

DZV11

DIAG PRT1
CNDZAAO

AH-T444A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT © 82-83
MADE IN USA



Grid of diagnostic data tables (fiches) with various columns and rows of text.



.REM 8

IDENTIFICATION

PRODUCT CODE: AC-T443A-MC
PRODUCT NAME: CNDZAA0 DZV11 DIAG PRT1
PRODUCT DATE: DEC, 1982
MAINTAINER: DIAGNOSTIC SERVICES/ISS
AUTHOR: BRUCE LUHRS

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) 1982,1983 DIGITAL EQUIPMENT CORPORATION

1. ABSTRACT

THE FUNCTION OF THE DZV11 DIAGNOSTICS IS TO VERIFY THE OPTION OPERATES ACCORDING TO SPECIFICATIONS. THE DIAGNOSTICS ALSO VERIFY THAT THE DZV11 OPERATES IN ITS ENVIRONMENT SUCH AS THE SYSTEM IN WHICH IT IS INSTALLED.

PARAMETERS MAY BE SUPPLIED TO THE PROGRAM BY EITHER 'AUTO SIZING' OR INPUT FROM THE USER ON THE CONSOLE BY HAVING SW00=1 AT START TIME. AUTO SIZING WILL BE DONE ONLY THE FIRST TIME THE PROGRAM IS STARTED AND SW07=0 AND SW00=0 AND SW03=0. THE AUTOSIZER IS DESIGNED TO DETECT DZV11 DEVICE ADDRESSES AND VECTORS ONLY. ALL REMAINING PARAMETERS WILL DEFAULT TO CERTAIN VALUES (SEE SEC.8.5). CONSOLE INPUT MAY BE CONTROLLED AT ANY START TIME THROUGH THE USE OF SW00, SW03, SW04, AND SW06 (SEE SEC. 4.1.1 FOR A DETAILED DESCRIPTION OF THESE SWITCHES).

CURRENTLY THERE ARE THREE STANDALONE DIAGNOSTICS (CNDZA, CNDZB, AND CNDZC) ONE SYSTEM MODULE FOR DEC X/11 (CXDZBA), AND AN OVERLAY FOR ITEP (CVDZD).

CNDZA TOGETHER WITH CNDZB WILL TEST ALL LOGICAL FUNCTIONS OF THE DZV11 INTERFACE MODULE.

CNDZC IS DESIGNED AS A NON-CHAINABLE STANDALONE DIAGNOSTIC PROVIDING THE OPERATOR WITH DIRECT CONTROL OVER THE TESTING OF ALL DZV11 EIA CABLES.

* NOTE: THIS DIAGNOSTIC HAS BEEN MODIFIED TO RUN IN KXT11 (SBC 11/21) :GPA *
* BASED SYSTEMS. *
*

SBC 11/21

CSR RANGE: 174000 TO 177770
VECTOR RANGE: 300 TO 370
AUTO SIZING FOR
CSR AND VECTOR: DISABLED

2. REQUIREMENTS

2.1 EQUIPMENT

AN LSI11 CPU WITH MINIMUM 4K OF MEMORY.
ASR 33 (OR EQUIVALENT FOR CONSOLE)
DZV11 INTERFACE MODULE
H329 STAGGERED TURNAROUND CONNECTOR.
H325 CABLE TURNAROUND CONNECTOR.

NOTE: A STAGGERED TURNAROUND CONNECTOR IS NEEDED IN ORDER TO TEST THE PARITY LOGIC.

2.2 STORAGE

PROGRAM WILL USE ALL 4K OF MEMORY EXCEPT WHERE ABL AND BOOTSTRAP LOADER RESIDE. LOCATION 1500 THRU 1740 ARE ESPECIALLY TO BE NOTED AND TO BE UNTOUCHED BY OPERATOR AFTER PARAMETERS HAVE BEEN INPUT FROM CONSOLE (SW00=1); OR AFTER THE 'AUTO SIZING' HAS BEEN DONE. THESE LOCATIONS MAY BE CHANGED IF THE USER UNDERSTANDS THEIR MEANING AND DIFFERENT PARAMETERS ARE REQUIRED.

3. LOADING PROCEEDURE

3.1 METHOD

ALL PROGRAMS ARE IN ABSOLUTE FORMAT AND ARE LOADED USING THE ABSOLUTE LOADER. NOTE: IF THE DIAGNOSTICS ARE ON A MEDIA SUCH AS DISK, MAGTAPE, DECTAPE, OR CASSETTE; FOLLOW INSTRUCTIONS FOR THE MONITOR WHICH HAS BEEN PROVIDED ON THAT SPECIFIC MEDIA.

ABSOLUTE LOADER STARTING ADDRESS *500

MEMORY * SIZE

4K	17
8K	37
12K	57
16K	77
20K	117
24K	137
28K	157

3.1.1 STARTING THE PROCESSOR AT THE ABSOLUTE LOADER STARTING ADDRESS WILL LOAD THE DIAGNOSTIC INTO MEMORY.

4. STARTING PROCEEDURE

- A. SET SWR TO ZERO FOR 'AUTO SIZING' OR SET SW00=1 FOR USER PARAMETER INPUT FROM CONSOLE TERMINAL. NOTE: LOC. 000176 IS USED AS A SOFTWARE SWITCH REGISTER IN ALL OF THE DZV11 DIAGNOSTICS. (SEE SEC. 4.1) ON THE FIRST STARTUP OF THE DIAGNOSTIC IF SW07=1 AND SW00=0 THE PROGRAM WILL ASSUME THAT THE STATUS TABLE HAS BEEN ALREADY BUILT FROM A PREVIOUS DZV11 DIAGNOSTIC RUN. NOTE: ANY DZV11 DIAGNOSTIC WILL OVERLAY THE STATUS TABLE WHEN LOADED TO PRESERVE ITS CONTENTS AND THUS WILL NOT ALTER A PREVIOUSLY BUILT TABLE.
- B. START THE DIAGNOSTIC AT LOC. 200(8). THE PROGRAM WILL TYPE MAINDEC AND PROGRAM NAMES (IF THIS WAS THE FIRST START UP OF THE PROGRAM) AND ALSO THE FOLLOWING: (ON THE FIRST PROGRAM RUN OR IF PARAMETERS WERE CHANGED)

```
'MAP OF DZV11 STATUS'  
1500 160100  
1502 000300  
1504 000017  
1506 017470  
1510 000000
```

THE ABOVE IS ONLY AN EXAMPLE! THIS WOULD INDICATE THE STATUS TABLE STARTING AT ADD. 1500 IN THE PROGRAM. THE STATUS TABLE MUST BE VERIFIED BY THE USER IF AUTO SIZING IS DONE. FOR INFORMATION OF STATUS TABLE SEE SECTION 8.4 FOR HELP.

THE PROGRAM WILL TYPE 'RUNNING' AND PROCEED TO RUN THE DIAGNOSTIC.

4.1 CONTROL SWITCH SETTINGS

NOTE: THIS PROGRAM UTILIZES A SOFTWARE SWITCH REGISTER WHICH MAY BE MODIFIED BY CHANGING LOC. 176 OR BY TYPING CONTROL 'G' (^G) ON THE CONSOLE TERMINAL WHILE THE PROGRAM IS RUNNING.

```
SW 15 SET: HALT ON ERROR  
SW 14 SET: LOOP ON CURRENT TEST  
SW 13 SET: INHIBIT ERROR PRINT OUT  
SW 12 SET: INHIBIT **ALL** TYPE OUT/BELL ON ERROR.  
SW 11 SET: INHIBIT ITERATIONS. (QUICK PASS)  
SW 10 SET: ESCAPE TO NEXT TEST  
SW 09 SET: LOOP WITH CURRENT DATA  
SW 08 SET: CATCH ERROR AND LOOP ON IT  
SW 07 SET: NO AUTO SIZE. IF 1ST START OF PROGRAM AFTER LOADING AND  
IF SW00=0 THEN THE PROGRAM WILL ASSUME THAT THE STATUS MAP  
HAS BEEN BUILT FROM A PREVIOUS DZV11 DIAGNOSTIC RUN.  
  
SW 06 SET: RESELECT DZV11'S DESIRED ACTIVE  
SW 05 SET: RESERVED  
SW 04 SET: SELECT DELAY PARAMETER (SEE SEC. 4.1.1)  
SW 03 SET: EXTRA PARAMETER INPUT (SEE SEC. 4.1.1)  
SW 02 SET: LOCK ON SELECTED TEST  
SW 01 SET: RESTART PROGRAM AT SELECTED TEST  
SW 00 SET: GET USERS PARAMETERS FROM CONSOLE
```

4.1.1 SWITCH REGISTER CONTROL OF PARAMETER INPUT FROM CONSOLE

- SW 00 GET USERS PARAMETERS FROM CONSOLE. SETTING THIS SWITCH AT START UP TIME ALLOWS THE USER TO INPUT AT THE CONSOLE TERMINAL THE FOLLOWING PARAMETERS: BASE DEVICE ADDRESS, BASE VECTOR ADDRESS, MODE OF OPERATION (EXTERNAL, INTERNAL, OR STAGGERED), AND THE NUMBER OF DZV11'S THAT ARE RUNNING. USING THIS SWITCH ALONE WILL DEFAULT THE FOLLOWING PARAMETERS: ALL 4 LINES ARE SET TO BE TESTED ON EACH DZV11, THE DEFAULT BAUD RATE IS SET AT 19.2 KBAUD AND THE CHARACTER LENGTH FOR THE MAJORITY OF TESTING IS SET AT EIGHT BITS PER CHARACTER WITH TWO STOP BITS.
- SW 03 EXTRA PARAMETER INPUT. SETTING THIS SWITCH AT START UP TIME PROVIDES THE USER WITH THE ABILITY TO SET THE LINES ACTIVE FOR TESTING AND TO SET THE DEFAULT BAUD RATE USED FOR THE MAJORITY OF THE DIAGNOSTIC TESTS. THE DELAY PARAMETER IS AUTOMATICALLY ADJUSTED TO THE BAUD RATE GIVEN BY THE USER.
- SW 04 SELECT DELAY PARAMETER. THE DELAY PARAMETER THIS SWITCH CONTROLS DETERMINES THE LENGTH OF TIME THE PROGRAM STALLS WAITING FOR A CHARACTER TO BE COMPLETELY TRANSMITTED OR RECEIVED. THIS DELAY COUNT IS AUTOMATICALLY SET TO PROVIDE ENOUGH DELAY TIME FOR THE DEFAULT BAUD RATE SPECIFIED WHEN RUNNING THE PROGRAM ON AN LSI11 WITH MOS MEMORY. WHEN RUNNING THIS PROGRAM ON A PROCESSOR WITH A FASTER MEMORY SPEED THIS DELAY COUNT SHOULD BE ADJUSTED PROPORTIONATELY HIGHER THAN THE FOLLOWING DEFAULTED VALUES:
- | | | |
|------|-----------|------------|
| 2450 | :TIME FOR | 50 BAUD |
| 1560 | :TIME FOR | 75 BAUD |
| 1120 | :TIME FOR | 110 BAUD |
| 0750 | :TIME FOR | 134 BAUD |
| 0660 | :TIME FOR | 150 BAUD |
| 0330 | :TIME FOR | 300 BAUD |
| 0150 | :TIME FOR | 600 BAUD |
| 0060 | :TIME FOR | 1200 BAUD |
| 0040 | :TIME FOR | 1800 BAUD |
| 0030 | :TIME FOR | 2000 BAUD |
| 0020 | :TIME FOR | 2400 BAUD |
| 0010 | :TIME FOR | 3600 BAUD |
| 0001 | :TIME FOR | 4800 BAUD |
| 0001 | :TIME FOR | 7200 BAUD |
| 0001 | :TIME FOR | 9600 BAUD |
| 0001 | :TIME FOR | 19.2 KBAUD |

4.1.2 SWITCH REGISTER RESTRICTIONS

- SW 06 RESELECT DZV11'S DESIRED ACTIVE. A MESSAGE IS TYPED OUT ON THE CONSOLE TERMINAL ASKING THE OPERATOR TO TYPE A BIT MAP OF THE DZV'S DESIRED ACTIVE. USING THIS SWITCH ALLOWS LOCATION DZVACTV TO BE ALTERED (SEE SEC. 8.3 FOR A DESCRIPTION OF THIS LOCATION).
EXAMPLE:
IF THE DEVICES CORRESPONDING TO THE DZV11'S NUMBERED ZERO, TWO, AND FOUR IN THE DZV11 STATUS MAP (LOC. 1500 THROUGH 1740) ARE TO BE TESTED, TYPE IN: 25
THIS WILL SET BITS ZERO, TWO, AND FOUR IN LOCATION DZVACTV. ALL REMAINING DEVICES IN THE STATUS MAP WILL THEN NOT BE TESTED.
- SW 01 RESTART PROGRAM AT SELECTED TEST IT IS STRONGLY SUGGESTED THAT AT LEAST ONE PASS HAS BEEN MADE BEFORE TRYING TO SELECT A TEST THAT IS NOT IN THE ORDER OF SEQUENCE THE REASON BEING IS THAT THE PROGRAM HAS TO CLEAR AREAS AND SET UP PARAMETERS.
NOTE: IF RUNNING MULTIPLE DZV11'S, THE DZV11 YOU DESIRE TO BE UNDER TEST MUST BE SELECTED BY THE USE OF SW06 BEFORE LOCKING ON THE TEST. IN OTHER WORDS; EACH TIME THE PROGRAM IS STARTED; THE FIRST DZV11 WILL BE SELECTED TO BE UNDER TEST UNLESS SW06 IS USED TO SELECT ONLY ONE.
- SW 09 LOOP ON CURRENT DATA: THIS SWITCH WILL ONLY WORK IF CALL 'SCOPI' IS IN THAT TEST. THE REASON BEING THAT MOST TESTS DEAL WITH BLOCKS OF DIFFERENT DATA TO BE SENT OR RECEIVED ALL AT ONCE THUS IN BLOCK DATA, ONE PATTERN CAN'T BE SINGLED OUT.
THIS SWITCH IS DESIGNED TO PROVIDE AN AID FOR A TRAINED TROUBLE-SHOOTER TO SAMPLE VARIOUS SIGNALS ON THE MODULE AND IS NOT MEANT TO BE USED AS A GENERAL USER CONTROL SWITCH.
- SW 04 SELECT DELAY PARAMETER: THIS SWITCH SHOULD BE USED WITH CARE AS TOO SHORT A DELAY WILL CAUSE VALID TESTS TO FAIL.
(SEE SEC. 4.1.1)

4.1.3 SWITCH REGISTER PRIORITIES

ERROR SWITCHES

1. SW 12 DELETE PRINT OUT/BELL ON ERROR.
2. SW 13 DELETE ERROR PRINTOUT.
3. SW 15 HALT ON THE ERROR.
4. SW 08 GO TO BEGINNING OF THE TEST(ON ERROR).
5. SW 10 GOTO NEXT TEST(ON ERROR).

SCOPE SWITCHES

1. SW 09 (IF ENABLED BY 'SCOPI'). IF AN '*' IS PRINTED IN FRONT OF THE TEST NO. ON AN ERROR REPORT (EX. *TEST NO. 10) SW09 IS INCORPORATED IN THAT TEST AND THEREFORE SW09 IS *USUALLY* THE BEST SWITCH FOR THE SCOPE LOOP (SW14=0, SW10=0, SW09=1, SW08=0) IF THE PROGRAM USER IS TECHNICALLY TRAINED TO ELECTRONICALLY ISOLATE SIGNAL PROBLEMS ON THE DZV11 MODULE. IF SW09 IS NOT ENABELED; AND THERE IS A *HARD* ERROR (CONSTANT); SW08 IS BEST.
2. FOR INTERMITTENT ERRORS EITHER START THE PROGRAM WITH SW01 AND SW02 SET WHICH WILL ALLOW THE USER TO LOCK ON A SELECTED TEST, OR ELSE SET SW14 AS AN ERROR IS BEING TYPED OUT ON THE TERMINAL. SW14 WILL CONTINUE TO LOOP ON THAT TEST REGARDLESS OF WHETHER AN ERROR OCCURS.
3. SW 14 LOOP ON CURRENT TEST.

4.2 STARTING ADDRESS

SA 200 - THE STARTING ADDRESS FOR ANY DZV11 DIAGNOSTIC IS LOC. 200

NOTE: IF ADDRESS 000042 IS NON-ZERO THE PROGRAM ASSUMES IT IS UNDER ACT11 OR XXDP CONTROL AND WILL ACT ACCORDINGLY. AFTER *ALL* AVAILABLE DZV11S ARE TESTED THE PROGRAM WILL RETURN TO 'XXDP' OR 'ACT-11'.

5. OPERATING PROCEEDURE

WHEN THE PROGRAM IS INITIALLY STARTED, MESSAGES AS DESCRIBED IN SECTION FOUR WILL BE PRINTED AND THE DIAGNOSTIC WILL BEGIN RUNNING.

5.1 NORMAL START OF DIAGNOSTIC

ON THE FIRST START OF THE DIAGNOSTIC AT ADDRESS 200, IF SW00=1 THEN THE FOLLOWING QUESTIONS ARE ASKED AND MUST BE ANSWERED:

'1ST CSR ADDRESS (160000:163770): ''
YOU MUST TYPE IN THE FIRST DZV11 CSR IN THE SYSTEM YOU WISH TESTING TO BEGIN AT. RANGE: 160000:163770

'1ST VECTOR ADDRESS (300:770): ''
YOU MUST TYPE IN THE VECTOR OF THE FIRST DZV11 IN THE SYSTEM UNDER TEST. RANGE 300:770

'MAINTENANCE MODE
[EXTERNAL <H325> (E)]
[INTERNAL <DZCSR03=1>(I)]
[STAGGERED <H329> (S)] :
TYPE 'E' OR 'I' OR 'S' DEPENDING ON WHICH MODE YOU WISH TO RUN IN. IF RUNNING 'EXTERNAL': ALL SELECTED LINES MUST BE TERMINATED BY AN H325 TEST CONNECTOR.

'# OF DZV11'S <IN OCTAL> (1:20): ''
TYPE TOTAL NUMBER OF DZV11'S TO BE TESTED IN THE SYSTEM. RANGE IS 1 THRU 20 IN OCTAL.

***** IF SW03=1 THEN THE FOLLOWING WILL BE PRINTED *****

'LINES ACTIVE BY BIT <IN OCTAL> (001:017):''
EACH BIT REPRESENTS A LINE AND ANY COMBINATION OF LINES MAY BE SELECTED (HOWEVER IN STAGGERED MODE TWO ADJACENT LINES MUST BE SELECTED (0-1, 2-3).

