLR     DRVKEY
2086           RDLIN
2087           MOV     (SP)+,R0
2088           TSTB   @R0
2089           BEQ     NOTLGL
2090           MOV     #DRVKEY,R1
2091 LOOP:      CMPB   #'A,@R0
2092           BNE     NOTA
2093           MOVB   (R0)+,(R1)+
2094           BR     NEXT
2095 NOTA:      CMPB   #'B,@R0
2096           BNE     NOTB
2097           MOVB   (R0)+,(R1)+
2098           BR     NEXT
2099 NOTB:      CMPB   #54,@R0
2100           BNE     NOTLGL
2101           TSTB   (R0)+
2102 NEXT:      TSTB   @R0
2103           BEQ     EXIT
2104           CMP     #DRVKEY+2,R1
2105           BHI     LOOP
2106 NOTLGL:   TYPE     $QUES
2107           BR     ASKDRV
2108 EXIT:      TST     DRVKEY
2109           BEQ     NOTLGL
2110           RTS     PC
2111

```

F06

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C  
DZTAECP11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 49  
ROUTINE TO ASK THE OPERATOR WHAT DRIVE(S) TO TEST

```

2112 :*****
2113 :SBTTL ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY
2114 :CALL
2115 :JSR PC,@#ASKADR
2116
2117 007730 010046 ASKADR: MOV RO,-(SP) ;SAVE RO
2118 007732 104401 013352 1$: TYPE ,MSGASK ;"TACS?"
2119 007736 104411 RDOCT ;GET VALUE
2120 007740 012600 MOV (SP)+,RO ;PICK UP THE OCTAL NUMBER
2121 007742 001411 BEQ 3$ ;IF "0" USE OLD VALUES
2122 007744 020027 160000 CMP RO,#160000 ;MAKE SURE IT IS A BUS ADDRESS
2123 007750 103770 BLO 1$
2124 007752 010037 001240 MOV RO,@#TACSL ;SAVE THE TACS
2125 007756 062700 000002 ADD #2,RO ;STEP TO TADB ADDRESS
2126 007762 010037 001242 MOV RO,@#TADBL ;AND SAVE IT
2127 007766 104401 013361 3$: TYPE ,MSGVEC ;"VECTOR?"
2128 007772 104411 RDOCT
2129 007774 012600 MOV (SP)+,RO
2130 007776 001411 BEQ 5$
2131 010000 020027 001000 CMP RO,#1000 ;MAKE SURE ADDRESS IS IN VECTOR AREA
2132 010004 103370 BHIS 3$
2133 010006 010037 001244 MOV RO,@#TAVEC ;SAVE AS VECTOR ADDRESS
2134 010012 062700 000002 ADD #2,RO
2135 010016 010037 001246 MOV RO,@#TAVEC+2
2136 010022 104401 013372 5$: TYPE ,MSGPRI ;ASK FOR PRIORITY
2137 010026 104411 RDOCT
2138 010030 012600 MOV (SP)+,RO
2139 010032 001413 BEQ 6$ ;IF "0" USE OLD VALUE
2140 010034 020027 000007 CMP RO,#7 ;MAKE SURE ITS VALID
2141 010040 101370 BHI 5$
2142 010042 000300 SWAB RO ;PUT INTO HIGH BYTE
2143 010044 006200 ASR RO ;AND SHIFT
2144 010046 006200 ASR RO ;INTO PROPER
2145 010050 006200 ASR RO ;POSITION
2146 010052 042700 177437 BIC #10<340>,RO ;SAVE ONLY PRIORITY BITS
2147 010056 010037 001250 MOV RO,@#TAPRIO ;STORE IT AWAY
2148 010062 104401 013405 6$: TYPE ,MTACS ;TACS="
2149 010066 016746 171146 MOV TACSL,-(SP) ;;SAVE TACSL FOR TYPEOUT
2150 010072 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2151 010074 104401 013414 TYPE ,MTADB ;"TADB="
2152 010100 016746 171136 MOV TADBL,-(SP) ;;SAVE TADBL FOR TYPEOUT
2153 010104 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2154 010106 104401 013423 TYPE ,MTAVEC ;"VECTOR="
2155 010112 016746 171126 MOV TAVEC,-(SP) ;;SAVE TAVEC FOR TYPEOUT
2156 010116 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2157 010120 104401 013435 TYPE ,MTAPRI ;"PRIORITY="
2158 010124 016746 171120 MOV TAPRIO,-(SP) ;;SAVE TAPRIO FOR TYPEOUT
2159 010130 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2160 010132 104401 013451 TYPE ,MSGOK ;"OK?"
2161 010136 104407 RDCHR ;GO READ ONE CHARACTER
2162 010140 012600 MOV (SP)+,RO ;GET IT
2163 010142 022700 000015 CMP #15,RO ;IS IT "CR"?
2164 010146 001406 BEQ 7$ ;BRANCH IF YES
2165 010150 022700 000131 CMP #Y,RO ;IS IT "Y"?
2166 010154 001403 BEQ 7$ ;IT WAS
2167 010156 104401 001202 TYPE ,SQUES ;TYPE "?"

```

G06

TA11 DATA RELIABILITY MAINDEC-11-DZTAE-C MACY11 27(1006) 17-MAR-77 14:55 PAGE 50  
DZTREC.P11 17-MAR-77 14:52 ROUTINE TO INPUT CSR,DBR, AND VECTOR ADDRESS AND PRIORITY

2168	010162	000663			BR	1\$		:AND LET HIM CORRECT THEM
2169	010164	104401	013456	7\$:	TYPE	.MYES		:TYPE OUT "YES"
2170	010170	012600			MOV	(SP)+,RO		:RESTORE RO
2171	010172	000207			RTS	PC		:AND RETURN