'DEFAULT BAUD RATE <IN OCTAL> (00:17): ''
THIS GIVES THE USER A CHANCE TO CHANGE THE DEFAULT BAUD RATE USED IN APP. 90% OF THE TEST. BAUD RATE CHOICES ARE:
'00'(50 BAUD), '01'(75 BAUD), '02'(110 BAUD), '03'(134 BAUD),
'04'(150 BAUD), '05'(300 BAUD), '06'(600 BAUD), '07'(1200 BAUD),
'10'(1800 BAUD), '11'(2000 BAUD), '12'(2400 BAUD), '13'(3600 BAUD),
'14'(4800 BAUD), '15'(7200 BAUD), '16'(9600 BAUD), '17'(19.2 KBAUD)
LOW DEFAULT BAUD RATES ARE NOT SUGGESTED SINCE THEY LENGTHEN THE TIME TO COMPLETE A PROGRAM PASS DRAMATICALLY.

IT IS IMPORTANT TO NOTE THAT ALL DZV11'S IN THE SYSTEM MUST BE CONTIGIOUS FOR BOTH ADDRESS AND VECTORS. ALSO ALL THE EXTRA PARAMETERS OTHER THAN CSR AND VECTORS ARE GIVEN TO THE EXISTING DZV11'S IN THE SYSTEM.

IF THE MODE OF OPERATION IS DIFFERENT FOR EACH DZV11 THIS MUST BE PATCHED INTO THE CORRECT STATUS MAP ENTRY WHICH IS PRINTED AT START TIME. AN ALTERNATIVE IS TO PUT SW00=1 AT START TIME; ANSWER QUESTIONS ABOUT DZV11 UNDER TEST AND INDICATE ONE DZV11 IN THE SYSTEM. IF THE STATUS MAP IS TO BE 'PATCHED' IT MUST BE DONE AFTER THE QUESTIONS ARE ANSWERED OR AFTER THE AUTO SIZE.

5.2 PROGRAM AND/OR OPERATOR ACTION

THE VARIETY OF PROGRAM CONTROL SWITCHES PROVIDED IN THIS DIAGNOSTIC PACKAGE IS DESIGNED TO PROVIDE THE USER WITH A WIDE RANGE OF TROUBLE-SHOOTING TECHNIQUES. BEFORE THE USER ATTEMPTS TO RUN THIS DIAGNOSTIC HE SHOULD BECOME FAMILIAR WITH THE USE OF THESE CONTROL SWITCHES AND THEIR RESTRICTIONS. (SEE SEC. 4.1, 4.1.1, 4.1.2, 4.1.3)

WHEN THE PROGRAM DETECTS AN ERROR THE TEST NUMBER AND PC WILL BE TYPED OUT AND POSSIBLY AN ERROR MESSAGE (DEPENDING ON THE PARTICULAR ERROR). IF IT IS NECESSARY TO KNOW MORE INFORMATION CONCERNING THE ERROR REPORT THEN LOOK IN THE PROGRAM LISTING FOR THAT TEST NUMBER AND THEN NOTE THE PC OF THE ERROR REPORT. THE REASON FOR THE ERROR REPORT WILL BECOME CLEARER WHEN READING THE COMMENTS IN THE PROGRAM LISTING.

6. ERRORS

AS DESCRIBED PREVIOUSLY THERE WILL ALWAYS BE A TEST NUMBER AND PC TYPED OUT AT THE TIME OF AN ERROR (PROVIDING SW 13=0 AND SW 12=0). IN MOST CASES ADDITIONAL INFORMATION WILL BE SUPPLIED TO THE THE ERROR MESSAGE WHICH IS TO GIVE THE OPERATOR AN INDICATION OF THE ERROR.

6.1 ERROR RECOVERY

IF FOR SOME REASON THE DZV11 SHOULD 'HANG THE BUS' (GAIN CONTROL OF BUS SO THAT CONSOLE MANUAL FUNCTIONS ARE INHIBITED) AN INIT OR POWER DOWN/UP IS NECESSARY FOR OPERATOR TO REGAIN CONTROL OF CPU. IF THIS SHOULD HAPPEN, LOOK IN LOCATION '\$STNM' (ADDRESS 1246) FOR THE NUMBER OF THE TEST THAT WAS RUNNING AT THE TIME OF THE CATASTROPHIC ERROR. IN THIS WAY THE OPERATOR WILL HAVE AN IDEA AS TO WHAT THE DZV11 WAS DOING AT THE TIME OF THE ERROR.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

SEE SECTION 4.1.2
THE STATUS TABLE SHOULD BE VERIFIED REGARDLESS OF HOW THE PROGRAM WAS STARTED. ALSO IT IS IMPORTANT TO USE THIS LISTING ALONG WITH THE INFORMATION PRINTED ON THE TTY TO COMPLETELY ISOLATE PROBLEMS.

7.2 OPERATING RESTRICTIONS

PARAMETER MUST BE INPUT FROM USER OR APT IF "AUTO SIZING" IS NOT USED.

8. MISCELLANEOUS

8.1 EXECUTION TIME

ALL DZV11 DEVICE DIAGNOSTICS WILL GIVE AN 'END PASS' MESSAGE (PROVIDING NO ERRORS AND SW12=0) WITHIN 2 MIN. THIS IS ASSUMING SW11=1 (INHIBIT ITERATIONS) IS SET TO GIVE THE FASTEST POSSIBLE EXECUTION.

8.2 PASS COMPLETE

NOTE: *EVERY* TIME THE PROGRAM IS STARTED; THE TESTS WILL RUN AS IF SW11 (DELETE ITERATIONS) WAS UP (=1). THIS IS TO 'VERIFY NO *HARD* ERRORS' AS SOON AS POSSIBLE. THEREFORE THE FIRST PASS -EACH TIME PROGRAM IS STARTED- WILL BE A 'QUICK PASS' UNTIL ALL DZV11'S IN SYSTEM ARE TESTED. WHEN THE DIAGNOSTIC HAS COMPLETED A PASS THE FOLLOWING IS AN EXAMPLE OF THE PRINT OUT TO BE EXPECTED.

END PASS CNDZA-A CSR: 160100 VEC: 300 PASSES: 000001 ERRORS: 000000

NOTE: THE NUMBERS FOR CSR AND VEC ARE NOT NECESSARILY THE VALUES FOR THE DEVICE. THEY ARE ONLY FOR THIS EXAMPLE.

8.3 KEY LOCATIONS

\$LPADR (1252) CONTAINS THE ADDRESS WHERE PROGRAM WILL RETURN WHEN ITERATION COUNT IS REACHED OR IF LOOP ON TEST IS ASSERTED.

NEXT (1362) CONTAINS THE ADDRESS OF THE NEXT TEST TO BE PERFORMED.

\$TSTNM (1246) CONTAINS THE NUMBER OF THE TEST NOW BEING PERFORMED.

RUN (1412) THE BIT IN 'RUN' ALWAYS POINTS ONE PAST THE DZV11 CURRENTLY BEING TESTED. EXAMPLE: (RUN) 1412/0000000001000000 MEANS THAT DZV11 NO.5 IS THE DZV11 NOW RUNNING.

STATUS MAP (1500)-(1740) THESE LOCATIONS CONTAIN THE INFORMATION NEEDED TO TEST UP TO 16 (DECIMAL) DZV11S SEQUENTIALY. THEY CONTAIN THE CSR,VECTOR AND STATUS CONCERNING THE CONFIGURATION OF EACH DZV11.

DZVACTV(1406) EACH BIT SET IN THIS LOCATION INDICATES THAT THE ASSOCIATED DZV11 WILL BE TESTED IN TURN. EXAMPLE: (DZVACTV) 1406/0000000000011111 MEANS THAT DZV11 NO. 00,01,02,03,04 WILL BE TESTED. EXAMPLE: (DZVACTV) 1406/0000000000010001 MEANS THAT DZV11 NO. 00,04 WILL BE TESTED.

\$BASE (1174) CONTAINS THE RECEIVER CSR OF THE CURRENT DZV11 UNDER TEST.

8.4 MORE ON THAT 'STATUS TABLE' (1500-1740)

'MAP OF DZV11 STATUS'
1500 160100
1502 000300
1504 000017
1506 017470
1510 000000

THE ABOVE INFORMATION WILL BE REPEATED FOR EACH OF UP TO 16 DZV11'S IN THE SYSTEM (THESE WILL FOLLOW UNDER THIS TABLE). EXPLANATION:

1500 160100 THIS IS THE SYSTEM CONTROL REGISTER FOR THE 1ST DZV11 IN THE SYSTEM.
1502 000300 THIS IS VECTOR 'A' FOR THE FIRST DZV11 IN THE SYSTEM.
1504 000017 THIS IS THE BINARY REPRESENTATION OF WHAT LINES ARE TO BE TESTED.
1506 017470 THIS IS THE PARAMETER LOCATION USED IN MOST OF THE TESTS. IT INDICATES PARAMETERS OF: RX ON, SPEED SELECT 17 (19.2K BAUD) EIGHT BITS PER CHAR, AND TWO STOP BITS. THE USER MAY ALTER THE STOP BITS AND THE SPEED, BUT THE REMAINING PARAMETERS SHOULD BE LEFT ALONE. THIS LOCATION IS USED TO LOAD THE DZV11 LINE PARAMETER REGISTER FOR EACH LINE. THE MEANING OF THE BITS SET IN THIS LOCATION IS THE SAME AS THE FUNCTION OF THE RELATED BITS IN THE DEVICE LINE PARAMETER REGISTER.
1510 000000 THIS LOCATION WILL CONTAIN EITHER ALL ZEROS INDICATING THAT INTERNAL LOOP WAS SELECTED AS MODE OF OPERATION OR IT WILL CONTAIN 100000 INDICATING THAT "STAGGERED MODE" WAS SELECTED OR IT WILL CONTAIN 000200 INDICATING THAT "EXTERNAL" WAS THE MODE SELECTED.

THE ABOVE IS REPEATED FOR EACH DZV11 IN THE SYSTEM. THE TABLE IS FILLED BY AUTO SIZING OR BY THE MANUAL PARAMETER INPUT PROGRAM AS DESCRIBED PREVIOUSLY. ALSO IF DESIRED BY USER; THE LOCATIONS MAY BE ALTERED BY HAND TO SUIT THE SPECIFIC CONFIGURATION.

8.5 *** METHOD OF AUTO SIZING ***

8.5.1 FINDING THE CONTROL STATUS REGISTER.

THE PROGRAM WILL START AT ADDRESS 160000 AND START 'REFERENCING' THE ADDRESS IN THE POINTER. IF A NON-EX MEMORY TRAP OCCURES, THE POINTER (HOLDING 160000) IS UPDATED BY 10 AND THE ABOVE IS REPEATED UNTIL ADDRESS 163770 IS REACHED. IF A 'BUS REPLY' RESPONSE WAS ISSUED BY THE DZV11 (OR ANY OTHER DEVICE) (NO NXM TRAP), 'MASTER SCAN ENABLE' IS ATTEMPTED TO BE SET AND THE TCR BITS FOR ALL FOUR LINES ARE SET. 'TRDY' IS THEN TESTED TO BE SET AND 'MASTER SCAN ENABLE' IS TESTED TO BE STILL SET. THE DIAGNOSTIC WILL THEN CHECK THAT AT LEAST ONE TCR BIT IS STILL SET. IF ALL OF THE ABOVE WORKED, THIS DEVICE IS ASSUMED TO BE A DZV11. IF ANY OF THE ABOVE FAILED, UPDATING OF THE POINTER IS DONE AND THE SEQUENCE IS REPEATED.

NOTE: IF THE PROGRAM DOES NOT FIND YOUR DZV11, SOMETHING IS WRONG AND AUTO SIZING SHOULD NOT BE DONE.

8.5.2 FINDING THE VECTOR

THE VECTOR AREA (ADDRESS 300-776) IS FILLED WITH THE INSTRUCTION IOT AND '+2' (NEXT ADDRESS). BIT14 AND BITS (TX INTERRUPT ENABLE AND MSTSCAN ENABLE) ARE SET INTO THE DZVCSR. ALL TCR BITS ARE SET, A DELAY OCCURS, AND IF NO INTERRUPT OCCURES (BECAUSE OF A BAD DZV11) THE PROGRAM ASSUMES VECTOR ADDRESS 300 AND THE PROBLEM SHOULD BE FIXED IN THE DIAGNOSTIC. ONCE THE PROBLEM IS FIXED, THE PROGRAM SHOULD BE SETUP AGAIN TO SET THE CORRECT VECTOR. IF AN INTERRUPT OCCURRED, THE ADDRESS TO WHICH THE DZV11 INTERRUPTED TO IS PICKED UP AND REPORTED AS THE VECTOR. NOTE: IF THE VECTOR REPORTED IS NOT THE VECTOR SET UP BY YOU, THERE IS A PROBLEM AND AUTO SIZING SHOULD NOT BE DONE.

8.5.3 PARAMETER ASSUMPTIONS.

SINCE TOO MUCH HARDWARE WOULD NEED TO BE TURNED ON TO SIZE THE REST OF THE PARAMETERS; THE PROGRAM MUST ASSUME THE REMAINING VARIATIONS. THE RESULT IF NOT TO YOUR SPECIFIC CONFIGURATION MAY BE ALTERED BY HAND. IN THIS WAY 95% OF THE PARAMETER SETUP WAS DONE BY THE PROGRAM AND 5% BY YOU.

THEREFORE:

- 1) ALL FOUR LINES ARE ASSUMED TO BE TESTED.
- 2) DEFAULT BAUD RATE IS SET TO 17 (19.2 KBAUD).
- 3) MODE OF OPERATION IS "INTERNAL MODE".

FOR ALL PARAMETER ADJUSTMENTS PLEASE REFER TO SECTION 8.4 FOR GREATER DETAIL.

9.0 RUNNING THE DZV11 DIAGNOSTIC UNDER APT

9.1.1 THE APT INTERFACE

THE DZV DIAGNOSTICS HAVE BEEN DESIGNED TO BE COMPATIBLE WITH THE APT (AUTOMATED PRODUCT TEST) SYSTEM. THE DZV LOGIC TEST DIAGNOSTICS (CVDZA, AND CVDZB) CAN BE RUN AS STANDALONE DIAGNOSTICS OR IN EITHER OF THE APT MODES. CVDZC, HOWEVER IS DESIGNED AS A STANDALONE DIAGNOSTIC ONLY AND REQUIRES DIRECT OPERATOR PARTICIPATION.

9.1.2 SETTING UP THE DIAGNOSTIC USING APT

THE DIAGNOSTIC USES SEVERAL VARIABLES IN THE REGION SUBTITLED " APT MAILBOX-ETABLE". THESE VARIABLES ARE:

\$SWREG -(1142) USED AS THE SOFTWARE SWITCH REGISTER WHILE RUNNING UNDER APT.

\$VECT1 -(1170) USED TO SPECIFY THE FIRST VECTOR ADDRESS

\$BASE -(1174) USED TO INDICATE BOTTOM ADDRESS OF DZV11 UNDER TEST

\$DEVM -(1176) A BIT MAP REPRESENTING WHICH DZV11'S WILL BE TESTED

\$CDW1 -(1200) USED TO INDICATE WHICH LINES TO RUN ON ALL DZV11'S

\$CDW2 -(1202) USED TO INDICATE THE DEFAULT TEST MODE. SET TO 0 FOR INTERNAL TESTING, 200 FOR EXTERNAL LOOP BACK (H325 INSTALLED), OR SET TO 100000 FOR STAGGERED LOOP BACK TESTING (H329 INSTALLED).

\$DDWO -(1204) EACH OF THE \$DDW WORDS DESCRIBES THE PARAMETERS (LPR) FOR A PARTICULAR DZV11, GOING UP TO 16 DZV11'S

9.1.3 RUNNING UNDER APT

ALL OF THE VARIABLES MENTIONED IN SECTION 9.1.2 SHOULD BE SET UP PRIOR TO RUNNING THE DIAGNOSTIC UNDER APT.

NOTE

BE SURE \$BASE POINTS TO THE FIRST DZV11 BEFORE RUNNING

BASED ON THESE VALUES, THE DIAGNOSTIC WILL SET UP THE STATUS TABLE. THE USER IS THEN FREE TO MONITOR UNDER APT AS NORMAL.

10.0 PROGRAM DESCRIPTION

THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC PACKAGE (MAINDEC-11-DZQAC-C3).

INITIAL ADDRESS OF THE STACK POINTER **1120**
MISCELLANEOUS DEFINITIONS

GENERAL PURPOSE REGISTER DEFINITIONS

PRIORITY LEVEL DEFINITIONS

"SWITCH REGISTER" SWITCH DEFINITIONS

DATA BIT DEFINITIONS (BIT00 TO BIT15)

BASIC "CPU" TRAP VECTOR ADDRESSES

BITS 15-11=CPU TYPE
11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
11/70=06,PDQ=07,Q=10
BIT 10=REAL TIME CLOCK
BIT 9=FLOATING POINT PROCESSOR
BIT 8=MEMORY MANAGEMENT

MEM.TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003

MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABO

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

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

INCREMENT THE PASS NUMBER (\$PASS)
IF THERES A MONITOR GO TO IT
IF THERE ISN'T JUMP TO CYCLE

THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>
THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:

SW14=1 LOOP ON TEST
SW11=1 INHIBIT ITERATIONS

CALL SCOPE ;:SCOPE=IOT

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

ROUTINE USED TO SET UP THE DIAGNOSTIC VIA APT.
IF BIT7 IN THE ENVIRONMENT MODE (\$ENVM) BYTE IS SET,
THE PROGRAM WILL LOAD ITS PARAMETERS FROM THE ETABLE.

ROUTINE USED TO "AUTO SIZE" THE DZV11
CSR AND VECTOR.

NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
ADDRESS RANGE (160000:163770)
AND THE VECTOR MAY BE ANY WHERE IN THE
FLOATING VECTOR RANGE (300:770)

***** TEST 1 *****
THIS TEST PROVES THE BUS REPLY RESPONSE
DURING A READ OR WRITE TO THE FOLLOWING ADDRESS:
DZVCSR, DZVRBUF, DZVTOR, DZVMSR

***** TEST 2 *****
THIS TEST PROVES THAT BIT 'DCLR'
CAN BE SET AND THAT IT WILL CLEAR
BY ITSELF

***** TEST 3 *****

TEST TO VERIFY THAT THE R/W BITS OF THE
DZVCSR REGISTER CAN BE SET. THEN VERIFY THAT
THESE BITS CAN BE CLEARED. AND FINALLY, VERIFY
THAT AFTER BEING SET AGAIN THEY CAN BE
CLEARED BY A 'DEVICE CLEAR'.
THE BITS TESTED ARE: MAINT, MSENAB, SILOEN,
RIE, AND TIE.