```

2172 ;*****
2173 ;
2174 ;SBTTL          ROUTINE TO EXAMINE DRIVE(S) FOR AVAILABILITY
2175 ;
2176 ;CALL:
2177 ;             MOV      #DRVKEY,RO
2178 ;             JSR      PC,@#EXAM          ;R1 IS DESTROYED
2179 ;             NORMAL RETURN
2180 ;             ERROR RETURN
2181 ;
2182 010174 013701 001240 EXAM: MOV      @#TACSL,R1          ;PICKUP THE "CONTROL & STATUS" REG. ADR.
2183 010200 005011          CLR      (R1)          ;DRIVE="A", FUNCTION="WFG"
2184 010202 122710 000101 CMPB    #'A,(R0)       ;EXAMINE DRIVE "A"?
2185 010206 001402          BEQ      1$           ;BR IF YES
2186 010210 052711 000400 1$: BIS      #UNIT,(R1)     ;SELECT DRIVE "B"
2187 010214 032711 000040 BIT      #READY,(R1)   ;WAIT ON READY
2188 010220 001775          BEQ      1$
2189 010222 005711          TST      (R1)          ;ANY ERROR?
2190 010224 100024          BPL      4$           ;BR IF NO
2191 010226 032711 001000 BIT      #OFFLINE,(R1) ;ERROR DUE TO "OFF LINE"?
2192 010232 001017          BNE      3$           ;BR IF YES
2193 010234 032711 010000 BIT      #WRTLOCK,(R1) ;ERROR DUE TO "WRITE LOCK"?
2194 010240 001411          BEQ      2$           ;BR IF NO
2195 010242 122777 000201 170670 CMPB    #BIT07!BIT00,@SWR ;"READONLY" SELECTED? (RD1PAS)
2196 010250 001412          BEQ      4$           ;BR IF YES
2197 010252 122777 000203 170660 CMPB    #BIT07!BIT01!BIT00,@SWR ;(RD2PAS)?
2198 010260 001406          BEQ      4$           ;BR IF YES
2199 010262 000403          BR       3$           ;TAKE THE ERROR EXIT
2200 010264 032711 020000 2$: BIT      #LEADER,(R1)  ;ERROR DUE TO "CLEAR LEADER"?
2201 010270 001002          BNE      4$           ;BR IF YES
2202 010272 062716 000002 3$: ADD      #2,(SP)      ;TAKE ERROR RETURN
2203 010276 000207          4$: RTS      PC          ;RETURN

```

.SBTTL TYPE ROUTINE

2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259

010300 105767 170653  
010304 100002  
010306 000000  
010310 000407  
010312 010046  
010314 017600 000002  
010320 112046  
010322 001005  
010324 005726  
010326 012600  
010330 062716 000002  
010334 000002  
010336 122716 000011  
010342 001430  
010344 122716 000200  
010350 001006  
010352 005726  
010354 104401  
010356 001203  
010360 105067 000130  
010364 000755  
010366 004767 000056  
010372 126726 170560  
010376 001350  
010400 016746 170550  
010404 105366 000001  
010410 002770  
010412 004767 000032  
010416 105367 000072  
010422 000770  
  
010424 112716 000040  
010430 004767 000014  
010434 132767 000007 000052  
010442 001372  
010444 005726

```
*****
:ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
:
:CALL:
:1) USING A TRAP INSTRUCTION
:* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
:*OR
:* TYPE
:* MESADR
:*
$TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
BPL 1$ ;; BR IF YES
HALT ;; HALT HERE IF NO TERMINAL
BR 3$ ;; LEAVE
1$: MOV RO,-(SP) ;; SAVE RO
MOV @2(SP),RO ;; GET ADDRESS OF ASCIZ STRING
2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
60$: MOV (SP)+,RO ;; RESTORE RO
3$: ADD #2,(SP) ;; ADJUST RETURN PC
RTI ;; RETURN
4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
BEQ 8$
CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
BNE 5$
TST (SP)+ ;; POP <CR><LF> EQUIV
TYPE A CR AND LF
CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
BR 2$ ;; GET NEXT CHARACTER
5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
BNE 2$ ;; IF NO GO GET NEXT CHAR.
MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
JSR PC,$TYPEC ;; GO TYPE A NULL
DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
BR 7$ ;; LOOP
;HORIZONTAL TAB PROCESSOR
8$: MOVB #' (SP) ;; REPLACE TAB WITH SPACE
9$: JSR PC,$TYPEC ;; TYPE A SPACE
BITB #7,$CHARCNT ;; BRANCH IF NOT AT
BNE 9$ ;; TAB STOP
TST (SP)+ ;; POP SPACE OFF STACK
```