***** TEST 4 *****

THIS TESTS THAT ALL OF THE TCR BITS
CAN BE: SET, CLEARED, AND CLEARED BY A DEVICE CLEAR.
THIS TEST ALSO DETERMINES IF THE DTR BITS CAN
BE SET, CLEARED, AND CLEARED BY A RESET.

***** TEST 5 *****

THIS TEST VERIFIES THAT
BITS 'RDONE,TRDY, BIT9, BIT8,
AND SILOAL' ARE READ ONLY AND THAT TRDY IS
ZERO UNTIL A LINE IS SELECTED AND MSENAB IS SET.

***** TEST 6 *****

THIS TEST VERIFIES THAT:
TIE,SILOEN,RIE,MSENAB,AND MAINT ARE THE
ONLY R/W BITS IN THE DZVCSR AND THAT
SETTING 'DCLR' IN THE CSR WILL CLEAR THESE BITS.

***** TEST 7 *****

THIS TEST PERFORMS RESET TESTING AND
TESTING OF READ ONLY REGISTER DZVRBUF
AND TESTING OF WRITE ONLY REGISTER DZVLPR

***** TEST 10 *****

THIS TEST PERFORMS RESET TESTING AND
TESTING OF READ ONLY REGISTER DZVMSR
AND TESTING OF WRITE ONLY REGISTER DZVTDR

***** TEST 11 *****

VERIFY THAT SETTING 'DTR' FOR A LINE WILL
BRING UP 'CO' AND 'RING' FOR:
THE SAME LINE IF IN EXTERNAL MODE
THE STAGGERED LINE IF IN STAGGERED MODE.
LINES ARE STAGGERED AS FOLLOWS:
LINE0 WITH LINE1; LINE2 WITH LINE3.
THIS TEST IS ONLY RUN IF AN H325,OR H329
IS CONNECTED ON THE DZV UNDER TEST.

***** TEST 12 *****

THIS TEST VERIFIES THAT TRDY IS SET WHEN A LINE
IS READY TO BE LOADED, AND THAT THE LINE SPECI-
FIED IN BITS 8-9 OF DZVCSR CORRESPOND
TO THE LINE SELECTED IN DZVTCR

***** TEST 13 *****
TEST TO TRANSMIT ONE CHAR AND
RECEIVE ONE CHAR ON ONE LINE
AT A TIME. THE CHAR IS '252' AND
ALL SELECTED LINES WILL BE TURNED ON .

THIS IS THE FIRST TIME ANY
DATA IS CHECKED IN THE RECEIVER.
USING SWITCH NINE WITH THIS TEST CREATES A TIGHT SCOPE LOOP
WHICH TRANSMITS A STEADY STREAM OF CHARACTERS.

***** TEST 14 *****
THIS TEST VERIFIES THAT EACH RECEIVING LINE CAN BE
DISABLED BY SETTING RCVON (BIT12 IN THE LPR REGISTER)
TO ZERO FOR EACH LINE.
THIS TEST ALSO VERIFIES THAT THE SILO CAN BE
EMPTIED BY ISSUING A DEVICE MASTER CLEAR.

***** TEST 15 *****
THIS TEST PROVES THAT THE TRANSMITTER TRANSMITS
CHARACTERS (FLAG MODE) AND THE RECEIVER RECEIVES (FLAG MODE)
(ONE LINE AT A TIME BASED UPON VALID LINES)
THIS IS THE FIRST TIME THAT ALL DATA IS CHECKED

***** TEST 16 *****
THIS TEST WILL PROVE THAT:
1) THE TRANSMITTER 'BREAK BIT' WORKS
2) THE RECEIVER CAN FLAG 'FRAMING ERRORS'
3) THE RECEIVER CAN FLAG 'PARITY ERRORS'
ONLY ONE LINE AT A TIME WILL BE EXERCISED.

***** TEST 17 *****
THIS TEST VERIFIES THAT THE DEVICE DOES NOT INTERRUPT
WHILE THE PROCESSOR STATUS DOES NOT ALLOW INTERRUPTS
BUT WILL INTERRUPT IF THE PROCESSOR STATUS
ALLOWS INTERRUPTS.

***** TEST 20 *****
THIS TEST VERIFIES THAT THE RECEIVER WILL
INTERRUPT BEFORE THE TRANSMITTER EVEN
THOUGH THE TRANSMITTER WAS ENABLED
FIRST. SET PS TO HIGH (MASK INTERRUPTS);
GET RDONE AND TRDY TO SET;
SET TX IE AND RX IE;
CLEAR PS AND EXPECT RX TO INTERRUPT FIRST


```
(2) 000200 CRLF= 200 ::CODE FOR CARRIAGE RETURN-LINE FEED
(2) 177776 PS= 177776 ::PROCESSOR STATUS WORD
(2) .EQUIV PS,PSW
(2) 177774 STKLMT= 177774 ::STACK LIMIT REGISTER
(2) 177772 PIRQ= 177772 ::PROGRAM INTERRUPT REQUEST REGISTER
(2) 177570 DSWR= 177570 ::HARDWARE SWITCH REGISTER
(2) 177570 DDISP= 177570 ::HARDWARE DISPLAY REGISTER
(2) :***** THE FOLLOWING ODT START ADDRESS FOR SBC 11/21 IS ADDED
(2) 170000 ODTST= 170000
(2) :*GENERAL PURPOSE REGISTER DEFINITIONS
(2) 000000 R0= X0 ::GENERAL REGISTER
(2) 000001 R1= X1 ::GENERAL REGISTER
(2) 000002 R2= X2 ::GENERAL REGISTER
(2) 000003 R3= X3 ::GENERAL REGISTER
(2) 000004 R4= X4 ::GENERAL REGISTER
(2) 000005 R5= X5 ::GENERAL REGISTER
(2) 000006 R6= X6 ::GENERAL REGISTER
(2) 000007 R7= X7 ::GENERAL REGISTER
(2) 000006 SP= X6 ::STACK POINTER
(2) 000007 PC= X7 ::PROGRAM COUNTER
(2) :*PRIORITY LEVEL DEFINITIONS
(2) 000000 PR0= 0 ::PRIORITY LEVEL 0
(2) 000040 PR1= 40 ::PRIORITY LEVEL 1
(2) 000100 PR2= 100 ::PRIORITY LEVEL 2
(2) 000140 PR3= 140 ::PRIORITY LEVEL 3
(2) 000200 PR4= 200 ::PRIORITY LEVEL 4
(2) 000240 PR5= 240 ::PRIORITY LEVEL 5
(2) 000300 PR6= 300 ::PRIORITY LEVEL 6
(2) 000340 PR7= 340 ::PRIORITY LEVEL 7
(2) :*'SWITCH REGISTER' SWITCH DEFINITIONS
(2) 100000 SW15= 100000
(2) 040000 SW14= 40000
(2) 020000 SW13= 20000
(2) 010000 SW12= 10000
(2) 004000 SW11= 4000
(2) 002000 SW10= 2000
(2) 001000 SW09= 1000
(2) 000400 SW08= 400
(2) 000200 SW07= 200
(2) 000100 SW06= 100
(2) 000040 SW05= 40
(2) 000020 SW04= 20
(2) 000010 SW03= 10
(2) 000004 SW02= 4
(2) 000002 SW01= 2
(2) 000001 SW00= 1
(2) .EQUIV SW09,SW9
(2) .EQUIV SW08,SW8
(2) .EQUIV SW07,SW7
(2) .EQUIV SW06,SW6
(2) .EQUIV SW05,SW5
(2) .EQUIV SW04,SW4
(2) .EQUIV SW03,SW3
(2) .EQUIV SW02,SW2
```

```
(2) .EQUIV SW01,SW1
(2) .EQUIV SW00,SW0
(2)
(2) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(2) 100000 BIT15= 100000
(2) 040000 BIT14= 40000
(2) 020000 BIT13= 20000
(2) 010000 BIT12= 10000
(2) 004000 BIT11= 4000
(2) 002000 BIT10= 2000
(2) 001000 BIT09= 1000
(2) 000400 BIT08= 400
(2) 000200 BIT07= 200
(2) 000100 BIT06= 100
(2) 000040 BIT05= 40
(2) 000020 BIT04= 20
(2) 000010 BIT03= 10
(2) 000004 BIT02= 4
(2) 000002 BIT01= 2
(2) 000001 BIT00= 1
(2) .EQUIV BIT09,BIT9
(2) .EQUIV BIT08,BIT8
(2) .EQUIV BIT07,BIT7
(2) .EQUIV BIT06,BIT6
(2) .EQUIV BIT05,BIT5
(2) .EQUIV BIT04,BIT4
(2) .EQUIV BIT03,BIT3
(2) .EQUIV BIT02,BIT2
(2) .EQUIV BIT01,BIT1
(2) .EQUIV BIT00,BIT0
(2)
(2) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(2) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(2) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(2) 000014 TBITVEC=14 ;: "T" BIT
(2) 000014 TRIVEC= 14 ;:TRACE TRAP
(2) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(2) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(2) 000024 PWRVEC= 24 ;:POWER FAIL
(2) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(2) 000034 TRAPVEC=34 ;:"TRAP" TRAP
(2) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(2) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(2) ;***** THE FOLLOWING BREAK VECTOR AND LINE CLOCK VECTOR ARE INCLUDED
(2) 000100 LKVEC= 100 ;:LINE CLOCK VECTOR
(2) 000140 BRKVEC= 140 ;:BREAK VECTOR
(2) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
(1)
(1) ;INSTRUCTION DEFINITIONS
(1) ;-----
(1) 005746 PUSH1SP=5746 ;:DECREMENT PROCESSOR STACK 1 WORD
(1) 005726 POP1SP=5726 ;:INCREMENT PROCESSOR STACK 1 WORD
(1) 010046 PUSHRO=10046 ;:SAVE R0 ON STACK
(1) 012600 POPRO=12600 ;:RESTORE R0 FROM STACK
```

```

(1)      024646      PUSH2SP=24646      ;DECREMENT STACK TWICE
(1)      022626      POP2SP=22626      ;INCREMENT STACK TWICE
(1)      000200      MASK=BIT7        ;SET INTERRUPT MASK (INHIBIT FURTHER INTERRUPTS)
(1)      000000      CLEAR=0          ;ALLOW INTERRUPTS (CLEAR PROCESSOR STATUS)
(1)
(1)
(1)      ;DZV11 CONTROL AND STATUS REGISTER DEFINITIONS
(1)      ;(DZVCSR)      BIT DEFINITIONS
(1)      ;-----
(1)
(1)      000010      MAINT = BIT3      ;MAINTENANCE MODE ENABLE
(1)      000020      DCLR=BIT4        ;DEVICE CLEAR
(1)      000040      MSENAB=BIT5      ;MASTER SCAN ENABLE
(1)      000100      RIE=BIT6         ;RECEIVER INTERRUPT ENABLE
(1)      000200      RDONE=BIT7       ;RECEIVER DONE
(1)      010000      SILOEN= BIT12    ;SILO ALARM ENABLE
(1)      020000      SILOAL = BIT13   ;SILO ALARM
(1)      040000      TIE=BIT14       ;TRANSMITTER INTERRUPT ENABLE
(1)      100000      TRDY=BIT15      ;TRANSMITTER READY
(1)
(1)      ;DZVCSR WORD DEFINITIONS
(1)      ;-----
(1)      000000      TLO=0           ;TRANSMIT LINE 0
(1)      000400      TL1=BIT8        ;TRANSMIT LINE 1
(1)      001000      TL2=BIT9        ;TRANSMIT LINE 2
(1)      001400      TL3=BIT9!BIT8   ;TRANSMIT LINE 3
(1)
(1)      ;DZVRBUF BIT DEFINITIONS
(1)      ;-----
(1)      010000      PARER=BIT12     ;PARITY ERROR
(1)      020000      FRMERR=BIT13    ;FRAME ERROR
(1)      040000      OVRRUN=BIT14    ;OVERRUN ERROR
(1)      100000      DVALID=BIT15    ;DATA VALID
(1)
(1)      ;DZVRBUF WORD DEFINITIONS
(1)      ;-----
(1)      000000      RLO=0           ;RECEIVER LINE 0
(1)      000400      RL1=BIT8        ;RECEIVER LINE 1
(1)      001000      RL2=BIT9        ;RECEIVER LINE 2
(1)      001400      RL3=BIT9!BIT8   ;RECEIVER LINE 3
(1)
(1)      ;DZVLPR WORD DEFINITIONS
(1)      ;-----
(1)      000000      LP0=0           ;LINE PARAMETER 0
(1)      000001      LP1=BIT0        ;LINE PARAMETER 1
(1)      000002      LP2=BIT1        ;LINE PARAMETER 2
(1)      000003      LP3=BIT1!BIT0   ;LINE PARAMETER 3
(1)
(1)      000000      FIVE=0          ;FIVE BITS/CHAR,1 STOP BIT
(1)      000010      SIX=BIT3        ;SIX BITS/CHAR,1 STOP BIT
(1)      000020      SEVEN=BIT4      ;SEVEN BITS/CHAR,1 STOP BIT
(1)      000030      EIGHT=BIT4!BIT3 ;EIGHT BITS/CHAR,1 STOP BIT
  
```

GENERAL DEFINITIONS AND EQUIVALENCES

(1)	000040	FIVES=BIT5	:FIVE BITS/CHAR, 2 STOP BITS
(1)	000050	SIXS=BIT5!BIT3	:SIX BITS/CHAR, 2 STOP BITS
(1)	000060	SEVENS=BIT5!BIT4	:SEVEN BITS/CHAR, 2 STOP BITS
(1)	000070	EIGHTS=BIT5!BIT4!BIT3	:EIGHT BITS/CHAR, 2 STOP BITS
(1)	000100	PARITY=BIT6	:PARITY ENABLED
(1)	000200	ODDPAR=BIT7	:ODD PARITY ENABLED
(1)	000000	ONESTOP=0	:ONE STOP BIT ENABLED
(1)	000040	TWOSTOP=BIT5	:TWO STOP BITS ENABLED
(1)	000000	EVEPAR=0	:EVEN PARITY ENABLED
(1)	010000	RCVON=BIT12	:ENABLE RECEIVER (RECEIVER ON)
(1)	000000	S50=0	:SPEED 50 BAUD
(1)	000400	S75=BIT8	:SPEED 75 BAUD
(1)	001000	S110=BIT9	:SPEED 110 BAUD
(1)	001400	S134=BIT9!BIT8	:SPEED 134.5 BAUD
(1)	002000	S150=BIT10	:SPEED 150 BAUD
(1)	002400	S300=BIT10!BIT8	:SPEED 300 BAUD
(1)	003000	S600=BIT10!BIT9	:SPEED 600 BAUD
(1)	003400	S1200=BIT10!BIT9!BIT8	:SPEED 1200 BAUD
(1)	004000	S1800=BIT11	:SPEED 1800 BAUD
(1)	004400	S2000=BIT11!BIT8	:SPEED 2000 BAUD
(1)	005000	S2400=BIT11!BIT9	:SPEED 2400 BAUD
(1)	005400	S3600=BIT11!BIT9!BIT8	:SPEED 3600 BAUD
(1)	006000	S4800=BIT11!BIT10	:SPEED 4800 BAUD
(1)	006400	S7200=BIT11!BIT10!BIT8	:SPEED 7200 BAUD
(1)	007000	S9600=BIT11!BIT10!BIT9	:SPEED 9600 BAUD
(1)	007400	S19200=BIT11!BIT10!BIT9!BIT8	:SPEED 19200 BAUD

:DZVTCR BIT DEFINITIONS

(1)	000001	TCR0=BIT0	:ENABLE TRANSMISSION ON LINE 0
(1)	000002	TCR1=BIT1	:ENABLE TRANSMISSION ON LINE 1
(1)	000004	TCR2=BIT2	:ENABLE TRANSMISSION ON LINE 2
(1)	000010	TCR3=BIT3	:ENABLE TRANSMISSION ON LINE 3
(1)	000400	DTR0=BIT8	:DATA TERMINAL READY FOR LINE 0
(1)	001000	DTR1=BIT9	:DATA TERMINAL READY FOR LINE 1
(1)	002000	DTR2=BIT10	:DATA TERMINAL READY FOR LINE 2
(1)	004000	DTR3=BIT11	:DATA TERMINAL READY FOR LINE 3

:DZVMSR BIT DEFINITIONS

(1)	000001	RING0=BIT0	:RING INDICATED ON LINE 0
(1)	000002	RING1=BIT1	:RING INDICATED ON LINE 1
(1)	000004	RING2=BIT2	:RING INDICATED ON LINE 2
(1)	000010	RING3=BIT3	:RING INDICATED ON LINE 3
(1)	000400	CO0=BIT8	:CARRIER PRESENT ON LINE 0
(1)	001000	CO1=BIT9	:CARRIER PRESENT ON LINE 1
(1)	002000	CO2=BIT10	:CARRIER PRESENT ON LINE 2
(1)	004000	CO3=BIT11	:CARRIER PRESENT ON LINE 3

:DZVTDR BIT DEFINITIONS

(1)	000400	BRK0=BIT8	:BREAK FOR LINE 0
(1)	001000	BRK1=BIT9	:BREAK FOR LINE 1

CNDZA-A MACY11 30(1046) 15-DEC-82 14:36
CNDZAA.F11 15-DEC-82 14:31

PAGE 81-5
GENERAL DEFINITIONS AND EQUIVALENCES

SEQ 0024

(1) 002000
(1) 004000

BRK2=BIT10
BRK3=BIT11

:BREAK FOR LINE 2
:BREAK FOR LINE 3

(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)
(1)

TABLE OF LOOP AROUND FUNCTIONS (H325)

I	^
V	^
REC	TRANS
DATA	DATA

I	^
V	^
CO	RTS

I	^
V	^
RING	DTR

```
(1) ;:*****  
(1) ;-----  
(1) ;TRAPCATCHER FOR ILLEGAL INTERRUPTS  
(1) ;THE STANDARD "TRAP CATCHER" IS PLACED  
(1) ;BETWEEN ADDRESS 0 TO ADDRESS 776.  
(1) ;IT LOOKS LIKE "PC+2 HALT".  
(1) ;-----  
(1) ;:*****  
(1) ;  
(1) 000000 . =0  
(1) ;STANDARD INTERRUPT VECTORS  
(1) ;-----  
(1) ;  
(1) . =20  
(1) 000020 000020 .SCOPE ;SCOPE LOOP HANDLER  
(1) 000022 000200 MASK ;HANDLE AT PRIORITY 7  
(1) 000024 007326 $PWRDN ;POWER FAIL HANDLER  
(1) 000026 000300 300 ;SERVICE AT PRIORITY LEVEL 6;VER:0  
(1) 000030 005434 $ERROR ;ERROR HANDLER  
(1) 000032 000300 300 ;SERVICE AT PRIORITY LEVEL 6;VER:0  
(1) 000034 006226 .TRPSRV ;GENERAL HANDLER DISPATCH SERVICE  
(1) 000036 000300 300 ;SERVICE AT PRIORITY LEVEL 6;VER:0  
(2) .SBTTL ACT11 HOOKS  
(2) ;:*****  
(3) ;HOOKS REQUIRED BY ACT11  
(2) 000040 $SVPC= . ;SAVE PC  
(2) 000046 . =46  
(2) 000046 004324 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .STOP  
(2) 000052 . =52  
(2) 000052 000000 .WORD 0 ;:2)SET LOC.52 TO ZERO  
(2) 000040 .=$SVPC ;: RESTORE PC  
(1) ;  
(1) . =174  
(1) 000174 000000 DISPREG:0 ;SOFTWARE DISPLAY REGISTER FOR SWITCHLESS 11S  
(1) 000176 000000 SWREG: 0 ;SOFTARE SWITCH REGISTER FOR SWITCHLESS 11S  
(1) 000200 000200 . =200  
(1) 000200 000137 002116 JMP .START ;GO TO START OF PROGRAM  
(2) ;  
(2) . =1000  
(2) 001000 005200 047103 055104 MTITLE: .ASCIZ <200><12>/CNDZAA/<200>/FOUR LINE ASYNC MUX TESTS, PART 1 OF 2/<200>  
(2)
```



```

(3) .SBTTL COMMON TAGS
(3)
(4) ::*****
(3) ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(3) ::*USED IN THE PROGRAM.
(3)
(3) 001244 $CMTAG: ::START OF COMMON TAGS
(3) 001244 000000 $TSTNM: .WORD 0 ::CONTAINS THE TEST NUMBER
(3) 001246 000 $ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG
(3) 001247 000 $ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT
(3) 001250 000000 $LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS
(3) 001252 000000 $LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS
(3) 001254 000000 $ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED
(3) 001256 000000 $ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE
(3) 001260 000 $ERMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST
(3) 001261 001 $ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION
(3) 001262 000000 $GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA
(3) 001264 000000 $BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA
(3) 001266 000000 $GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA
(3) 001270 000000 $BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA
(3) 001272 000000 .WORD 0 ::RESERVED--NOT TO BE USED
(3) 001274 000000 .WORD 0
(3) 001276 000000 .WORD 0
(3) 001300 000 $AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR
(3) 001301 000 $INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR
(3) 001302 000000 .WORD 0
(3) 001304 177570 SWR: .WORD DSWR ::ADDRESS OF SWITCH REGISTER
(3) 001306 177570 DISPLAY: .WORD DDISP ::ADDRESS OF DISPLAY REGISTER
(3) 001310 177560 $TKS: 177560 ::TTY KBD STATUS
(3) 001312 177562 $TKB: 177562 ::TTY KBD BUFFER
(3) 001314 177564 $TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS
(3) 001316 177566 $TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS
(3) 001320 000 $NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS
(3) 001321 002 $FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED
(3) 001322 012 $FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A 'LINE FEED'
(3) 001323 000 $TPFLG: .BYTE 0 ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(3) 001324 000000 $REGAD: .WORD 0 ::CONTAINS THE ADDRESS FROM
(3) WHICH ($REGO) WAS OBTAINED
(5) 001326 000000 $REG0: .WORD 0 ::CONTAINS (($REGAD)+0)
(5) 001330 000000 $REG1: .WORD 0 ::CONTAINS (($REGAD)+2)
(5) 001332 000000 $REG2: .WORD 0 ::CONTAINS (($REGAD)+4)
(5) 001334 000000 $REG3: .WORD 0 ::CONTAINS (($REGAD)+6)
(5) 001336 000000 $REG4: .WORD 0 ::CONTAINS (($REGAD)+10)
(5) 001340 000000 $REG5: .WORD 0 ::CONTAINS (($REGAD)+12)
(5) 001342 000000 $TMP0: .WORD 0 ::USER DEFINED
(5) 001344 000000 $TMP1: .WORD 0 ::USER DEFINED
(5) 001346 000000 $TMP2: .WORD 0 ::USER DEFINED
(5) 001350 000000 $TMP3: .WORD 0 ::USER DEFINED
(5) 001352 000000 $TMP4: .WORD 0 ::USER DEFINED
(3) 001354 000000 $TIMES: 0 ::MAX. NUMBER OF ITERATIONS
(3) 001356 077 $QUES: .ASCII /?/ ::QUESTION MARK
(3) 001357 015 $CRLF: .ASCII <15> ::CARRIAGE RETURN
(3) 001360 000012 $LF: .ASCIIZ <12> ::LINE FEED

```

```

(3) .SBTTL ERROR POINTER TABLE
(3)
(3) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(3) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(3) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(3) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(3) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(3)
(3) ;* EM ::POINTS TO THE ERROR MESSAGE
(3) ;* DH ::POINTS TO THE DATA HEADER
(3) ;* DT ::POINTS TO THE DATA
(3) ;* DF ::POINTS TO THE DATA FORMAT
(3)
(3) 001362 $ERRTB:
(2) ;PROGRAM CONTROL PARAMETERS
(2) :-----
(2) NEXT: 0 ;ADDRESS OF NEXT TEST TO BE EXECUTED
(2) 001364 000000 LOCK: 0 ;ADDRESS FOR LOCK ON CURRENT TEST,TIGHT LOOP
(2)
(2) ;PROGRAM VARIABLES
(2) :-----
(2) 001366 000017 LINE: 17 ;DEFAULT ALL FOUR LINES RUNNING
(2) 001370 017470 PAR: 17470 ;PARAMETERS: 8 BITS/CHAR,2 STOP BITS,19200 BAUD,NO PARIT
(2) 001372 000000 MODE: 0 ;DEFAULT MAINTENANCE MODE
(2) 001374 000000 SAVLIN: 0 ;LINE NUMBER
(2) 001376 000000 XMTLIN: 0 ;TRANSMISSION LINE NUMBER
(2) 001400 000000 XMTCNT: 0 ;COUNT OF WORDS IN A TRANSMISSION PATTERN
(2) 001402 000000 REGIST: 0 ;DEVICE ADDRESS STORAGE LOCATION
(2) 001404 000000 SAVPC: 0 ;PROGRAM COUNTER STORAGE
(2) 001406 000001 DZVACTV: .BLKW 1 ;*DZV11'S SELECTED ACTIVE.
(2) 001410 000001 SAVACTV: .BLKW 1 ;*A BIT MAP OF DZV11'S IN THE SYSTEM
(2) 001412 000001 RUN: 1 ;*POINTER ONE PAST RUNNING DEVICE.
(2) 001414 000001 DZVNUM: .BLKB 1 ;*OCTAL NUMBER OF DZV11'S IN THE SYSTEM
(2) 001415 001 SAVNUM: .BYTE 1 ;*WORKABLE NUMBER.
(2) 001416 000001 SAVNO: .BLKB 1 ;*OCTAL NO. OF DZV11'S BEING TESTED
(2) 001420 001420 .EVEN
(2) 001500 ACTIVE: DZV.MAP ;TABLE POINTER.

```



```

(2)
(2)
(2)
(2)
(2) 001422 000 INIFLG: .BYTE 0 ;PROGRAM INITIALIZATION FLAG
(2) 001423 000 HDRFLG: .BYTE 0 ;PROGRAM INITIALIZATION FLAG FOR HEADER MAP
(2) 001424 000 MNTFLG: .BYTE 0 ;MAINTENANCE BIT SET FLAG
(2) 001425 000 DONFLG: .BYTE 0 ;TRANSMISSION COMPLETION FLAG
(2) .EVEN
(2) ;DATA VARIABLES
(2) 001426 000000 TD0: .WORD 0
(2) 001430 000000 TD1: .WORD 0
(2) 001432 000000 TD2: .WORD 0
(2) 001434 000000 TD3: .WORD 0
(2) 001436 000000 TR0: .WORD 0
(2) 001440 000000 TR1: .WORD 0
(2) 001442 000000 TR2: .WORD 0
(2) 001444 000000 TR3: .WORD 0
(2) 001446 STOP:
(2) .SBTTL APT PARAMETER BLOCK
(2)
(3)
(2) ;*****
(2) ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(3) ;*****
(2) ;$X= ;:SAVE CURRENT LOCATION
(2) ;=24 ;:SET POWER FAIL TO POINT TO START OF PROGRAM
(2) 000024 000200 200 ;:FOR APT START UP
(2) ;=44 ;:POINT TO APT INDIRECT ADDRESS PNTR.
(2) 000044 001446 $APTHDR ;:POINT TO APT HEADER BLOCK
(2) 001446 ;=. $X ;:RESET LOCATION COUNTER
(3) ;*****
(2) ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(2) ;INTERFACE SPEC.
(2)
(2) 001446 $APTHD:
(2) 001446 000000 $SHIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(2) 001450 001120 $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
(2) 001452 000120 $STMT: .WORD 80. ;:RUN TIM OF LONGEST TEST
(2) 001454 000024 $PASTM: .WORD 20. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(2) 001456 000000 $UNITM: .WORD 0. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(2) 001460 000052 .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
(1) ;DZV11 STATUS TABLE AND ADDRESS ASSIGNMENTS
(1) ;-----
(1)
(1) ;=1500
(1) 001500 DZV.MAP:
(3)
(3) 001500 000001 DZCR0: .BLKW 1 ;:CONTROL STATUS REGISTER FOR DZV11 NUMBER 0
(3) 001502 000001 DZVC0: .BLKW 1 ;:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 0
(3) 001504 000001 LINE0: .BLKW 1 ;:ALL LINES SELECTED
(3) 001506 000001 PAR0: .BLKW 1 ;:PARAMETERS
(3) 001510 000001 MANT0: .BLKW 1 ;:MAINTENANCE MODE FOR THIS DEVICE
(3)
(3) 001512 000001 DZCR1: .BLKW 1 ;:CONTROL STATUS REGISTER FOR DZV11 NUMBER 1
(3) 001514 000001 DZVC1: .BLKW 1 ;:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 1
(3) 001516 000001 LINE1: .BLKW 1 ;:ALL LINES SELECTED

```

(3)	001520	000001	PAR1:	.BLKW	1	:PARAMETERS
(3)	001522	000001	MANT1:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001524	000001	DZCR2:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 2
(3)	001526	000001	DZVC2:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 2
(3)	001530	000001	LINE2:	.BLKW	1	:ALL LINES SELECTED
(3)	001532	000001	PAR2:	.BLKW	1	:PARAMETERS
(3)	001534	000001	MANT2:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001536	000001	DZCR3:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 3
(3)	001540	000001	DZVC3:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 3
(3)	001542	000001	LINE3:	.BLKW	1	:ALL LINES SELECTED
(3)	001544	000001	PAR3:	.BLKW	1	:PARAMETERS
(3)	001546	000001	MANT3:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001550	000001	DZCR4:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 4
(3)	001552	000001	DZVC4:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 4
(3)	001554	000001	LINE4:	.BLKW	1	:ALL LINES SELECTED
(3)	001556	000001	PAR4:	.BLKW	1	:PARAMETERS
(3)	001560	000001	MANT4:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001562	000001	DZCR5:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 5
(3)	001564	000001	DZVC5:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 5
(3)	001566	000001	LINE5:	.BLKW	1	:ALL LINES SELECTED
(3)	001570	000001	PAR5:	.BLKW	1	:PARAMETERS
(3)	001572	000001	MANT5:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001574	000001	DZCR6:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 6
(3)	001576	000001	DZVC6:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 6
(3)	001600	000001	LINE6:	.BLKW	1	:ALL LINES SELECTED
(3)	001602	000001	PAR6:	.BLKW	1	:PARAMETERS
(3)	001604	000001	MANT6:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001606	000001	DZCR7:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 7
(3)	001610	000001	DZVC7:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 7
(3)	001612	000001	LINE7:	.BLKW	1	:ALL LINES SELECTED
(3)	001614	000001	PAR7:	.BLKW	1	:PARAMETERS
(3)	001616	000001	MANT7:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001620	000001	DZCR10:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 10
(3)	001622	000001	DZVC10:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 10
(3)	001624	000001	LINE10:	.BLKW	1	:ALL LINES SELECTED
(3)	001626	000001	PAR10:	.BLKW	1	:PARAMETERS
(3)	001630	000001	MANT10:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001632	000001	DZCR11:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 11
(3)	001634	000001	DZVC11:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 11
(3)	001636	000001	LINE11:	.BLKW	1	:ALL LINES SELECTED
(3)	001640	000001	PAR11:	.BLKW	1	:PARAMETERS
(3)	001642	000001	MANT11:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)	001644	000001	DZCR12:	.BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 12
(3)	001646	000001	DZVC12:	.BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 12
(3)	001650	000001	LINE12:	.BLKW	1	:ALL LINES SELECTED
(3)	001652	000001	PAR12:	.BLKW	1	:PARAMETERS
(3)	001654	000001	MANT12:	.BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE

(3)					
(3)	001656	000001	DZCR13: .BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 13
(3)	001660	000001	DZVC13: .BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 13
(3)	001662	000001	LINE13: .BLKW	1	:ALL LINES SELECTED
(3)	001664	000001	PAR13: .BLKW	1	:PARAMETERS
(3)	001666	000001	MANT13: .BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)					
(3)	001670	000001	DZCR14: .BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 14
(3)	001672	000001	DZVC14: .BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 14
(3)	001674	000001	LINE14: .BLKW	1	:ALL LINES SELECTED
(3)	001676	000001	PAR14: .BLKW	1	:PARAMETERS
(3)	001700	000001	MANT14: .BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)					
(3)	001702	000001	DZCR15: .BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 15
(3)	001704	000001	DZVC15: .BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 15
(3)	001706	000001	LINE15: .BLKW	1	:ALL LINES SELECTED
(3)	001710	000001	PAR15: .BLKW	1	:PARAMETERS
(3)	001712	000001	MANT15: .BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)					
(3)	001714	000001	DZCR16: .BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 16
(3)	001716	000001	DZVC16: .BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 16
(3)	001720	000001	LINE16: .BLKW	1	:ALL LINES SELECTED
(3)	001722	000001	PAR16: .BLKW	1	:PARAMETERS
(3)	001724	000001	MANT16: .BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(3)					
(3)	001726	000001	DZCR17: .BLKW	1	:CONTROL STATUS REGISTER FOR DZV11 NUMBER 17
(3)	001730	000001	DZVC17: .BLKW	1	:RECEIVER AND BASE VECTOR FOR DZV11 NUMBER 17
(3)	001732	000001	LINE17: .BLKW	1	:ALL LINES SELECTED
(3)	001734	000001	PAR17: .BLKW	1	:PARAMETERS
(3)	001736	000001	MANT17: .BLKW	1	:MAINTENANCE MODE FOR THIS DEVICE
(1)					
(1)	001740	177777	DZV.END:	177777	

```
(1)                                     ;DEFINITIONS FOR TRAP SUBROUTINE CALLS
(1)                                     ;POINTERS TO SUBROUTINES CAN BE FOUND
(1)                                     ;IN THE TABLE IMMEDIATELY FOLLOWING THE DEFINITIONS
(1)                                     ;:*****
(1)                                     ;-----
(1) 001742                             .TRPTAB:
(3) 001742 104400                       ADVANCE=TRAP+0           ;CALL TO ADVANCE TO NEXT TEST( OR SCOPE THIS ONE)
(2) 001742 006322                       .ADVANCE
(3) 001742 104401                       SCOP1=TRAP+1          ;CALL TO LOOP ON CURRENT DATA HANDLER
(2) 001744 004634                       .SCOP1
(3) 001744 104402                       TYPE=TRAP+2         ;CALL TO TELETYPE OUTPUT ROUTINE
(2) 001746 004660                       .TYPE
(3) 001746 104403                       INSTR=TRAP+3        ;CALL TO ASCII STRING INPUT ROUTINE
(2) 001750 005426                       .INSTR
(3) 001750 104404                       INSTER=TRAP+4       ;CALL TO INPUT ERROR HANDLER
(2) 001752 005532                       .INSTER
(3) 001752 104405                       PARAM=TRAP+5        ;CALL TO NUMERICAL DATA INPUT ROUTINE
(2) 001754 005552                       .PARAM
(3) 001754 104406                       SETFLG=TRAP+6       ;CALL TO SET FLAG ROUTINE
(2) 001756 010164                       .SETFLG
(3) 001756 104407                       SAV05=TRAP+7        ;CALL TO REGISTER SAVE ROUTINE
(2) 001760 005752                       .SAV05
(3) 001760 104410                       RES05=TRAP+10       ;CALL TO REGISTER RESTORE ROUTINE
(2) 001762 006012                       .RES05
(3) 001762 104411                       CONVRT=TRAP+11      ;CALL TO DATA OUTPUT ROUTINE
(2) 001764 006044                       .CONVRT
(3) 001764 104412                       CNVRT=TRAP+12       ;CALL TO DATA OUTPUT ROUTINE WITHOUT CR/LF.
(2) 001766 006050                       .CNVRT
(3) 001766 104413                       DEVICE.CLR=TRAP+13  ;CALL TO ISSUE A DEVICE CLEAR
(2) 001770 006250                       .DEVICE.CLR
(3) 001770 104414                       DELAY=TRAP+14       ;CALL TO DELAY FOR FAST CPU'S
(2) 001772 006302                       .DELAY
(3) 001772 104415                       PARMD=TRAP+15       ;CONVERT DECIMAL STRING TO OCTAL
(2) 001774 011170                       .PARMD
(3) 001774 104416                       PAWCH=TRAP+16       ;SET FLAG ECHO OR CABLE
(2) 001776 010304                       .PAWCH
(3) 001776 104417                       DCLASM=TRAP+17      ;CLEAR DEVICE, SET MAINT. BIT IF I MODE
(2) 002000 006270                       .DCLASM
(3) 002000 104420                       SHIFT=TRAP+20       ;CALL TO ROTATE LINE POINTER
(2) 002002 006334                       .SHIFT
(3) 002002 104421                       LPRSET=TRAP+21      ;CALL TO SET UP LPR DEVICE REGISTER
(2) 002004 006352                       .LPRSET
(3) 002004 104422                       BUFSET=TRAP+22      ;CALL TO ZERO BUFFER AREA
(2) 002006 006412                       .BUFSET
(1)                                     ;-----
(1)                                     ;:*****
```

```

(1)                                     :DZV11 VECTOR AND REGISTER INDIRECT POINTERS
(1)                                     :WORKING AREA
(1)
(1) 002010 160040 DZVCSR: 160040 :R/W
(1) 002012 160041 HDZVCSR:160041 :R/W
(1) 002014 160042 DZVRBUF:160042 :READ ONLY
(1) 002016 160043 HDZVRBUF:160043 :READ ONLY
(1) 002020 160042 DZVLPR: 160042 :WRITE ONLY
(1) 002022 160043 HDZVLPR:160043 :WRITE ONLY
(1) 002024 160044 DZVTCR: 160044 :R/W
(1) 002026 160045 HDZVTCR:160045 :R/W
(1) 002030 160046 DZVMSR: 160046 :READ ONLY
(1) 002032 160047 HDZVMSR:160047 :READ ONLY
(1) 002034 160046 DZVTDR: 160046 :WRITE ONLY
(1) 002036 160047 HDZVTDR:160047 :WRITE ONLY
(1)
(1)                                     :DEFAULT DZV VECTORS
(1)
(1) 002040 000300 DZVRIV: 300 :REC INTR VECTOR
(1) 002042 000302 DZVRIS: 302 :REC INTR STATUS
(1) 002044 000304 DZVTIV: 304 :XMIT INTR VECTOR
(1) 002046 000306 DZVTIS: 306 :XMIT INTR STATUS
(1)
(1)