```

2260 010446 000724 BR 2$ ;;GET NEXT CHARACTER
2261 010450 105777 170474 $TYPEPC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
2262 010454 100375 BPL $TYPEPC
2263 010456 116677 000002 170466 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2264 010464 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
2265 010472 001003 BNE 1$ ;;BRANCH IF NO
2266 010474 105067 000014 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
2267 010500 000406 BR $TYPEX ;;EXIT
2268 010502 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
2269 010510 001402 BEQ $TYPEX ;;BRANCH IF YES
2270 010512 105227 INCB (PC)+ ;;COUNT THE CHARACTER
2271 010514 000000 $CHARCNT: WORD 0 ;;CHARACTER COUNT STORAGE
2272 010516 000207 $TYPEX: RTS PC
2273
2274 .SBTTL TTY INPUT ROUTINE
2275
2276 ;:*****
2277 .ENABL LSB
2278
2279 ;:*****
2280 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2281 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2282 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2283 ;*WHEN OPERATING IN TTY FLAG MODE.
2284 010520 022767 000176 170412 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
2285 010526 001074 BNE 15$ ;;BRANCH IF NO
2286 010530 105777 170410 TSTB @STKS ;;CHAR THERE?
2287 010534 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
2288 010536 117746 170404 MOVB @STKB,-(SP) ;;SAVE THE CHAR
2289 010542 042716 177600 BIC #C177,(SP) ;;STRIP-OFF THE ASCII
2290 010546 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
2291 010552 001062 BNE 15$ ;;NO, RETURN TO USER
2292 010554 126727 170354 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
2293 010562 001456 BEQ 15$ ;;BRANCH IF YES
2294
2295 010564 104401 011245 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (↑G)
2296 010570 104401 011252 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
2297 010574 016746 167376 MOV $WREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
2298 010600 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
2299 010602 104401 011263 TYPE , $MNEW ;;PROMPT FOR NEW SWR
2300 010606 005046 19$: CLR -(SP) ;;CLEAR COUNTER
2301 010610 005046 7$: CLR -(SP) ;;THE NEW SWR
2302 010612 105777 170326 TSTB @STKS ;;CHAR THERE?
2303 010616 100375 BPL 7$ ;;IF NOT TRY AGAIN
2304
2305 010620 117746 170322 MOVB @STKB,-(SP) ;;PICK UP CHAR
2306 010624 042716 177600 BIC #C177,(SP) ;;MAKE IT 7-BIT ASCII
2307
2308
2309
2310 010630 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
2311 010634 001005 BNE 10$ ;;BRANCH IF NOT
2312 010636 104401 011240 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (↑U)
2313 010642 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
2314 010646 000757 BR 19$ ;;LET'S TRY IT AGAIN
2315

```

2316											
2317	010650	021627	000015		10\$:	CMP	(SP),#15		:: IS IT A <CR>?		
2318	010654	001022				BNE	16\$		:: BRANCH IF NO		
2319	010656	005766	000004			TST	4(SP)		:: YES, IS IT THE FIRST CHAR?		
2320	010662	001403				BEQ	11\$		:: BRANCH IF YES		
2321	010664	016677	000002	170246		MOV	2(SP),@SWR		:: SAVE NEW SWR		
2322	010672	062706	000006		11\$:	ADD	#6,SP		:: CLEAR UP STACK		
2323	010676	104401	001203		14\$:	TYPE	\$CRLF		:: ECHO <CR> AND <LF>		
2324	010702	126727	170227	000001		CMPB	\$INTAG,#1		:: RE-ENABLE TTY KBD INTERRUPTS?		
2325	010710	001003				BNE	15\$		:: BRANCH IF NOT		
2326	010712	012777	000100	170224		MOV	#100,@STKS		:: RE-ENABLE TTY KBD INTERRUPTS		
2327	010720	000002			15\$:	RTI			:: RETURN		
2328	010722	004767	177522		16\$:	JSR	PC,\$TYPEC		:: ECHO CHAR		
2329	010726	021627	000060			CMP	(SP),#60		:: CHAR < D?		
2330	010732	002420				BLT	18\$		:: BRANCH IF YES		
2331	010734	021627	000067			CMP	(SP),#67		:: CHAR > 7?		
2332	010740	003015				BGT	18\$		:: BRANCH IF YES		
2333	010742	042726	000060			BIC	#60,(SP)+		:: STRIP-OFF ASCII		
2334	010746	005766	000002			TST	2(SP)		:: IS THIS THE FIRST CHAR		
2335	010752	001403				BEQ	17\$		:: BRANCH IF YES		
2336	010754	006316				ASL	(SP)		:: NO, SHIFT PRESENT		
2337	010756	006316				ASL	(SP)		:: CHAR OVER TO MAKE		
2338	010760	006316				ASL	(SP)		:: ROOM FOR NEW ONE.		
2339	010762	005266	000002		17\$:	INC	2(SP)		:: KEEP COUNT OF CHAR		
2340	010766	056616	177776			BIS	-2(SP),(SP)		:: SET IN NEW CHAR		
2341	010772	000707				BR	7\$		:: GET THE NEXT ONE		
2342	010774	104401	001202		18\$:	TYPE	\$QUES		:: TYPE ?<CR><LF>		
2343	011000	000720				BR	20\$		:: SIMULATE CONTROL-U		
2344					.DSABL	LSB					

2344											
2345											
2346											
2347											
2348											
2349											
2350											
2351											
2352											
2353											
2354											
2355	011002	011646			\$RDCHR:	MOV	(SP),-(SP)		:: PUSH DOWN THE PC		
2356	011004	016666	000004	000002		MOV	4(SP),2(SP)		:: SAVE THE PS		
2357	011012	105777	170126		1\$:	TSTB	@STKS		:: WAIT FOR		
2358	011016	100375				BPL	1\$		:: A CHARACTER		
2359	011020	117766	170122	000004		MOVB	@STKB,4(SP)		:: READ THE TTY		
2360	011026	042766	177600	000004		BIC	#1C<177>,4(SP)		:: GET RID OF JUNK IF ANY		
2361	011034	026627	000004	000023		CMP	4(SP),#23		:: IS IT A CONTROL-S?		
2362	011042	001013				BNE	3\$		:: BRANCH IF NO		
2363	011044	105777	170074		2\$:	TSTB	@STKS		:: WAIT FOR A CHARACTER		
2364	011050	100375				BPL	2\$		:: LOOP UNTIL ITS THERE		
2365	011052	117746	170070			MOVB	@STKB,-(SP)		:: GET CHARACTER		
2366	011056	042716	177600			BIC	#1C177,(SP)		:: MAKE IT 7-BIT ASCII		
2367	011062	022627	000021			CMP	(SP)+,#21		:: IS IT A CONTROL-Q?		
2368	011066	001366				BNE	2\$		:: IF NOT DISCARD IT		
2369	011070	000750				BR	1\$		:: YES, RESUME		
2370	011072	026627	000004	000140	3\$:	CMP	4(SP),#140		:: IS IT UPPER CASE?		
2371	011100	002407				BLT	4\$		:: BRANCH IF YES		