```



```

(1) 002752 006203 ASR R3 ;GET THE NEXT FLAG
(1) 002754 103370 BCC 75$ ;IF IT ISN'T SET, THERE'S AN ERROR
(1) 002756 000241 CLC ;INITIALIZE THE 'C' BIT FOR TESTING OF THE NEXT PAIR
(1) 002760 000761 BR 70$ ;GO TEST THE NEXT PAIR OF FLAGS
(1) ;GET THE LINE PARAMETER REGISTER ARGUMENT
(1)
(1) 002762 85$: INSTR ;CALL THE STRING INPUT ROUTINE
(2) 002762 104403 94$ ;POINTER TO MESSAGE TO BE PRINTED
(2) 002764 003240 PARAM ;CALL THE OCTAL TO ASCII CONVERT ROUTINE
(2) 002766 104405 0 ;LOWEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 002770 000000 17 ;HIGHEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 002772 000017 PARO ;POINTER TO MAP LOCATION TO BE FILLED
(2) 002774 001506 .BYTE 0 ;MASK OF INVALID BITS FOR THIS PARAMETER
(2) 002776 000 .BYTE 1 ;NUMBER OF PARAMETERS TO STORE
(2) 002777 001 MOV #LINE0,R2 ;POINT TO THE LINE SELECTION PARAMETER
(1) 003000 012702 001504 MOV #PARO,R3 ;POINT TO THE CHOSEN PARAMETERS
(1) 003004 012703 001506 MOV (R3),R4 ;USE BAUD RATE AS AN INDEX IN DELAY TABLE
(1) 003010 011304 ASL R4 ;ALIGN INDEX ON WORD BOUNDARY
(1) 003012 006304 MOV DLYTBL(R4),DLYCNT ;SET THE DELAY COUNT FOR THIS BAUD RATE
(1) 003014 016437 017214 006320 SWAB (R3) ;PLACE IN HIGH BYTE
(1) 0C3022 000313 BIS #10070,(R3) ;PLACE EXTRA PARAMETERS INTO LOC
(1) 003024 052713 010070 90$: MOV (R2),12(R2) ;LOAD THE LINES
(1) 003030 011262 000012 MOV (R3),12(R3) ;LOAD THE PARAMETERS
(1) 003034 011363 000012 ADD #12,R2 ;POINT TO THE NEXT SET
(1) 003040 062702 000012 ADD #12,R3 ;... OF BOTH PARAMETERS
(1) 003044 062703 000012 CMP R3,#PAR17 ;HAVE THE TABLE BOUNDARIES BEEN EXCEEDED?
(1) 003050 020327 001734 BNE 90$ ;IF NOT, GO LOAD SOME MORE PARAMETERS
(1) 003054 001365 RTS ;RETURN TO CALLING BLOCK
(1) 003056 000207 PC
(1) 003060 030600 052123 041440 91$: .ASCIZ <200>/1ST CSR ADDRESS (160000:163770): /
(1) 003124 030600 052123 053040 92$: .ASCIZ <200>/1ST VECTOR ADDRESS (300:770): /
(1) 003165 200 044514 042516 93$: .ASCIZ <200>/LINES ACTIVE BY BIT <IN OCTAL>(001:17): /
(1) 003240 042200 043105 052501 94$: .ASCIZ <200>/DEFAULT BAUD RATE <IN OCTAL>(00:17): /
(1) 003310 021600 047440 020106 95$: .ASCIZ <200>/# OF DZV11'S <IN OCTAL> (1:20): /
(1) 003353 200 040515 047111 96$: .ASCII <200>/MAINTENANCE MODE/
(1) 003374 020200 042533 052130 .ASCII <200>/ [EXTERNAL <H325> (E)]/
(1) 003430 020200 044533 052116 .ASCII <200>/ [INTERNAL <DZVCSR03=1>(I)]/
(1) 003465 200 055440 052123 .ASCIZ <200>/ [STAGGERED <H329> (S)]: /
(1) 003524 042600 052116 051105 97$: .ASCIZ <200>/ENTER DELAY PARAMETER: /
(1) 003556 003556 .EVEN
(1) 003556 122737 000377 001422 100$: CMPB #377,INIFLG ;ONLY DO AUTO SIZE ON 1ST START
(1) 003564 001013 BNE 105$ ;
(1) 003566 032777 000200 175510 BIT #BIT7,@SWR ;BIT7=1??
(1) 003574 001007 BNE 105$ ;BR IF NO AUTO SIZE
(1) 003576 005737 017532 TST KXTFLAG ; FALCON ?? ;:GPA
(1) 003602 001402 BEQ 1002$ ; SKIP NEXT IF NOT. ;:GPA
(1) 003604 000137 002342 JMP 20$ ; YES, DON'T AUTO-SIZE. ;:GPA
(1) 003610 004737 011320 1002$: JSR PC,AUTO.SIZE ;:GPA
(1) 003614 105737 001423 105$: TSTB HDRFLG ;GO DO THE AUTO SIZE
(1) 003620 001021 BNE 120$ ;HAS THE TABLE BEEN TYPED YET?
(1) 003622 105337 001423 DECB HDRFLG ;IF SO, DON'T TYPE IT AGAIN
(1) 003626 104402 010062 TYPE ,XHEAD ;INDICATE THAT THE TABLE WILL BE TYPED
(1) 003632 012700 001500 MOV #DZV.MAP,R0 ;TYPE MAP HEADER
;SET POINTER
  
```

```

(1) 003636 010037 001344      110$:  MOV      R0,$TMP1      ;POINT TO THE MAP LOCATION
(1) 003642 012037 001346      MOV      (R0)+,$TMP2    ;SET DATA
(1) 003646 022737 177777 001346  CMP      #-1,$TMP2     ;END OF LIST?
(1) 003654 001403      BEQ      120$          ;BR IF YES
(1) 003656 104411      115$:  CONVRT      ;CALL THE OCTAL TO ASCII CONVERSION ROUTINE
(1) 003660 010152      XSTATQ    ;CONVERT THE DATA AT THIS ADDRESS
(1) 003662 000765      BR        110$        ;GO PRINT THE NEXT PARAMETER
(1) 003664 013737 001410 001406 120$:  MOV      SAVACTV,DZVACTV ;COPY BIT MAP OF SYSTEM DEVICES ACTIVE
(1) 003672 113737 001414 001416  MOVB     DZVNUM,SAVNO   ;COPY NO. OF SYSTEM DEVICES ACTIVE
(1) 003700 032777 000100 175376  BIT      #SW06,@SWR    ;DESELECT SPECIFIC DEVICES??
(1) 003706 001431      BEQ      135$        ;BR IF NO.
(1) 003710      121$:  INSTR      ;CALL THE STRING INPUT ROUTINE
(2) 003710 104403      MNEW      ;POINTER TO MESSAGE TO BE PRINTED
(2) 003712 010000      PARAM     ;CALL THE OCTAL TO ASCII CONVERT ROUTINE
(2) 003714 104405      1        ;LOWEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 003716 000001      177777   ;HIGHEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 003720 177777      DZVACTV   ;POINTER TO MAP LOCATION TO BE FILLED
(2) 003722 001406      .BYTE    0          ;MASK OF INVALID BITS FOR THIS PARAMETER
(2) 003724      000      .BYTE    1          ;NUMBER OF PARAMETERS TO STORE
(2) 003725      001      CMP      DZVACTV,SAVACTV ;IS THE VALUE VALID?
(1) 003726 023737 001406 001410  BLOS     122$        ;BRANCH IF YES
(1) 003734 101403      TYPE     ,MERR3     ;IF NOT THEN TYPE ERROR
(1) 003736 104402 007652      BR        121$      ;GO REASK QUESTION
(1) 003742 000762      CLR      SAVNO      ;CLEAR NO. OF DEVICES BEING TESTED
(1) 003744 105037 001416 001344 122$:  MOV      DZVACTV,$TMP1 ;COPY BIT MAP OF ACTIVE DEVICES BEING TESTED
(1) 003750 013737 001406 001344 126$:  ASR      $TMP1      ;SHIFT OUT AN ACTIVE BIT
(1) 003756 006237 001344      BCC      127$        ;IF NOT ACTIVE SKIP INCREMENT
(1) 003762 103002      INCB     SAVNO      ;IF ACTIVE RECORD IT
(1) 003764 105237 001416 127$:  BNE      126$        ;IF ALL ACTIVE BITS RECORDED DON'T BRANCH
(1) 003770 001372      BIT      #SW04,@SWR  ;CHECK TO SEE IF DELAY COUNT CHANGES
(1) 003772 032777 000020 175304 135$:  BEQ      140$        ;IF NOT, GO CLEAR VECTOR AREA
(1) 004000 001407      INSTR    ;CALL THE STRING INPUT ROUTINE
(2) 004002 104403      97$     ;POINTER TO MESSAGE TO BE PRINTED
(2) 004004 003524      PARAM     ;CALL THE OCTAL TO ASCII CONVERT ROUTINE
(2) 004006 104405      1        ;LOWEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 004010 000001      177777   ;HIGHEST LEGITIMATE VALUE OF EXPECTED RESPONSE
(2) 004012 177777      DLYCNT   ;POINTER TO MAP LOCATION TO BE FILLED
(2) 004014 006320      .BYTE    0          ;MASK OF INVALID BITS FOR THIS PARAMETER
(2) 004016      000      .BYTE    1          ;NUMBER OF PARAMETERS TO STORE
(2) 004017      001      140$:  MOV      #300,R0      ;PREPARE TO CLEAR THE FLOATING
(1) 004020 012700 000300      MOV      #302,R1      ;VECTOR AREA. 300-776
(1) 004024 012701 000302      MOV      R1,(R0)+    ;START PUTTING 'PC+2 - HALT'
(1) 004030 010120 145$:  CLR      (R1)+      ;IN VECTOR AREA.
(1) 004032 005021      CMP      (R0)+,(R1)+ ;POP POINTERS
(1) 004034 022021      TST     KXTFLAG     ; IF FALCON...
(1) 004036 005737 017532      BEQ     1001$       ;: :GPA
(1) 004042 001403      CMP     R0,#400     ;: :GPA
(1) 004044 020027 000400      402     ;...STOP AT 400.
(1) 004050 000402      1001$:  ; SKIP NEXT ;: :GPA
(1) 004052      ;: :GPA
(1) 004052 022700 001000      CMP     #1000,R0    ;ALL DONE??
(1) 004056 001364      BNE     145$        ;BR IF NO.
(1) ;TEST START AND RESTART
(1) ;-----
(1)

```

```

(1) 004060 012706 001120      .BEGIN: MOV      #STACK,SP      ;SET UP STACK
(1) 004064 106427 000200      MTPS     #MASK      ;LOCK OUT INTERRUPTS
(1) 004070 005737 000042      TST      @#42      ;IS PROGRAM UNDER MONITOR CONTROL
(1) 004074 001015                BNE      2$        ;BR IF YES
(1) 004076 032777 000004 175200 BIT      #BIT2,@SWR ;CHECK FOR LOCK ON TEST
(1) 004104 001406                BEQ      1$        ;BR IF NO LOCK DESIRED.
(1) 004106 104402 007676      TYPE     ,MLOCK    ;TYPE LOCK SELECTED.
(1) 004112 012737 000240 004402 MOV      #NOP,TTST ;ADJUST SCOPE ROUTINE.
(1) 004120 000403                BR       2$        ;CONTINUE ALONG.
(1) 004122 013737 004630 004402 1$: MOV     BRW,TTST  ;PREPARE NORMAL SCOPE ROUTINE
(1) 004130 012737 010464 001252 2$: MOV     #CYCLE,$LPADR ;START AT "CYCLE" FIND WHICH DEVICE TO TEST
(1) 004136 113737 001416 001415 MOVB    SAVNO,SAVNUM ;COPY ACTIVE DEVICES BEING TESTED
(1) 004144 104402 007567      TYPE     ,MR       ;TYPE 'RUNNING'
(1) 004150 000177 175076      JMP      @$LPADR   ;START TESTING
8704      ;;GPA  PRGEND  DZV11,<END PASS CNDZA-A >,10.

```



```

(2) 004346 000001      XVEC: 1
(2) 004350 003        .BYTE 3.2
(2) 004352 002040     DZVRIV
(2) 004354 000001     XPASS: 1
(2) 004356 006        .BYTE 6.2
(2) 004360 001126     $PASS
(2) 004362 000001     XERR: 1
(2) 004364 006        .BYTE 6.2
(2) 004366 001256     $ERTTL

(2)                                     ;SCOPE LOOP AND ITERATION HANDLER
(2) -----
(2)
(3)
(3) .SBTTL SCOPE HANDLER ROUTINE STARTS
(3) ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
(3) ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
(3) ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
(3) ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
(3) ;*SW14=1      LOOP ON TEST
(3) ;*SW11=1      INHIBIT ITERATIONS
(3) ;*CALL
(3) ;*          SCOPE          ;;SCOPE=IOT

(3) 004370          $SCOPE:
(5) 004370 005037 001262 .SCOPE: CLR      $ERRPC      ;CLEAR LAST ERROR PC.
(5) 004374 022716 012026   CMP      #TST1+2,(SP) ;IS THIS THE SCOPE AT THE BEGINNING OF TST1?
(5) 004400 001413         BEQ      $XTSTR    ;IF SO, DON'T LOOP ON IT
(5) 004402 000406         BR       1$      ;GOTO 1$ (IF LOCK SW02=1; THIS LOC =240)
(5) 004404 105777 174700   TSTB   @STKS     ;KEYBOARD DONE?
(5) 004410 100067         BPL     $OVER    ;BR IF NO. (LOCK: HIT KEY TO GOTO NEXT TEST)
(5) 004412 017766 174674 177776   MOV    @STKB,-2(SP) ;CLEAR DONE BIT
(3) 004420 032777 040000 174656 1$: BIT   #BIT14,@SWR ;LOOP ON PRESENT TEST?
(3) 004426 001060         BNE     $OVER    ;YES IF SW14=1
(3) ;#####START OF CODE FOR THE XOR TESTER#####
(3) 004430 000416 $XTSTR: BR   6$      ;IF RUNNING ON THE 'XOR' TESTER CHANGE
(3) ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
(3) 004432 013746 000004         MOV   @#ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
(3) 004436 012737 004456 000004   MOV   #5$,@#ERRVEC ;SET FOR TIMEOUT
(3) 004444 005737 177060         TST   @#177060    ;TIME OUT ON XOR?
(3) 004450 012637 000004         MOV   (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 004454 000436         BR    $SVLAD    ;GO TO THE NEXT TEST
(3) 004456 022626 5$: CMP   (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
(3) 004460 012637 000004         MOV   (SP)+,@#ERRVEC ;RESTORE THE ERROR VECTOR
(3) 004464 000441         BR    $OVER    ;LOOP ON THE PRESENT TEST
(3) 004466 6$: ;#####END OF CODE FOR THE XOR TESTER#####
(3) 004466 105737 001247 2$: TSTB  $ERFLG    ;HAS AN ERROR OCCURRED?
(3) 004472 001404         BEQ   3$      ;BR IF NO
(3) 004474 105037 001247 4$: CLRB  $ERFLG    ;ZERO THE ERROR FLAG
(3) 004500 005037 001354         CLR  $TIMES   ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(3) 004504 032777 004000 174572 3$: BIT  #BIT11,@SWR ;INHIBIT ITERATIONS?
(3) 004512 001011         BNE  1$      ;BR IF YES
(3) 004514 005737 001126         TST  $PASS    ;IF FIRST PASS OF PROGRAM
(3) 004520 001406         BEQ  1$      ;INHIBIT ITERATIONS
(3) 004522 005237 001250         INC  $ICNT    ;INCREMENT ITERATION COUNT
(3) 004526 023737 001354 001250   CMP  $TIMES,$ICNT ;CHECK THE NUMBER OF ITERATIONS MADE
(3) 004534 002015         BGE  $OVER    ;BR IF MORE ITERATION REQUIRED