```

*****
*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
*****

```

```

2372 011102 026627 000004 000175      CMP      4(SP),#175      ;; IS IT A SPECIAL CHAR?
2373 011110 003003                    BGT      4$             ;; BRANCH IF YES
2374 011112 042766 000040 000004      BIC      #40,4(SP)      ;; MAKE IT UPPER CASE
2375 011120 000002                    RTI                      ;; GO BACK TO USER
2376                                     ;; *****
2377                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2378                                     ;; *CALL:
2379                                     ;; *
2380                                     ;; *   RDLIN                      ;; INPUT A STRING FROM THE TTY
2381                                     ;; *   RETURN HERE              ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2382                                     ;; *                               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2383 011122 010346                    $RDLIN: MOV      R3, -(SP)      ;; SAVE R3
2384 011124 012703 011230              1$:   MOV      #$TTYIN,R3      ;; GET ADDRESS
2385 011130 022703 011240              2$:   CMP      #$TTYIN+8.,R3      ;; BUFFER FULL?
2386 011134 101405                    BLOS     4$             ;; BR IF YES
2387 011136 104407                    RDCHR                      ;; GO READ ONE CHARACTER FROM THE TTY
2388 011140 112613                    MOV      (SP)+,(R3)      ;; GET CHARACTER
2389 011142 122713 000177              10$:  CMP      #177,(R3)      ;; IS IT A RUBOUT
2390 011146 001003                    BNE     3$             ;; SKIP IF NOT
2391 011150 104401 001202              4$:   TYPE     $QUES      ;; TYPE A '?'
2392 011154 000763                    BR      1$             ;; CLEAR THE BUFFER AND LOOP
2393 011156 111367 000044              3$:   MOV      (R3),9$      ;; ECHO THE CHARACTER
2394 011162 104401 011226              TYPE     ,9$
2395 011166 122723 000015              CMP      #15,(R3)+      ;; CHECK FOR RETURN
2396 011172 001356                    BNE     2$             ;; LOOP IF NOT RETURN
2397 011174 105063 177777              CLRB    -1(R3)         ;; CLEAR RETURN (THE 15)
2398 011200 104401 001204              TYPE     $LF           ;; TYPE A LINE FEED
2399 011204 012603                    MOV      (SP)+,R3      ;; RESTORE R3
2400 011206 011646                    MOV      (SP),-(SP)     ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2401 011210 016666 000004 000002      MOV      4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
2402 011216 012766 011230 000004      MOV      #$TTYIN,4(SP)
2403 011224 000002                    RTI                      ;; RETURN
2404 011226 000          9$:   .BYTE     0              ;; STORAGE FOR ASCII CHAR. TO TYPE
2405 011227 000          .BYTE     0              ;; TERMINATOR
2406 011230 000010          $TTYIN: .BLKB     8.      ;; RESERVE 8 BYTES FOR TTY INPUT
2407 011240 052536 005015 000          $CNTLU: .ASCIZ   /?U<15><12>
2408 011245 136 006507 000012          $CNTLG: .ASCIZ   /?G<15><12>
2409 011252 005015 053523 020122          $MSWR:  .ASCIZ   <15><12>/SWR = /
2410 011260 020075 000
2411 011263 040 047040 053505          $MNEW:  .ASCIZ   / NEW = /
2412 011270 036440 000040

```

2413  
2414  
2415  
2416  
2417  
2418  
2419  
2420  
2421  
2422  
2423  
2424  
2425  
2426  
2427  
2428  
2429  
2430  
2431  
2432  
2433  
2434  
2435  
2436  
2437  
2438  
2439  
2440  
2441  
2442  
2443  
2444  
2445  
2446  
2447  
2448  
2449  
2450  
2451  
2452  
2453  
2454  
2455  
2456  
2457  
2458  
2459  
2460  
2461  
2462  
2463  
2464  
2465  
2466  
2467  
2468

```

.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
$TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
        MOV      1(SP),SOFILL    ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1  ;;NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)         ;;ADJUST RETURN ADDRESS
        BR       $TYPON
$TYPOC: MOV      #1,SOFILL       ;;SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1     ;;SET FOR SIX(6) DIGITS
$TYPON: MOV      #5,SOCNT        ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)        ;;SAVE R3
        MOV      R4,-(SP)        ;;SAVE R4
        MOV      R5,-(SP)        ;;SAVE R5
        MOV      SOMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,SOMODE       ;;SAVE IT FOR USE
        MOV      SOFILL,R4       ;;GET THE ZERO FILL SWITCH
        MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
        CLR      R3              ;;CLEAR THE OUTPUT WORD
1$:    ROL      R5               ;;ROTATE MSB INTO "C"
        BR      3$              ;;GO DO MSB
2$:    ROL      R5               ;;FORM THIS DIGIT
        ROL      R5
        ROL      R5
3$:    MOV      R5,R3           ;;GET LSB OF THIS DIGIT
        ROL      R3              ;;TYPE THIS DIGIT?
        DECB    $SOMODE         ;;BR IF NO
        BPL     7$              ;;GET RID OF JUNK
        BIC     #177770,R3     ;;TEST FOR 0
        BNE     4$              ;;SUPPRESS THIS 0?
        TST     R4              ;;BR IF YES
        BEQ     5$

```