```



```

(3) 004536 012737 000001 001250 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
(3) 004544 013737 004632 001354 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
(3) 004552 105237 001246 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
(3) 004556 113737 001246 001124 MOV $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
(3) 004564 011637 001252 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
(3) 004570 013777 001246 174510 $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(3) 004576 013716 001252 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
(5) 004602 004737 007062 JSR PC,SERV.G ;;FIND OUT IF ^G WAS TYPED
(5) 004606 105037 001424 CLR MNTFLG ;;CLEAR THE MAINTENANCE BIT SETTER AFTER EACH TEST
(5) 004612 005737 001372 TST MODE ;;HAS THE MODE BEEN CHANGED?
(5) 004616 001003 BNE 4$ ;;IF NOT INTERNAL , GO DO A TEST
(5) 004620 112737 000010 001424 MOV #MAINT,MNTFLG ;;IF INTERNAL MODE NOW, SET THE MAINTENANCE BIT
(5) 004626 000002 4$: RTI ;;GO DO THE TEST
(5) 004630 000406 BRW: 406
(3) 004632 000005 $MXCNT: 5 ;;MAX. NUMBER OF ITERATIONS
(1)
(1) ;CHECK FOR FREEZE ON CURRENT DATA
(1) ;-----
(1)
(1) 004634 032777 001000 174442 .SCOPI: BIT #SW09,@SWR ;;IS SW09=1(SET)?
(1) 004642 001405 BEQ 1$ ;;BR IF NOT SET.
(1) 004644 005737 001364 TST LOCK ;;IS THERE A TIGHT LOOP SPECIFIED?
(1) 004650 001402 BEQ 1$ ;;IF NO, RETURN
(1) 004652 013716 001364 MOV LOCK,(SP) ;;IF YES, GOTO THE ADDRESS IN LOCK.
(1) 004656 000002 1$: RTI ;;GO BACK.
(1)
(1) 004660 032777 010000 174416 .TYPE: BIT #SW12,@SWR ;;INHIBIT ALL PRINTOUT??
(1) 004666 001403 BEQ $TYPE ;;IF NOT, GO TYPE
(1) 004670 062716 000002 ADD #2,(SP) ;;SKIP OVER MESSAGE POINTER
(1) 004674 000002 RTI ;;RETURN TO WHERE PROCEDURE WAS INVOKED
(2)
(2) .SBTTL TYPE ROUTINE
(3)
(2) ;*****
(2) ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(2) ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(2) ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(2) ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(2) ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(2) ;*
(2) ;*CALL:
(2) ;*1) USING A TRAP INSTRUCTION
(2) ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(2) ;*OR
(2) ;* TYPE
(2) ;* MESADR
(2) ;*
(2)
(2) 004676 105737 001323 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(2) 004702 100002 BPL 1$ ;;BR IF YES
(2) 004704 000000 HALT ;;HALT HERE IF NO TERMINAL
(2) 004706 000430 BR 3$ ;;LEAVE
(2) 004710 010046 1$: MOV R0,-(SP) ;;SAVE R0
(2) 004712 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
(2) 004716 122737 000001 001140 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(2) 004724 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
(2) 004726 132737 000100 001141 BITB #APTPOOL,$ENVM ;;SPOOL MESSAGE TO APT

```



```

(2) .SBTTL APT COMMUNICATIONS ROUTINE
(2)
(3) *****
(2) 005160 112737 000001 005424 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROR
(2) 005166 112737 000001 005422 $ATY3:  MOV  #1,$MFLG      ;;TO TYPE A MESSAGE
(2) 005174 000403                BR      $ATYC
(2) 005176 112737 000001 005424 $ATY4:  MOV  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
(2) 005204 $ATYC:
(4) 005204 010046                MOV  R0,-(SP)      ;;PUSH R0 ON STACK
(4) 005206 010146                MOV  R1,-(SP)      ;;PUSH R1 ON STACK
(2) 005210 105737 005422                TST  $MFLG        ;;SHOULD TYPE A MESSAGE?
(2) 005214 001450                BEQ  5$           ;;IF NOT: BR
(2) 005216 122737 000001 001140        CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
(2) 005224 001031                BNE  3$           ;;IF NOT: BR
(2) 005226 132737 000100 001141        BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(2) 005234 001425                BEQ  3$           ;;IF NOT: BR
(2) 005236 017600 000004                MOV  @4(SP),R0     ;;GET MESSAGE ADDR.
(2) 005242 062766 000002 000004        ADD  #2,4(SP)      ;;BUMP RETURN ADDR.
(2) 005250 005737 001120 1$:          TST  $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
(2) 005254 001375                BNE  1$           ;;IF NOT: WAIT
(2) 005256 010037 001134                MOV  R0,$MSGAD
(2)                ;;PUT ADDR IN MAILBOX
(2) 005262 105720 2$:          TSTB (R0)+        ;;FIND END OF MESSAGE
(2) 005264 001376                BNE  2$
(2) 005266 163700 001134                SUB  $MSGAD,R0     ;;SUB START OF MESSAGE
(2) 005272 006200                ASR  R0            ;;GET MESSAGE LNGTH IN WORDS
(2) 005274 010037 001136                MOV  R0,$MSGGLT   ;;PUT LENGTH IN MAILBOX
(2) 005300 012737 000004 001120        MOV  #4,$MSGTYPE  ;;TELL APT TO TAKE MSG.
(2) 005306 000413                BR   5$
(2) 005310 017637 000004 005334 3$:          MOV  @4(SP),4$    ;;PUT MSG ADDR IN JSR LINKAGE
(2) 005316 062766 000002 000004        ADD  #2,4(SP)     ;;BUMP RETURN ADDRESS
(4) 005324 013746 177776                MOV  177776,-(SP) ;;PUSH 177776 ON STACK
(2) 005330 004737 004676                JSR  PC,$TYPE     ;;CALL TYPE MACRO
(2) 005334 000000 4$:          .WORD 0
(2) 005336 5$:
(2) 005336 105737 005424 10$:          TSTB $FFLG        ;;SHOULD REPORT FATAL ERROR?
(2) 005342 001416                BEQ  12$          ;;IF NOT: BR
(2) 005344 005737 001140                TST  $ENV         ;;RUNNING UNDER APT?
(2) 005350 001413                BEQ  12$          ;;IF NOT: BR
(2) 005352 005737 001120 11$:          TST  $MSGTYPE     ;;FINISHED LAST MESSAGE?
(2) 005356 001375                BNE  11$          ;;IF NOT: WAIT
(2) 005360 017637 000004 001122        MOV  @4(SP),$FATAL ;;GET ERROR #
(2) 005366 062766 000002 000004        ADD  #2,4(SP)     ;;BUMP RETURN ADDR.
(2) 005374 005237 001120                INC  $MSGTYPE     ;;TELL APT TO TAKE ERROR
(2) 005400 105037 005424 12$:          CLRB $FFLG       ;;CLEAR FATAL FLAG
(2) 005404 105037 005423                CLRB $LFLG       ;;CLEAR LOG FLAG
(2) 005410 105037 005422                CLRB $MFLG       ;;CLEAR MESSAGE FLAG
(4) 005414 012601                MOV  (SP)+,R1     ;;POP STACK INTO R1
(4) 005416 012600                MOV  (SP)+,R0     ;;POP STACK INTO R0
(2) 005420 000207                RTS  PC           ;;RETURN
(2) 005422 000                $MFLG: .BYTE 0    ;;MESSG. FLAG
(2) 005423 000                $LFLG: .BYTE 0
(2)                ;;LOG FLAG
(2) 005424 000                $FFLG: .BYTE 0    ;;FATAL FLAG
(2)                .EVEN
(2) 000200                APTSIZE=200
  
```

```

(2)          000001          APTENV=001
(2)          000100          APTSPool=100
(2)          000040          APTCSUP=040
(1)
(1)
(1)          :STRING INPUT ROUTINE
(1)          :-----
(1) 005426 010346          .INSTR: MOV      R3,-(SP)          :SAVE R3 ON STACK
(1) 005430 010446          MOV      R4,-(SP)          :SAVE R4 ON STACK
(1) 005432 017637 000004 005450  MOV      @4(SP),.MSG       :GET THE ADDRESS OF THE MESSAGE TO BE PRINTED
(1) 005440 062766 000002 000004  ADD      #2,4(SP)         :POINT TO INSTRUCTION AFTER ADDRESS POINTER
(1) 005446 104402          .INST1: TYPE              :PRINT THE MESSAGE
(1) 005450 000000          .MSG: 0                   :MESSAGE IS POINTED TO FROM HERE
(1) 005452 012704 010360          MOV      #INBUF,R4        :POINT R4 TO THE INPUT BUFFER
(1) 005456 012703 000007          MOV      #7,R3            :SET THE MAXIMUM NUMBER OF CHARACTERS ALLOWED
(1) 005462 105777 173622          1$:  TSTB   @$TKS         :HAS A CHARACTER BEEN RECEIVED?
(1) 005466 100375          BPL     1$                :IF NO, KEEP WAITING FOR IT
(1) 005470 117714 173616          MOV     @$TKB,(R4)        :IF YES, SAVE IT IN THE INPUT BUFFER
(1) 005474 142714 000200          BICB   #200,(R4)         :KEEP ONLY THE 7-BIT ASCII INFORMATION
(1) 005500 122427 000015          CMPB   (R4)+,#15         :IS THIS CHARACTER A LINE FEED?
(1) 005504 001417          BEQ    INSTR2            :IF SO, TERMINATE THE INPUT SEQUENCE
(1) 005506 105777 173602          2$:  TSTB   @$TSP         :IF NOT, CHECK TO SEE IF THE CHARACTER CAN PRINT
(1) 005512 100375          BPL     2$                :IF WE CAN'T, WAIT UNTIL WE CAN
(1) 005514 017777 173572 173574  MOV     @$TKB,$$TPB       :ECHO THE CHARACTER BACK
(1) 005522 005303          DEC     R3                :REDUCE THE NUMBER OF CHARACTERS RECEIVED
(1) 005524 001356          BNE    1$                :IF WE DON'T HAVE 7, GO GET SOME MORE
(1) 005526 012604          MOV     (SP)+,R4          :IF WE HAVE 7, RESTORE R4
(1) 005530 012603          MOV     (SP)+,R3          :RESTORE R3
(1) 005532 010346          .INSTE: MOV     R3,-(SP)   :SAVE R3 ON THE STACK
(1) 005534 010446          MOV     R4,-(SP)         :SAVE R4 ON THE STACK
(1) 005536 104402 001356          TYPE   .QUES             :PRINT A QUESTION MARK... WHAT'S GOING ON?
(1) 005542 000741          BR     .INST1            :GO PRINT THE MESSAGE AGAIN
(1) 005544 012604          INSTR2: MOV     (SP)+,R4   :RESTORE R4
(1) 005546 012603          MOV     (SP)+,R3         :RESTORE R3
(1) 005550 000002          RTI                      :RETURN TO THE MAIN PROCEDURE
(1)
(1)          :CONVERT ASCII STRING TO OCTAL
(1)          :-----
(1) 005552 010546          .PARAM: MOV     R5,-(SP)   :SAVE R5 ON THE STACK
(1) 005554 010446          MOV     R4,-(SP)         :SAVE R4 ON THE STACK
(1) 005556 016605 000004          MOV     4(SP),R5         :GET THE SETUP INFORMATION POINTER
(1) 005562 012537 005742          MOV     (R5)+,LOLIM      :SET THE LOW LIMIT FOR THE INPUT
(1) 005566 012537 005744          MOV     (R5)+,HILIM      :SET THE HIGH LIMIT FOR THE INPUT
(1) 005572 012537 005746          MOV     (R5)+,DEVADR     :SAVE THE ADDRESS WHERE THE RESULT WILL BE STORED
(1) 005576 112537 005750          MOV     (R5)+,LOBITS     :GET THE MASK OF THE INCORRECT BITS
(1) 005602 112537 005751          MOV     (R5)+,ADRCNT     :GET THE COUNT OF ITEMS TO BE STORED
(1) 005606 010566 000004          MOV     R5,4(SP)        :POINT TO WHERE MAIN LINE PROGRAM WILL RESUME
(1) 005612 005005          PARAM1: CLR     R5        :INITIALIZE THE ASCII TO OCTAL RESULT WORD
(1) 005614 012704 010360          MOV     #INBUF,R4        :POINT TO THE INPUT BUFFER
(1) 005620 122714 000015          CMPB   #15,(R4)         :IS THIS CHARACTER A CARRIAGE RETURN?
(1) 005624 001420          BEQ    PARERR           :IF SO, PRINT THE MESSAGE AGAIN
(1) 005626 121427 000060          1$:  CMPB   (R4),#60      :IS THIS CHARACTER BELOW THE NUMERIC RANGE?
(1) 005632 002415          BLT    PARERR           :IF SO, GO PRINT THE MESSAGE AGAIN
(1) 005634 121427 000067          CMPB   (R4),#67        :IS THIS CHARACTER ABOVE THE NUMERIC RANGE?
(1) 005640 003012          BGT    PARERR           :IF SO, GO PRINT THE MESSAGE AGAIN

```

```

(1) 005642 142714 000060      BICB   #60,(R4)      ;ISOLATE THE NUMBER THE CHARACTER REPRESENTS
(1) 005646 152405      BISB   (R4)+,R5      ;CONCATENATE THESE BITS TO THE ALREADY EXISTING STRING
(1) 005650 122714 000015      CMPB   #15,(R4)      ;IS THE NEXT CHARACTER A CARRIAGE RETURN?
(1) 005654 001406      BEQ    LIMITS        ;IF SO, GO SEE IF NUMBER IS WITHIN LIMITS
(1) 005656 006305      ASL   R5              ;CLEAR BIT POSITION 0, MOVE EXISTING STRING TO LEFT
(1) 005660 006305      ASL   R5              ;CLEAR POSITION 1, MOVE STRING TO LEFT AGAIN
(1) 005662 006305      ASL   R5              ;MOVE THE STRING ONE MORE TIME TO MAKE ROOM FOR
(1)                                ;NEXT THREE BITS
(1) 005664 000760      BR     1$             ;GO GET THE NEXT CHARACTER
(1) 005666 104404      PARERR: INSTER        ;THERE WAS AN ERROR... GO PRINT MESSAGE AGAIN
(1) 005670 000750      BR     PARAM1        ;TRY GETTING THE PARAMETERS AGAIN
(1)                                ;TEST TO SEE IF NUMBER IS WITHIN LIMITS
(1)                                ;-----
(1) 005672 020537 005744      LIMITS: CMP R5,HILIM   ;DOES RESULT EXCEED ITS MAXIMUM CORRECT VALUE?
(1) 005676 101373      BHI   PARERR         ;IF YES, GO PRINT THE MESSAGE AGAIN
(1) 005700 020537 005742      CMP   R5,LOLIM       ;IS THE RESULT LOWER THAN ALLOWED?
(1) 005704 103770      BLO   PARERR         ;IF YES, GO PRINT THE MESSAGE AGAIN
(1) 005706 133705 005750      BITB  LOBITS,R5      ;ARE ANY INCORRECT BITS SET IN THE RESULT?
(1) 005712 001365      BNE   PARERR         ;IF SO, GO PRINT THE MESSAGE AGAIN
(1)                                ;STORE NUMBER AT SPECIFIED ADDRESS
(1) 005714 013704 005746      1$:  MOV  DEVADR,R4    ;POINT TO THE LOCATION WHERE THE RESULT WILL BE STORED
(1) 005720 010524      MOV  R5,(R4)+        ;STORE THE RESULT
(1) 005722 062705 000002      ADD  #2,R5           ;CALCULATE THE NEXT DATUM
(1) 005726 105337 005751      DECB ADRCNT          ;REDUCE COUNT OF STORED RESULTS. IS IT EXCEEDED?
(1) 005732 001372      BNE  1$              ;IF NOT, GO STORE THE NEXT DATUM
(1) 005734 012604      MOV  (SP)+,R4        ;RESTORE R4
(1) 005736 012605      MOV  (SP)+,R5        ;RESTORE R5
(1) 005740 000002      RTI                  ;RETURN TO THE MAIN PROGRAM
(1) 005742 000000      LOLIM: 0              ;LOWEST ACCEPTABLE VALUE
(1) 005744 000000      HILIM: 0              ;HIGHEST ACCEPTABLE
(1) 005746 000000      DEVADR: 0             ;LOCATION WHERE RESULT WILL BE STORED
(1) 005750 000      LOBITS: .BYTE 0      ;INCORRECT BITS MASK
(1) 005751 000      ADRCNT: .BYTE 0      ;COUNT OF ITEMS TO BE STORED
(1)                                ;SAVE PC OF TEST THAT FAILED AND R0-R5
(1)                                ;-----
(1) 005752 016637 000004 001404 .SAV05: MOV  4(SP),SAVPC   ;SAVE R7 (PC)
(1)                                ;SAVE R0-R5
(1) 005760 010537 001340      SV05: MOV  R5,$REG5    ;SAVE R5
(1) 005764 010437 001336      MOV  R4,$REG4        ;SAVE R4
(1) 005770 010337 001334      MOV  R3,$REG3        ;SAVE R3
(1) 005774 010237 001332      MOV  R2,$REG2        ;SAVE R2
(1) 006000 010137 001330      MOV  R1,$REG1        ;SAVE R1
(1) 006004 010037 001326      MOV  R0,$REG0        ;SAVE R0
(1) 006010 000002      RTI                  ;LEAVE.
(1)                                ;RESTORE R0-R5

```

```

(1) 006012 013700 001326 .RES05: MOV $REG0,R0 ;RESTORE R0
(1) 006016 013701 001330 MOV $REG1,R1 ;RESTORE R1
(1) 006022 013702 001332 MOV $REG2,R2 ;RESTORE R2
(1) 006026 013703 001334 MOV $REG3,R3 ;RESTORE R3
(1) 006032 013704 001336 MOV $REG4,R4 ;RESTORE R4
(1) 006036 013705 001340 MOV $REG5,R5 ;RESTORE R5
(1) 006042 000002 RTI ;LEAVE
(1)
(1) ;CONVERT OCTAL NUMBER TO ASCII AND OUTPUT TO TELEPRINTER
(1) -----
(1)
(1) 006044 104402 001357 .CONVR: TYPE ,$CRLF ;PRINT A CARRIAGE RETURN
(1) 006050 010046 .CNVRT: MOV R0,-(SP) ;SAVE R0
(1) 006052 010146 MOV R1,-(SP) ;SAVE R1
(1) 006054 010346 MOV R3,-(SP) ;SAVE R3
(1) 006056 010446 MOV R4,-(SP) ;SAVE R4
(1) 006060 010546 MOV R5,-(SP) ;SAVE R5
(1) 006062 017601 000012 MOV @12(SP),R1 ;PLACE THE ADDRESS OF THE ARGUMENTS IN R1
(1) 006066 062766 000002 000012 ADD #2,12(SP) ;POINT TO WHERE MAIN PROGRAM WILL RESUME
(1) 006074 012137 006220 MOV (R1)+,WRDCNT ;GET NUMBER OF WORDS TO BE PRINTED
(1) 006100 112105 1$: MOV (R1)+,R5 ;GET THE NUMBER OF CHARACTERS TO BE PRINTED
(1) 006102 112100 MOV (R1)+,R0 ;GET THE NUMBER OF SPACES TO PRINT
(1) 006104 013104 MOV @ (R1)+,R4 ;COPY THE WORD TO BE CONVERTED
(1) 006106 110537 006222 MOV R5,CHRCNT ;COPY THE CHARACTER COUNT
(1) 006112 010403 3$: MOV R4,R3 ;COPY THE ARGUMENT WORD AGAIN
(1) 006114 042703 177770 BIC #<C<7>,R3 ;ISOLATE THREE BITS TO BE TREATED AS A CHARACTER
(1) 006120 062703 000060 ADD #060,R3 ;MAKE AN ASCII CHARACTER OUT OF THEM
(1) 006124 110346 MOV R3,-(SP) ;SAVE THAT CHARACTER
(1) 006126 006004 ROR R4 ;MOVE THE NEXT THREE BITS INTO PLACE
(1) 006130 006204 ASR R4 ;MOVE THEM AGAIN
(1) 006132 006204 ASR R4 ;AND FINALLY A THIRD TIME
(1) 006134 005305 DEC R5 ;REDUCE CHARACTER COUNT. ARE ALL CHARACTERS
(1) ;BUILT?
(1) 006136 001365 BNE 3$ ;IF NO, GO BUILD THE NEXT ONE.
(1) 006140 012703 010422 MOV #MDATA,R3 ;NOW POINT TO WHERE NUMBER WILL BE PRINTED FROM
(1) 006144 112623 4$: MOV (SP)+,(R3)+ ;STORE THE CHARACTER, STARTING WITH THE MOST
(1) 006146 105337 006222 DEC CHRCNT ;REDUCE COUNT. ARE ALL CHARACTERS TRANSFERRED?
(1) 006152 001374 BNE 4$ ;IF NO, GO TRANSFER ANOTHER
(1) 006154 105700 TSTB R0 ;ARE ANY SPACES TO BE PRINTED?
(1) 006156 001404 BEQ 6$ ;IF NO, DON'T SET UP ANY
(1) 006160 112723 000040 5$: MOV #040,(R3)+ ;ADD A SPACE TO THE OUTPUT BUFFER
(1) 006164 105300 DEC R0 ;REDUCE THE COUNT. SHOULD WE PRINT MORE?
(1) 006166 001374 BNE 5$ ;IF YES, GO ADD ANOTHER SPACE
(1) 006170 105013 6$: CLRB (R3) ;TERMINATE THE OUTPUT BUFFER WITH A ZERO
(1) 006172 104402 010422 TYPE ,MDATA ;PRINT THE STRING WE JUST BUILT
(1) 006176 005337 006220 DEC WRDCNT ;REDUCE THE WORD COUNT. ARE ANY MORE WORDS LEFT?
(1) 006202 001336 BNE 1$ ;IF YES, GO CONVERT THEM
(1) 006204 012605 MOV (SP)+,R5 ;RESTORE R5
(1) 006206 012604 MOV (SP)+,R4 ;RESTORE R4
(1) 006210 012603 MOV (SP)+,R3 ;RESTORE R3
(1) 006212 012601 MOV (SP)+,R1 ;RESTORE R1
(1) 006214 012600 MOV (SP)+,R0 ;RESTORE R0
(1) 006216 000002 RTI ;RETURN TO THE MAIN PROGRAM
(1) 006220 000000 WRDCNT: 0
(1) 006222 000 CHRCNT: .BYTE
(1) 006223 000 SPACNT: .BYTE 0

```


39

CNDZA-A MACY11 30(1046) 15-DEC-82 14:36 PAGE 81-33
CNDZAA.F11 15-DEC-82 14:31 APT COMMUNICATIONS ROUTINE

N 4

SEQ 0052

(1) 006350 000002
(1)

1\$: RTI

;RETURN TO THE PRESENT TEST


```

(1)                                     ;LINE PARAMETER REGISTER SETUP ROUTINE
(1)
(1) 006352 010146 .LPRSET:MOV R1,-(SP) ;SAVE CONTENTS OF R1
(1) 006354 010246 MOV R2,-(SP) ;SAVE CONTENTS OF R2
(1) 006356 013701 001370 MOV PAR,R1 ;MOVE DEFAULT PARAM. INTO R1
(1) 006362 012702 000001 MOV #1,R2 ;INIT. FOR LINE 1
(1) 006366 010177 173426 1$: MOV R1,@DZVLPR ;LOAD PARAM. REGISTER
(1) 006372 005201 INC R1 ;SET R1 FOR NEXT LINE
(1) 006374 106302 ASLB R2 ;SET R2 FOR NEXT LINE
(1) 006376 032702 000020 BIT #BIT4,R2 ;ALL LINES DONE?
(1) 006402 001771 BEQ 1$ ;IF NO LOAD NEXT LINE
(1) 006404 012602 MOV (SP)+,R2 ;RELOAD R2
(1) 006406 012601 MOV (SP)+,R1 ;RELOAD R1
(1) 006410 000002 RTI ;RETURN
(1)
(1)                                     ;ROUTINE TO ZERO DATA BUFFER
(1)
(1) 006412 010046 .BUFSET:MOV R0,-(SP) ;SAVE CONTENTS OF R0
(1) 006414 012700 001426 MOV #TDO,R0 ;SET R0 TO TOP OF BUFFER
(1) 006420 005020 1$: CLR (R0)+ ;CLEAR BUFFER LOCATION
(1) 006422 022700 001446 CMP #STOP,R0 ;IS BUFFER ALL CLEARED
(1) 006426 001374 BNE 1$ ;IF NOT CLEAR NEXT LOCATION
(1) 006430 012600 MOV (SP)+,R0 ;RELOAD R0
(1) 006432 000002 RTI ;RETURN
(1)
(1)                                     ;ERROR HANDLER
(1)
(1)                                     :-----
(1)
(1) 006434 004737 007062 $ERROR: JSR PC,SERV.G ;FIND OUT IF <^G> WAS HIT
(1) 006440 032777 010000 172636 BIT #SW12,@SWR ;BELL ON ERROR?
(1) 006446 001406 BEQ XBX ;BR IF NO BELL
(1) 006450 105777 172640 TSTB @STPS ;TTY READY.
(1) 006454 100003 BPL XBX ;DON'T WAIT IF TTY NOT READY.
(1) 006456 112777 000207 172632 MOVB #207,@STPB ;PUSH A BELL AT THE TTY.
(1) 006464 032777 020000 172612 XBX: BIT #SW13,@SWR ;DELETE ERROR PRINT OUT?
(1) 006472 001113 BNE HALTS ;BR IF NO PRINT OUT WANTED.
(1) 006474 021637 001262 CMP (SP),$ERRPC ;WAS THIS ERROR FOUND LAST TIME?
(1) 006500 001404 BEQ 1$ ;BR IF YES
(1) 006502 011637 001262 MOV (SP),$ERRPC ;RECORD BEING HERE
(1) 006506 105037 001247 CLRB $ERFLG ;PREPARE HEADER
(1) 006512 104407 1$: SAVO5 ;SAVE ALL PROC REGISTERS
(1) 006514 011605 MOV (SP),R5 ;GET THE PC OF ERROR
(1) 006516 162705 000002 SUB #2,R5 ;GET ADDRESS OF TRAP CALL
(1) 006522 011504 MOV (R5),R4 ;GET ERROR INSTRUCTION
(1) 006524 110437 001260 MOVB R4,$ITEMB ;COPY TEST NUMBER FOR APT HANDLING
(1) 006530 006304 ASL R4 ;MULT BY TWO
(1) 006532 061504 ADD (R5),R4 ;DOUBLE IT
(1) 006534 006304 ASL R4 ;MULT AGAIN
(1) 006536 042704 177001 BIC #177001,R4 ;CLEAR JUNK
(1) 006542 062704 016034 ADD #.ERRTAB,R4 ;GET POINTER
(1) 006546 012437 006672 MOV (R4)+,ERRMSG ;GET ERROR MESSAGE
(1) 006552 012437 006704 MOV (R4)+,DATAHD ;GET DATA HEADRER
(1) 006556 011437 006716 MOV (R4),DATABP ;GET DATA TABLE
(1) 006562 105737 001247 TSTB $ERFLG ;TYPE HEADER
(1) 006566 001403 BEQ TYPMSG ;BR IF YES
(1) 006570 005737 006716 TST DATABP ;DOES DATA TABLE EXIST?

```



```

(1) 007062 017746 172224 SERV.G: MOV @STKB,-(SP) ;OTHERWISE, GET THE LAST CHARACTER TYPED
(1) 007066 042716 000200 BIC #BIT7,(SP) ;STRIP PARITY(EIGHTH) BIT
(1) 007072 122726 000007 CMPB #7,(SP)+ ;IS IT ^G?
(1) 007076 001076 BNE 6$ ;IF NOT, IGNORE INPUT
(1) 007100 032777 004000 172202 BIT #4000,@STKS ;RX BUSY?
(1) 007106 001365 BNE SERV.G ;BR IF YES
(1) 007110 007110 GETSWR= ;GPA
(1) 007110 017737 172170 007316 MOV @SWR,90$ ;SAVE (SWR).
(1) 007116 104402 007276 1$: TYPE ,89$ ;TYPE HEADER FOR OLD SWITCH REGISTER
(1) 007122 104412 007310 CNVRT ,88$ ;TYPE THE NUMBER ITSELF
(1) 007126 104402 007320 TYPE ,91$ ;AFTER HAVING CONVERTED IT TO ASCII
(1) 007132 105037 007324 CLRB 92$ ;CLEAR SWR CHANGE FLAG
(1) 007136 005077 172142 CLR @SWR ;CLEAR THE SOFTWARE SWITCH REGISTER
(1) 007142 105777 172142 3$: TSTB @STKS ;WAIT FOR DONE.
(1) 007146 100375 BPL 3$ ;CONTINUE WAITING FOR IT
(1) 007150 017746 172136 MOV @STKB,-(SP) ;PUT THE CHARACTER ON THE STACK
(1) 007154 042716 000200 BIC #BIT7,(SP) ;STRIP PARITY BIT
(1) 007160 122726 000015 CMPB #15,(SP)+ ;IS IT THE CARRIAGE RETURN CHAR?
(1) 007164 001433 BEQ 4$ ;IF SO, GO PRINT CRLF
(1) 007166 105777 172122 2$: TSTB @STPS ;IS THE OUTPUT BUFFER AVAILABLE
(1) 007172 100375 BPL 2$ ;IF NOT, WAIT FOR IT TO BE READY
(1) 007174 105237 007324 INCB 92$ ;INDICATE THAT THE SWR WAS CHANGED
(1) 007200 014677 172112 MOV -(SP),@STPB ;PLACE THE CHARACTER THERE(ECHO BACK)
(1) 007204 000241 CLC ;GET READY TO ROTATE
(1) 007206 006177 172072 ROL @SWR ;MOVE THE EXISTING BITS OVER
(1) 007212 006177 172066 ROL @SWR ;TO MAKE ROOM FOR THE INCOMING
(1) 007216 006177 172062 ROL @SWR ;THREE BITS FROM THIS CHARACTER
(1) 007222 103735 BCS 1$ ;ERROR
(1) 007224 022627 000060 CMP (SP)+,#60 ;IS IT LOWER THAN 0?
(1) 007230 002732 BLT 1$ ;IF SO, GO ASK AGAIN
(1) 007232 026627 177776 000067 CMP -2(SP),#67 ;IS IT HIGHER THAN 7?
(1) 007240 003326 BGT 1$ ;IF SO, GO ASK AGAIN
(1) 007242 042746 177770 BIC #^C<?>,-(SP) ;ISOLATE INFORMATION BITS
(1) 007246 052677 172032 BIS (SP)+,@SWR ;ADD THEM TO THE SWITCH REGISTER
(1) 007252 000733 BR 3$ ;GO CHECK FOR THE NEXT CHARACTER
(1) 007254 105737 007324 4$: TSTB 92$ ;HAS THE SWR BEEN CHANGED?
(1) 007260 001003 BNE 5$ ;IF YES GO TYPE CRLF
(1) 007262 013777 007316 172014 MOV 90$,@SWR ;IF NOT RESTORE SWR
(1) 007270 104402 001357 5$: TYPE ,CRLF ;TYPE A CARRIAGE RETURN AND LINE FEED
(1) 007274 000207 6$: RTS PC ;RETURN TO CALLING PROCEDURE
(1) 007276 020200 051450 051127 89$: .ASCIZ <200>? (SWR)=/?
(1) 007304 036451 000057 .EVEN
(1) 007310 000001 88$: 1
(1) 007312 006 000 .BYTE 6,0
(1) 007314 007316 90$: .WORD 0
(1) 007316 000000 91$: .ASCIZ ?/=/?
(1) 007320 036457 000057 92$: .BYTE 0
(1) 007324 000 .EVEN
(2) .SBTTL POWER DOWN AND UP ROUTINES
(2)
(3)
(2)
(2) 007326 012737 007472 000024 $PWRDN: MOV #SILLUP,@PWRVEC ;;SET FOR FAST UP
  
```



```

(2)                                     ;COMPARE THE FIRST CHARACTER IN THE TELETYPE INPUT
(2)                                     ;BUFFER TO THE CHARACTERS 'E' AND 'C'.
(2)                                     ;IF THE CHARACTER IS 'E' CLEAR THE FLAG
(2)                                     ;IF THE CHARACTER IS 'C' SET THE FLAG
(2) 010304 017605 000000                .PAWCH:MOV      @ (SP),R5
(2) 010310 142737 000040 010360        BICB      #40,INBUF      ;SET FOR LOWER CASE INPUT
(2) 010316 122737 000105 010360        CMPB      #'E',INBUF    ;IS IT 'E' ?
(2) 010324 001002                       BNE       1$
(2) 010326 105015                       CLRB      (R5)          ;000
(2) 010330 000406                       BR        2$
(2) 010332 122737 000103 010360 1$:    CMPB      #'C',INBUF    ;IS IT 'C' ?
(2) 010340 001005                       BNE       3$
(2) 010342 112715 177777                MOVB      #-1,(R5)     ;3177
(2) 010346 062716 000002                2$:      ADD      #2,(SP)
(2) 010352 000002                       RTI
(2) 010354 104404                       3$:      INSTER                      ;RETRY
(2) 010356 000752                       BR        .PAWCH

(2)                                     ;BUFFERS FOR INPUT-OUTPUT
(2)
(2) 010360 000000                INBUF: 0
(2) 010422 010422                .=.+40
(2)                                     : TEMP: 0                ; TEMP AREA UNUSED.
(2)                                     : .=.+40                ; DELETED TO CONSERVE SPACE
(2) 010422 000000                MDATA: 0
(2) 010464 010464                .=.+40
  
```

::GPA
 ::GPA

```

(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2)
(2) 010464 005737 001406          CYCLE: TST      DZVACTV      ;ARE ANY DZV11'S TO BE TESTED?
(2) 010470 001004                BNE      1$          ;BR IF OK.
(2) 010472 104402 007603          TYPE     ,MERR2     ;NO DZV11'S SELECTED!!
(2) 010476 000000                HALT                    ;STOP THE SHOW.
(2) 010500 000776                BR      .-2         ;DISQUALIFY CONT. SW.
(2) 010502 013737 004632 001354 1$: MOV      $MXCNT,$TIMES ;RESTORE THE NUMBER OF ITERATIONS TO MAKE
(2) 010510 033737 001412 001406 BIT      RUN,DZVACTV ;IS THIS ONE "ACTIVE"
(2) 010516 001017                BNE     2$          ;BR IF GOOD ONE FOUND.
(2) 010520 006137 001412          ROL     RUN         ;UPDATE POINTER
(2) 010524 005537 001412          ADC     RUN         ;CATCH CARRY FROM RUN
(2) 010530 062737 000012 001420 ADD     #12,ACTIVE  ;UPDATE ADDRESS POINTER.
(2) 010536 022737 001740 001420 CMP     #DZV.END,ACTIVE ;HAVE WE PASSED THE END OF THE MAP?
(2) 010544 001356                BNE     1$          ;IF NO, KEEP GOING; NOT ALL TESTED FOR.
(2) 010546 012737 001500 001420 MOV     #DZV.MAP,ACTIVE ;RESET ADDRESS POINTER.
(2) 010554 000752                BR      1$          ;KEEP LOOKING FOR ACTIVE DZV11
(2) 010556 006137 001412          ROL     RUN         ;UPDATE POINTER.
(2) 010562 005537 001412          ADC     RUN         ;CATCH CARRY.
(2) 010566 013700 001420          MOV     ACTIVE,RO  ;GET ADDRESS POINTER.
(2) 010572 062737 000012 001420 ADD     #12,ACTIVE  ;UPDATE.
(2) 010600 022737 001740 001420 CMP     #DZV.END,ACTIVE
(2)
(2) 010606 001003                BNE     3$          ;ALL DONE?
(2) 010610 012737 001500 001420 MOV     #DZV.MAP,ACTIVE ;BR IF NO.
(2) 010616 012037 001174          MOV     (RO)+,$BASE ;RESTORE POINTER.
(2) 010622 012037 002040          MOV     (RO)+,DZVRIV ;LOAD SYSTEM CTRL. REG
(2) 010626 012037 001366          MOV     (RO)+,LINE  ;LOAD VECTOR
(2) 010632 012037 001370          MOV     (RO)+,PAR   ;SET UP DZV LINES ACTIVE
(2) 010636 012037 001372          MOV     (RO)+,MODE  ;SET UP PARAMETERIZATION
(2) 010642 105037 001424          CLRB   MNTFLG ;RESET MAINT. FLAG IF
(2) 010646 005737 001372          TST     MODE       ;RUNNING TESTS
(2) 010652 001003                BNE     9$          ;IN
(2) 010654 112737 000010 001424 MOVB    #MAINT,MNTFLG ;INTERNAL MAINT. MODE
(2) 010662 004737 011030          JSR    PC,DZVLEV   ;SET UP
(2) 010666 005737 000042          TST    @#42        ;ARE WE UNDER MONITOR CONTROL?
(2) 010672 001051                BNE     7$          ;IF YES, SKIP THIS SETUP
(2) 010674 032777 000002 170402 BIT     #SW01,@SWR  ;IF SW01=1, GET STARTING TEST #
(2) 010702 001445                BEQ     7$          ;BR IF NO TEST IS TO BE INPUTTED
(2) 010704 104402 001357          TYPE   ,SRLF
(3) 010710 104403                INSTR
(3) 010712 007763                MTSTN
(3) 010714 104405                PARAM
(3) 010716 000001                1
(3) 010720 001000                1000
(3) 010722 001246                $STNM
(3) 010724 000                .BYTE 0
(3) 010725 001                .BYTE 1
(2) 010726 012700 012024          MOV     #TST1,RO