```

011274 017646 000000
011300 116667 000001 000211
011306 112667 000207
011312 062716 000002
011316 000406
011320 112767 000001 000171
011326 112767 000006 000165
011334 112767 000005 000154
011342 010346
011344 010446
011346 010546
011350 116704 000145
011354 005404
011356 062704 000006
011362 110467 000132
011366 116704 000125
011372 016605 000012
011376 005003
011400 006105
011402 000404
011404 006105
011406 006105
011410 006105
011412 010503
011414 006103
011416 105367 000076
011422 100016
011424 042703 177770
011430 001002
011432 005704
011434 001403

```

# N06

TAI1 DATA RELIABILITY MAINDEC-11-DZTAE-C MACY11 27(1006) 17-MAR-77 14:55 PAGE 57  
 DZTREC.P11 17-MAR-77 14:52 BINARY TO OCTAL (ASCII) AND TYPE

2469	011436	005204		4\$:	INC	R4	:: DON'T SUPPRESS ANYMORE 0'S
2470	011440	052703	000060		BIS	#'0,R3	:: MAKE THIS DIGIT ASCII
2471	011444	052703	000040	5\$:	BIS	#',R3	:: MAKE ASCII IF NOT ALREADY
2472	011450	110367	000040		MOVB	R3,8\$	:: SAVE FOR TYPING
2473	011454	104401	011514		TYPE	8\$	:: GO TYPE THIS DIGIT
2474	011460	105367	000032	7\$:	DECB	\$OCNT	:: COUNT BY 1
2475	011464	003347			BGT	2\$	:: BR IF MORE TO DO
2476	011466	002402			BLT	6\$	:: BR IF DONE
2477	011470	005204			INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
2478	011472	000744			BR	2\$	:: GO DO THE LAST DIGIT
2479	011474	012605		6\$:	MOV	(SP)+,R5	:: RESTORE R5
2480	011476	012604			MOV	(SP)+,R4	:: RESTORE R4
2481	011500	012603			MOV	(SP)+,R3	:: RESTORE R3
2482	011502	016666	000002 000004		MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
2483	011510	012616			MOV	(SP)+,(SP)	
2484	011512	000002			RTI		:: RETURN
2485	011514	000		8\$:	.BYTE	0	:: STORAGE FOR ASCII DIGIT
2486	011515	000			.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
2487	011516	000		\$OCNT:	.BYTE	0	:: OCTAL DIGIT COUNTER
2488	011517	000		\$OFILL:	.BYTE	0	:: ZERO FILL SWITCH
2489	011520	000000		\$OMODE:	.WORD	0	:: NUMBER OF DIGITS TO TYPE

2490  
2491  
2492  
2493  
2494  
2495  
2496  
2497  
2498  
2499  
2500  
2501  
2502  
2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520  
2521  
2522  
2523  
2524  
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533

011522 010046  
011524 016600 000002  
011530 005740  
011532 111000  
011534 006300  
011536 016000 011556  
011542 000200  
  
011544 011646  
011546 016666 000004 000002  
011554 000002  
  
011556 011544  
011560 010300  
011562 011320  
011564 011274  
011566 011334  
  
011570 010570  
  
011572 010520  
011574 011002  
011576 011122  
011600 007462  
011602 006626  
011604 006664

.SBTTL TRAP DECODER

\*\*\*\*\*  
 \*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.

```
$TRAP:  MOV    RO, -(SP)           ;; SAVE RO
        MOV    2(SP), RO         ;; GET TRAP ADDRESS
        TST   -(RO)             ;; BACKUP BY 2
        MOVB  (RO), RO          ;; GET RIGHT BYTE OF TRAP
        ASL   RO                ;; POSITION FOR INDEXING
        MOV   $TRPAD(RO), RO     ;; INDEX TO TABLE
        RTS   RO                ;; GO TO ROUTINE
```

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

```
$TRAP2: MOV    (SP), -(SP)       ;; MOVE THE PC DOWN
        MOV    4(SP), 2(SP)     ;; MOVE THE PSW DOWN
        RTI                          ;; RESTORE THE PSW
```

.SBTTL TRAP TABLE

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

	ROUTINE		
\$TRPAD:	.WORD	\$TRAP2	
	\$TYPE	;; CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
	\$TYPOC	;; CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
	\$TYPOS	;; CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
	\$TYPON	;; CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
	\$GTSWR	;; CALL=GTSWR	TRAP+5(104405) GET SOFT-SWR SETTING
	\$CKSWR	;; CALL=CKSWR	TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
	\$RDCHR	;; CALL=RDCHR	TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
	\$RDLIN	;; CALL=RDLIN	TRAP+10(104410) TTY TYPEIN STRING ROUTINE
	\$RDOCT	;; CALL=RDOCT	TRAP+11(104411) READ AN OCTAL NUMBER FROM TTY
	\$SAVREG	;; CALL=SAVREG	TRAP+12(104412) SAVE RO-R5 ROUTINE
	\$RESREG	;; CALL=RESREG	TRAP+13(104413) RESTORE RO-R5 ROUTINE



.SBTTL POWER DOWN AND UP ROUTINES

2534  
2535  
2536  
2537  
2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558  
2559  
2560  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578

011606 012737 011752 000024  
011614 012737 000340 000026  
011622 010046  
011624 010146  
011626 010246  
011630 010346  
011632 010446  
011634 010546  
011636 017746 167276  
011642 010667 000110  
011646 012737 011660 000024  
011654 000000  
011656 000776  
  
011660 012737 011752 000024  
011666 016706 000064  
011672 005067 000060  
011676 005267 000054  
011702 001375  
011704 012677 167230  
011710 012605  
011712 012604  
011714 012603  
011716 012602  
011720 012601  
011722 012600  
011724 012737 011606 000024  
011732 012737 000340 000026  
011740 104401  
011742 011760  
011744 012716  
011746 004412  
011750 000002  
011752 000000  
011754 000776  
011756 000000  
011760 005015 047520 042527  
011766 000122

\*\*\*\*\*

:POWER DOWN ROUTINE

```
$PWRDN: MOV $ILLUP, @PWRVEC ;; SET FOR FAST UP
MOV #340, @PWRVEC+2 ;; PRIO:7
MOV RO, -(SP) ;; PUSH RO ON STACK
MOV R1, -(SP) ;; PUSH R1 ON STACK
MOV R2, -(SP) ;; PUSH R2 ON STACK
MOV R3, -(SP) ;; PUSH R3 ON STACK
MOV R4, -(SP) ;; PUSH R4 ON STACK
MOV R5, -(SP) ;; PUSH R5 ON STACK
MOV @SWR, -(SP) ;; PUSH @SWR ON STACK
MOV SP, $SAVR6 ;; SAVE SP
MOV $PWRUP, @PWRVEC ;; SET UP VECTOR
HALT
BR .-2 ;; HANG UP
```

\*\*\*\*\*

:POWER UP ROUTINE

```
$PWRUP: MOV $ILLUP, @PWRVEC ;; SET FOR FAST DOWN
MOV $SAVR6, SP ;; GET SP
CLR $SAVR6 ;; WAIT LOOP FOR THE TTY
1$: INC $SAVR6 ;; WAIT FOR THE INC
BNE 1$ ;; OF WORD
MOV (SP)+, @SWR ;; POP STACK INTO @SWR
MOV (SP)+, R5 ;; POP STACK INTO R5
MOV (SP)+, R4 ;; POP STACK INTO R4
MOV (SP)+, R3 ;; POP STACK INTO R3
MOV (SP)+, R2 ;; POP STACK INTO R2
MOV (SP)+, R1 ;; POP STACK INTO R1
MOV (SP)+, RO ;; POP STACK INTO RO
MOV $PWRDN, @PWRVEC ;; SET UP THE POWER DOWN VECTOR
MOV #340, @PWRVEC+2 ;; PRIO:7
TYPE $POWER ;; REPORT THE POWER FAILURE
$PWRMG: .WORD $POWER ;; POWER FAIL MESSAGE POINTER
MOV (PC)+, (SP) ;; RESTART AT $EOP
$PWRAD: .WORD $EOP ;; RESTART ADDRESS
RTI
$ILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0 ;; PUT THE SP HERE
$POWER: .ASCIZ <15><12>"POWER"
.EVEN
```

2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613  
2614  
2615

;\*\*\*\*\*

;DATA POINTERS

011770	001116	001162	001166	DT1:	.WORD	\$ERRPC,\$REG0,\$REG2,BYTNUM,\$GDDAT,\$BDDAT,\$GDADR,\$BDADR,0
011776	001230	001124	001126			
012004	001120	001122	000000			
012012	001116	001162	001166	DT2:	.WORD	\$ERRPC,\$REG0,\$REG2,\$TMPO,\$TMP2,0
012020	001172	001176	000000			
012026	001116	001162	001166	DT101:	.WORD	\$ERRPC,\$REG0,\$REG2,\$TMPO,0
012034	001172	000000				
012040	001116	001240	000000	DT201:	.WORD	\$ERRPC,\$ACSL,0
012046	001116	000000		DT202:	.WORD	\$ERRPC,0

;DATA FORMATS

000000  
000001

DN=0 ;OCTAL NUMBER  
DN=1 ;DECIMAL NUMBER

012052	000	DF1:	.BYTE	ON	;\$ERRPC
012053	001		.BYTE	DN	;\$REG0
012054	001		.BYTE	DN	;\$REG2
012055	001		.BYTE	DN	;\$BYTNUM
012056	000		.BYTE	ON	;\$GDDAT
012057	000		.BYTE	ON	;\$BDDAT
012060	000		.BYTE	ON	;\$GDADR
012061	000		.BYTE	ON	;\$BDADR
012062	000	DF2:	.BYTE	ON	;\$ERRPC
012063	001		.BYTE	DN	;\$REG0
012064	001		.BYTE	DN	;\$REG2
012065	001		.BYTE	DN	;\$TMPO
012066	001		.BYTE	DN	;\$TMP1

E07

TAII DATA RELIABILITY MAINDEC-11-DZTAE-C  
DZTAECP11 17-MAR-77 14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 61  
ASCII MESSAGES

```

2616 ;*****
2617 ;
2618 ;MESSAGES
2619
2620 012067 127 044522 042524 MXWFG: .ASCIZ /WRITE-FILE-GAP/
2621 012074 043055 046111 026505
2622 012102 040507 000120
2623 012106 051127 052111 000105 MXWRIT: .ASCIZ /WRITE/
2624 012114 042522 042101 000 MXREAD: .ASCIZ /READ/
2625 012121 102 041501 026513 MXBSFG: .ASCIZ /BACK-SPACE-FILE-GAP/
2626 012126 050123 041501 026505
2627 012134 044506 042514 043455
2628 012142 050101 000
2629 012145 102 041501 026513 MXBSBG: .ASCIZ /BACK-SPACE-BLK-GAP/
2630 012152 050123 041501 026505
2631 012160 046102 026513 040507
2632 012166 000120
2633 012170 050123 041501 026505 MXSFFG: .ASCIZ /SPACE-FWD-FILE-GAP/
2634 012176 053506 026504 044506
2635 012204 042514 043455 050101
2636 012212 000
2637 012213 123 040520 042503 MXSFBG: .ASCIZ /SPACE-FWD-BLK-GAP/
2638 012220 043055 042127 041055
2639 012226 045514 043455 050101
2640 012234 000
2641 012235 122 053505 047111 MXRWND: .ASCIZ /REWIND/
2642 012242 000104
2643 012244 040504 040524 042440 EM1: .ASCIZ "DATA ERROR"
2644 012252 051122 051117 000
2645 012257 123 047131 020103 EM2: .ASCIZ "SYNC ERROR"
2646 012264 051105 047522 000122
2647 012272 051104 053111 020105 EM101: .ASCIZ "DRIVE IS OFF-LINE"
2648 012300 051511 047440 043106
2649 012306 046055 047111 000105
2650 012314 051104 053111 020105 EM102: .ASCIZ "DRIVE IS WRITE-LOCK"
2651 012322 051511 053440 044522
2652 012330 042524 046055 041517
2653 012336 000113
2654 012340 046103 040505 020122 EM103: .ASCIZ "CLEAR LEADER ERROR"
2655 012346 042514 042101 051105
2656 012354 042440 051122 051117
2657 012362 000
2658 012363 106 046111 020105 EM104: .ASCIZ "FILE GAP ERROR"
2659 012370 040507 020120 051105
2660 012376 047522 000122
2661 012402 044524 044515 043516 EM105: .ASCIZ "TIMING ERROR"
2662 012410 042440 051122 051117
2663 012416 000
2664 012417 102 047514 045503 EM106: .ASCIZ "BLOCK CHECK ERROR"
2665 012424 041440 042510 045503
2666 012432 042440 051122 051117
2667 012440 000
2668 012441 125 045516 047516 EM107: .ASCIZ "UNKNOWN INTERRUPT"
2669 012446 047127 044440 052116
2670 012454 051105 052522 052120
2671 012462 000

```



G07

TA11 DATA RELIABILITY  
DZTREC.P11 17-MAR-77

MAINDEC-11-DZTAE-C  
14:52

MACY11 27(1006) 17-MAR-77 14:55 PAGE 63  
ASCII MESSAGES

2728	013127	200	054502	042524	MSG5:	.ASCIZ	<CRLF>"BYTES WRITTEN="
2729	013134	020123	051127	052111			
2730	013142	042524	036516	000			
2731	013147	200	040524	042520	MSG6:	.ASCIZ	<CRLF>"TAPE PASSES="
2732	013154	050040	051501	042523			
2733	013162	036523	000				
2734	013165	200	044506	042514	MSG7:	.ASCIZ	<CRLF>"FILES/PASS="
2735	013172	027523	040520	051523			
2736	013200	000075					
2737	013202	041200	047514	045503	MSG8:	.ASCIZ	<CRLF>"BLOCKS/FILE="
2738	013210	027523	044506	042514			
2739	013216	000075					
2740	013220	040440	042116	000	MSG9:	.ASCIZ	" AND"
2741	013225	040	044527	046114	MSG10:	.ASCIZ	" WILL BE TESTED"<CRLF>
2742	013232	041040	020105	042524			
2743	013240	052123	042105	000200			
2744	013246	025052	020052	047506	MSG11:	.ASCIZ	"*** FORMAT ***"
2745	013254	046522	052101	025040			
2746	013262	025052	000				
2747	013265	052	025052	051040	MSG12:	.ASCIZ	"*** READ ***"
2748	013272	040505	020104	020040			
2749	013300	025052	000052				
2750	013304	025052	020052	051127	MSG13:	.ASCIZ	"*** WRITE ***"
2751	013312	052111	020105	025040			
2752	013320	025052	000				
2753	013323	040	000040		MSG14:	.ASCIZ	" "
2754	013326	042040	044522	042526	MSG15:	.ASCIZ	" DRIVE "
2755	013334	000040					
2756	013336	042200	044522	042526	MSGDRV:	.ASCIZ	<CRLF>"DRIVE(S)? "
2757	013344	051450	037451	000040			
2758	013352	052200	041501	037523	MSGASK:	.ASCIZ	<CRLF>"TACS?"
2759	013360	000					
2760	013361	200	042526	052103	MSGVEC:	.ASCIZ	<CRLF>"VECTOR?"
2761	013366	051117	000077				
2762	013372	050200	044522	051117	MSGPRI:	.ASCIZ	<CRLF>"PRIORITY?"
2763	013400	052111	037531	000			
2764	013405	200	040524	051503	MTACS:	.ASCIZ	<CRLF>"TACS="
2765	013412	000075					
2766	013414	052040	042101	036502	MTADB:	.ASCIZ	/ TADB=
2767	013422	000					
2768	013423	040	053040	041505	MTAVEC:	.ASCIZ	" VECTOR="
2769	013430	047524	036522	000			
2770	013435	040	050040	044522	MTAPRI:	.ASCIZ	" PRIORITY="
2771	013442	051117	052111	036531			
2772	013450	000					
2773	013451	200	045517	000077	MSGOK:	.ASCIZ	<CRLF>"OK?"
2774	013456	042531	100123	000	MYES:	.ASCIZ	"YES"<CRLF>
2775		013464			.EVEN		
2776							
2777					.SBTTL		READ AND WRITE BUFFER
2778							
2779	013464	002004			BUFFER:	.BLKB	1028.
2780		015470			LASTADDRESS=		.
2781							
2782		000001			.END		

















\$SETUP= 000137	537#	555	556	558	560	562	564	565	586	589	1219	1315	1350
	1369	1371	2279	2413									
\$STUP = 177777	537#												
\$SUPRS 007422	1243	1248	1253	1258	1263	1270	1275	2008#					
\$SVLAD 005030	1325	1330#											
\$SWR = 160000	2#	12	61	62	63	64	65	66	278	565	566	1212	1220
	1232	1298	1300	1309	1310	1311	1316	1328	1330	1332	1336#	1342	1343
	1344	1345	1346	1354	1361	1366	1370	1376	1377#	2572			
\$SWRMK= 000000	1311												
\$TKB 001146	261#	2277	2288	2305	2359	2365							
\$TKS 001144	260#	2277	2286	2302	2326*	2357	2363						
\$TMP0 001172	274#	774*	775*	788*	790	791	1487*	1526	1536	1560	1687*	2587	2590
\$TMP1 001174	275#												
\$TMP2 001176	276#	1688*	2587										
\$TMP3 001200	277#												
\$TN = 000001	2#	12											
\$TNPWR 007302	1938	1939	1959#										
\$TPB 001152	263#	2263*	2274										
\$TPFLG 001157	267#	2221	2274										
\$TPS 001150	262#	2261	2274										
\$TRAP 011522	560	2498#											
\$TRAP2 011544	2509#	2520											
\$TRP = 000014	2513#	2522#	2523#	2524#	2525#	2526	2527#	2528	2529#	2530#	2531#	2532#	2533#
	2534#												
\$TRPAD 011556	2503	2520#											
\$TSTNM 001102	240#	1219*	1308	1330*	1333	1336	1353	1376					
\$TTYIN 011230	2384	2385	2402	2406#									
\$TYPBN= ***** U	2525												
\$TYPDS= ***** U	2525												
\$TYPE 010300	2221#	2513	2521										
\$TYPEC 010450	2242	2249	2256	2261#	2262	2328							
\$TYPEX 010516	2267	2269	2272#										
\$TYPOC 011320	2443#	2522											
\$TYPON 011334	2442	2445#	2524										
\$TYPOS 011274	2438#	2523											
\$XTSTR 004772	1319#												
\$SET4= 000001	1232#	1290#											
\$OFILL 011517	2439#	2443*	2453	2488#									
\$4OCAT= ***** U	1316	1363											
= 015470	218#	222#	237#	281	553	565	600#	1300	1303	1336	1376	1377	1979#
	2274	2277	2406#	2407	2413	2550	2574	2775#	2779#	2780			

BGOTO	216#	1509													
CALL	216#	853	863	877	888	900	931	1003	1019	1030	1084	1131	1141	1155	
COMMEN	185#														
DECBLK	216#	1539													
DECFIL	216#	1528													
ENDCOM	185#														
EOPCOD	216#	1232													
ERROR	79#	645	708	913	920	1042	1049	1520	1523	1550	1553	1556	1559	1565	
ESCAPE	185#														
GETPRI	185#														
GETSWR	185#	589#													
INCBLK	216#	943	1069	1168											
INCFIL	216#	948	1075	1173											
JGOTO	216#	789													
MORETA	216#	282													
MULT	185#														
NEWTST	185#														
OUTDBL	216#	1251	1256												
OUTSGL	216#	1241	1246	1261	1273										
POP	185#	1628	1812	1888	1903	2054	2559	2560							
PUSH	185#	1611	1792	1876	2033	2540	2546								
REMARK	216#	850	860	872	885	896	908	915	926	940	1000	1010	1015	1026	1037
	1044	1056	1066	1081	1128	1138	1150	1165	1233	1278					
REPORT	185#														
SCOPE	80#	907	1055	1164	1200	1218	1232								
SETBLK	216#	844	951	994	1078	1122	1176								
SETFIL	216#	847	997	1125											
SETPRI	185#														
SETTRA	2513#	2522	2523	2524	2526	2528	2529	2530	2531	2532	2533				
SETUP	185#	548													
SKIP	185#														
SLASH	185#	318	339	361	415	527	715	799	803	850	852	860	862	872	874
	885	887	896	898	908	910	915	917	926	928	940	942	955	1000	1002
	1010	1012	1015	1017	1026	1028	1037	1039	1044	1046	1056	1058	1066	1068	1081
	1083	1092	1128	1130	1138	1140	1150	1152	1165	1167	1180	1233	1235	1278	1280
	1409	1429	1463	1472	1479										
SPACE	185#														
STARS	12	185#	229	233	281	441	463	511	529	535	602	611	619	635	654
	676	717	727	755	773	801	810	838	962	988	1099	1115	1185	1199	1209
	1305	1338	1377	1567	1590	1634	1653	1692	1741	1758	1772	1776	1821	1871	1917
	1920	1982	2000	2023	2059	2076	2112	2172	2206	2276	2279	2347	2376	2415	2492
	2536	2552	2579	2616											
SWRSU	185#	566#													
TRMTRP	2513#														
TYPBIN	185#														
TYPDEC	185#														
TYPNAM	185#	582													
TYPNUM	185#														
TYPOCS	185#														
TYPOCT	185#	1419	1431	1467	2149	2152	2155	2158	2297						
TYPTXT	185#														
\$\$CMRE	231#	270	271	272	273										
\$\$CMTM	231#	274	275	276	277										
\$\$ESCA	185#														
\$\$NEWT	185#														
\$\$SET	2513#	2522	2523	2524	2526	2528	2529	2530	2531	2532	2533				

\$\$\$SKIP	185#	
.EQUAT	2#	75
.HEADE	2#	
.SETUP	2#	537
.SWRHI	2#	57
.SWRLO	67#	
.SCATC	2#	216
.SCMTA	2#	231
.SDB2D	2#	1918
.SEOP	2#	1207
.SERRO	2#	1336
.SPOWE	2#	2534
.SRDOC	2#	2021
.SREAD	2#	2274
.\$SAVE	2#	1774
.\$SB2D	2#	1980
.\$SCOP	2#	1303
.\$SPAC	2#	
.\$SUPR	2#	1998
.\$TRAP	2#	2490
.\$TYPE	2#	2204
.\$TYPO	2#	2413

. ABS. 015470 000

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

DSKZ:DZTAECP11, DSKZ:DZTAECP11/CRF/SOL=DSKM:DZTAECP11  
RUN-TIME: 15 8 .9 SECONDS  
RUN-TIME RATIO: 108/25=4.3  
CORE USED: 25K (50 PAGES)