```

```

(2) 010732 022710 000004      5$:  CMP      #4,(R0)
(2) 010736 001020              BNE      6$
(2) 010740 022760 012737 000002  CMP      #12737,2(R0)
(2) 010746 001014              BNE      6$
(2) 010750 023760 001246 000004  CMP      $STNM,4(R0)      :IS THIS THE TEST ?
(2) 010756 001010              BNE      6$              :IF NOT, DON'T PROCESS NUMBER
(2) 010760 010037 001252      MOV      R0,$LPADR        :SAVE PC
(2) 010764 062737 000002 001252  ADD      #2,$LPADR        :POP OVER PREVIOUS SCOPE
(2) 010772 104402 001357      TYPE    $CRLF
(2) 010776 000412              BR       8$
(2) 011000 005720              6$:  TST      (R0)+
(2) 011002 020027 015502      CMP      R0,#TLAST+10
(2) 011006 001351              BNE      5$
(2) 011010 104402 001356      TYPE    $QUES
(2) 011014 000733              BR       4$
(2) 011016 012737 012024 001252  7$:  MOV      #TST1,$LPADR    :PREPARE TEST ADDRESS
(2) 011024              8$:
(2) 011024 000177 170222      RESTART:JMP  @ $LPADR      :GO START TESTING.***WARNING!***
(2)
(2)
(2)
(2) 011030 013700 002040      :THIS UTILITY SETS UP CSR'S,SETS UP VECTORS.
DZVLEV: MOV      DZVRIV,R0      :PLACE THE BASE VECTOR ADDRESS IN R0
(2) 011034 062700 000002      ADD      #2,R0              :CALCULATE THE RECEIVER INTERRUPT STATUS ADDR.
(2) 011040 010037 002042      MOV      R0,DZVRIS          :STORE IT HERE
(2) 011044 062700 000002      ADD      #2,R0              :CALCULATE THE TRANSMITTER INTERRUPT VECTOR
(2) 011050 010037 002044      MOV      R0,DZVTIV          :STORE IT HERE
(2) 011054 062700 000002      ADD      #2,R0              :CALCULATE THE TRANSMITTER VECTOR STATUS ADDRESS
(2) 011060 010037 002046      MOV      R0,DZVTIS          :STORE IT HERE
(2)
(2)
(2)
(2)
(2)
(2)
(2) 011064 013700 001174      :THIS SEGMENT SETS UP POINTERS FOR THE GIVEN DZV11. $BASE IS THE BASE ADDRESS
(2) 011070 010037 002010      :OF THE DEVICE
MOV      $BASE,R0              :COPY THE ADDRESS BEING LOADED
(2) 011074 005200              MOV      R0,DZVCSR          :XXX0
(2) 011076 010037 002012      INC      R0
MOV      R0,HDZVCSR           :XXX1
(2) 011102 005200              INC      R0
MOV      R0,DZVRBUF          :XXX2
(2) 011104 010037 002014      MOV      R0,DZVLPR          :XXX2
(2) 011110 010037 002020      MOV      R0,HDZVRBUF       :XXX3
(2) 011114 005200              INC      R0
MOV      R0,HDZVLPR          :XXX3
(2) 011116 010037 002016      MOV      R0,DZVTICR        :XXX4
(2) 011122 010037 002022      INC      R0
MOV      R0,HDZVTICR        :XXX5
(2) 011126 005200              MOV      R0,DZVMSR         :XXX6
(2) 011130 010037 002024      INC      R0
MOV      R0,DZVTDR          :XXX6
(2) 011134 005200              MOV      R0,HDZVMSR        :XXX7
(2) 011136 010037 002026      INC      R0
MOV      R0,HDZVTDR         :XXX7
(2) 011142 005200              MOV      R0,PC
(2) 011144 010037 002030      RTS
(2) 011150 010037 002034
(2) 011154 005200
(2) 011156 010037 002032
(2) 011162 010037 002036
(2) 011166 000207
  
```


CNDZA-A MACY11 30(1046) 15-DEC-82 14:36 PAGE 81-43 K 5
CNDZAA.F11 15-DEC-82 14:31 POWER DOWN AND UP ROUTINES

SEQ 0062

(2)

; END OF .PARMD DELETE RANGE

&

::GPA

```

(2)                                     ;*ROUTINE USED TO SET UP THE DIAGNOSTIC VIA APT.
(2)                                     ;*IF BIT7 IN THE ENVIRONMENT MODE ($ENVM) BYTE IS SET,
(2)                                     ;*THE PROGRAM WILL LOAD ITS PARAMETERS FROM THE ETABLE.
(2)
(2) 011172 012700 001500      SETAPT: MOV    #DZV.MAP,R0      ;POINT TO THE DEVICE MAP TABLE
(2) 011176 013701 001174      MOV    $BASE,R1      ;BUILD DEVICE ADDRESSES IN R1
(2) 011202 013702 001170      MOV    $VECT1,R2     ;BUILD DEVICE VECTORS IN R2
(2) 011206 042702 177007      BIC    #*C<770>,R2   ;STRIP AWAY OTHER INFORMATION
(2)
(2) 011212 012704 001204      MOV    #SDDW0,R4     ;POINT TO THE BEGINNING OF DEVICE PARAMETERS
(2) 011216 013705 001176      MOV    $DEVN,R5     ;GET THE MAP OF ACTIVE DEVICES
(2) 011222 105037 001414      CLRB  DZVNUM        ;INITIALIZE NO. OF DEVICES IN SYSTEM
(2) 011226 005037 001410      CLR   SAVACTV       ;CLEAR THE ACTIVE BIT MAP
(2) 011232 006005              1$:  ROR    R5          ;GET A DEVICE SELECTION BIT
(2) 011234 103407              BCS   3$            ;IF IT IS SELECTED, GO SET UP A MAP
(2) 011236 001422              BEQ   5$            ;IF NO MORE ARE SELECTED, GET OUT OF SETUP
(2) 011240 005724              TST   (R4)+         ;POINT TO NEXT DEVICE DESCRIPTOR
(2) 011242 062701 000010      2$:  ADD   #10,R1     ;SET UP THE NEXT ADDRESS
(2) 011246 062702 000010      ADD   #10,R2        ;SET UP THE NEXT VECTOR GROUP
(2) 011252 000767              BR    1$            ;GO SEE IF MORE DEVICES REMAIN
(2) 011254 006137 001410      3$:  ROL   SAVACTV    ;SET BIT IN ACTIVE DEVICE MAP
(2) 011260 105237 001414      INCB  DZVNUM        ;INCREMENT NO. OF ACTIVE DEVICES IN SYSTEM
(2) 011264 010120              MOV   R1,(R0)+     ;LOAD DEVICE ADDRESS
(2) 011266 010220              MOV   R2,(R0)+     ;LOAD THE VECTOR ADDRESS
(2) 011270 013720 001200      MOV   $CDW1,(R0)+  ;GET THE NUMBER OF LINES IN OPERATION
(2) 011274 012420              MOV   (R4)+,(R0)+  ;LOAD DEVICE PARAMETERS
(2) 011276 013720 001202      MOV   $CDW2,(R0)+  ;LOAD DEFAULT TESTING MODE
(2) 011302 000757              BR    2$            ;GO BUILD THE NEXT ADDRESS
(2) 011304 012710 177777      5$:  MOV   #-1,(R0)   ;TERMINATE THE DEVICE MAP
(2) 011310 012737 001142 001304  MOV   #$$SWREG,SWR ;SET TO SOFTWARE APT SWITCH REGISTER
(2) 011316 000207              RTS   PC            ;RETURN TO PRINT STATUS TABLE

```

```

(2)
(2)                                     ;*ROUTINE USED TO "AUTO SIZE" THE DZV11
(2)                                     ;*CSR AND VECTOR.
(2)                                     ;*NOTE: THE CSR MAY BE ANY WHERE IN THE FLOATING
(2)                                     ;*      ADDRESS RANGE (160000:163770)
(2)                                     ;*      AND THE VECTOR MAY BE ANY WHERE IN THE
(2)                                     ;*      FLOATING VECTOR RANGE (300:770)
(2)                                     ;*
(2)
(2) 011320              AUTO.SIZE:
(2) 011320 000005              RESET              ;INSURE A BUS INIT.
(2) 011322 105337 001422      DECB  INIFLG        ;SHOW THAT I WAS HERE
(2) 011326 012702 001500      CSRMAP: MOV #DZV.MAP,R2 ;LOAD MAP POINTER.
(2) 011332 012703 001204      MOV   #SDDW0,R3    ;POINT TO ETABLE DEVICE DESCRIPTOR WORDS
(2) 011336 005022              1$:  CLR   (R2)+       ;ZERO ENTIRE MAP
(2) 011340 022702 001740      CMP   #DZV.END,R2 ;ALL DONE?
(2) 011344 001374              BNE   1$            ;BR IF NO
(2) 011346 105037 001414      CLRB  DZVNUM        ;SET OCTAL NUMBER OF DZV11'S TO 0
(2) 011352 012702 001500      MOV   #DZV.MAP,R2
(2) 011356 012701 160000      MOV   #160000,R1   ;SET FOR FIRST ADDRESS TO BE TESTED
(2) 011362 012737 011626 000004  MOV   #6$,R4        ;SET FOR NON-EXISTENT DEVICE TIME OUT
(2) 011370 052711 000040      2$:  BIS   #BIT5,(R1)  ;TRY TO SET MASTER SCAN ENABLE
(2) 011374 052761 000017 000004  BIS   #17,4(R1)    ;TRY TO TRANSMIT ON ANY LINE
(2) 011402 005000              CLR   R0            ;USE R0 AS A COUNTER

```

```

(2) 011404 005711      7$:  TST      (R1)      ;HAS TRANSMITTER READY COME UP?
(2) 011406 100403      BMI      8$      ;IF SO, GO GET A FINAL CHECK
(2) 011410 005300      DEC      R0      ;REDUCE COUNT. TIME UP?
(2) 011412 001374      BNE      7$      ;IF NOT, KEEP WAITING
(2) 011414 000437      BR       3$      ;ASSUME IT'S NOT A DZV11
(2) 011416 032761 000017 000004 8$:  BIT      #17,4(R1) ;ARE ANY TCR BITS STILL SET? THEY SHOULD BE
(2) 011424 001433      BEQ      3$      ;IF IT'S NOT, ASSUME IT'S NOT A DZV11
(2) 011426 032711 000040      BIT      #BIT5,(R1) ;IS MASTER SCAN ENABLE STILL SET?
(2) 011432 001430      BEQ      3$      ;IF NOT, ASSUME IT'S NOT A DZV11
(2) 011434 052711 000020      BIS      #20,(R1) ;SET DEVICE CLEAR
(2) 011440 000240      NOP
(2) 011442 032711 000040      BIT      #40,(R1) ;DID SCANNER CLEAR
(2) 011446 001022      BNE      3$      ;IF NOT ASSUME IT IS NOT DZV
(2) 011450 005061 000004      CLR      4(R1)    ;GET RID OF TCR BITS
(2)                                ;AT THIS POINT IT IS ASSUMED THAT R1 HOLDS A DZV11 CSR ADDRESS.
(2) 011454 010122      MOV      R1,(R2)+ ;STORE CSR IN CORE TABLE.
(2) 011456 005722      TST      (R2)+    ;POP OVER VECTOR STORE AREA
(2) 011460 012722 000017      MOV      #17,(R2)+ ;SET THE DEFAULT LINE SELECTION PARAMETER
(2) 011464 012712 017470      MOV      #17470,(R2) ;SET THE DEFAULT PARAMETERS
(2) 011470 012223      MOV      (R2)+,(R3)+ ;COPY PARAMETERS INTO ETABLE DESCRIPTOR
(2) 011472 005022      CLR      (R2)+    ;SET THE DEFAULT MODE OF OPERATION
(2) 011474 012712 177777      MOV      #-1,(R2) ;TERMINATE LIST
(2) 011500 105237 001414      INCB    DZVNUM    ;UPDATE DEVICE COUNTER
(2) 011504 122737 000020 001414      CMPB   #20,DZVNUM ;ARE MAX. NO. OF DEV FOUND?
(2) 011512 001405      BEQ     100$     ;YES DON'T LOOK FOR ANY MORE.
(2) 011514 062701 000010      3$:  ADD     #10,R1  ;UPDATE CSR POINTER ADDRESS
(2) 011520 022701 164000      CMP     #164000,R1
(2) 011524 001321      BNE     2$      ;BR IF MORE ADDRESS TO CHECK.
(2) 011526                                100$:
(2) 011526 105737 001414      TSTB   DZVNUM    ;WERE ANY DZV11'S FOUND AT ALL?
(2) 011532 001430      BEQ     5$      ;ERROR AUTO SIZER FOUND NO DZV11'S IN THIS SYS.
(2) 011534 113701 001414      MOVB   DZVNUM,R1
(2) 011540 012737 000001 001410      MOV     #1,SAVACTV ;CREATE A BIT MAP OF THE ACTIVE
(2) 011546 005301      4$:  DEC     R1      ;DEVICES IN THE SYSTEM
(2) 011550 001404      BEQ     98$     ;
(2) 011552 000261      SEC
(2) 011554 006137 001410      ROL    SAVACTV
(2) 011560 000772      BR     4$
(2) 011562 013737 001500 001174 98$:  MOV     DZCRO,$BASE ;POINT TO THE ADDRESS OF FIRST DEVICE
(2) 011570 013737 001510 001202      MOV     MANTO,$CDW2 ;INDICATE TO ETABLE WHAT MODE IS BEING USED
(2) 011576 012737 000006 000004 99$:  MOV     #6,@#4    ;RESTORE TRAP VECTOR
(2) 011604 013737 001410 001176      MOV     SAVACTV,$DEV ;SAVE ACTIVE REGISTER
(2) 011612 000410      BR     VECMAP   ;GO FIND THE VECTOR NOW.
(2) 011614 104402 007603      5$:  TYPE   ,MERR2   ;NOTIFY OPR THAT NO DZV11'S FOUND.
(2) 011620 005000      CLR    R0      ;MAKE DATA DISPLAY ZERO
(2) 011622 000000      HALT
(2) 011624 000776      BR     -2      ;STOP THE SHOW
(2) 011626 012716 011514      6$:  MOV     #3$,(SP) ;DISABLE CONT. SW.
(2) 011632 000002      RTI    ;ENTERED BY NON-EXISTENT TIME-OUT
(2)                                ;RETURN TO MAINSTREAM
(2) 011634 012737 000200 000022 VECMAP: MOV     #MASK,@#22 ;SET IOT TRAP PRIORITY
(2) 011642 012737 011756 000020      MOV     #4$,@#20 ;SET IOT TRAP VECTOR
(2) 011650 012702 001500      MOV     #DZV.MAP,R2 ;SET SOFTWARE POINTER
(2) 011654 012700 000300      MOV     #300,R0   ;FLOATING VECTORS START HERE.
(2) 011660 012701 000302      MOV     #302,R1   ;PC OF IOT INSTR.
(2) 011664 010120      1$:  MOV     R1,(R0)+ ;START FILLING VECTOR AREA

```

```

(2) 011666 012721 000004      MOV      #4,(R1)+      ;WITH .+2; IOT
(2) 011672 022021             CMP      (R0)+,(R1)+  ;ADD 2 TO R0 +R1
(2) 011674 020127 001000      CMP      R1,#1000     ;HAS THE VECTOR AREA BEEN EXCEEDED?
(2) 011700 101771             BLOS    1$           ;BR IF MORE TO FILL
(2) 011702 013704 001410      MOV      SAVACTV,R4   ;STORE TEMPORARILY
(2) 011706 006004             2$:    ROR      R4     ;BRING OUT A BIT
(2) 011710 103036             BCC     5$           ;BR IF ALL DONE
(2) 011712 106427 000000      MTPS    #0           ;ZERO CPU PRIO
(2) 011716 012772 040040 000000 MOV      #BIT14+BIT5,@(R2) ;SET TIE AND MAS SCAN
(2) 011724 011201             MOV      (R2),R1     ;GET CSR
(2) 011726 112761 000017 000004 MOVVB   #17,4(R1)    ;SET THE TCR BITS FOR ALL LINES
(2)                               ;ATTEMPT TO FORCE AN INTERRUPT
(2)                               ;STALL
(2) 011734 005200             INC      R0           ;STALL
(2) 011736 001376             BNE     -2           ;FOR TIME TO INTERRUPT
(2) 011740 012762 000300 000002 MOV      #300,2(R2)   ;NO INTERRUPT ASSUME 300 AND FIX DZV11 LATER
(2) 011746 000005             RESET                    ;INIT
(2) 011750 062702 000012             3$:    ADD      #12,R2  ;POP SOFTWARE POINTER
(2) 011754 000754             BR      2$           ;KEEP GOING
(2) 011756 011662 000002             4$:    MOV      (SP),2(R2) ;GET VECTOR ADDRESS
(2) 011762 162762 000010 000002 SUB      #10,2(R2)    ;POINT BACK TO THE CORRECT VECTOR
(2) 011770 042762 000007 000002 BIC     #7,2(R2)     ;CLEAR JUNK
(2) 011776 022626             POP2SP                    ;POP IOT JUNK OFF STACK
(2) 012000 012716 011750             MOV      #3$,(SP)    ;SET FOR RETURN
(2) 012004 000002             RTI                               ;
(2) 012006 013737 001502 001170 5$:    MOV      DZVCO,$VECT1 ;COPY VECTOR OF FIRST DEVICE INTO ETABLE
(2) 012014 012737 004370 000020      MOV      #.SCOPE,IOTVEC ;RESTORE THE SCOPE TRAP
(2) 012022 000207             RTS      PC          ;ALL DONE WITH "AUTO SIZING"
(2)

```

8709

8710

(1)

(1)

(1)

(3)

(6)

(5)

012024 000004

(5)

(3)

012026 012737 000001 001246

(3)

012034 012737 012214 001362

(1)

012042 012737 012202 000004

(1)

012050 012737 000200 000006

(1)

012056 012737 012064 001364

(1)

012064 013700 002010

(1)

012070 011001

(1)

012072 000240

(1)

012074 005010

(1)

012076 000240

(1)

012100 012737 012106 001364

(1)

012106 013700 002014

(1)

012112 011001

(1)

012114 000240

(1)

012116 005010

(1)

012120 000240

(1)

012122 012737 012130 001364

(1)

012130 013700 002024

(1)

012134 011001

(1)

012136 000240

(1)

012140 005010

(1)

012142 000240

(1)

012144 012737 012152 001364

(1)

012152 013700 002030

(1)

012156 011001

(1)

012160 000240

(1)

012162 005010

(1)

012164 000240

(1)

012166 012737 000006 000004

(1)

012174 005037 000006

(1)

012200 104400

(1)

012202 011601

(1)

012204 022626

(1)

012206 104001

(1)

012210 104401

(1)

012212 000111

8711

8712

8713

8714

8716

(5)

(4)

012214 000004

(4)

(2)

012216 012737 000002 001246

(2)

012224 012737 012260 001362

8717

012232 013700 002010

```
***** TEST 1 *****
*THIS TEST PROVES THE BUS REPLY RESPONSE
*DURING A READ OR WRITE TO THE FOLLOWING ADDRESS:
*      DZVCSR, DZVRBUF, DZVTCR, DZVMSR
```

```
::* TEST 1
```

```
*****
```

```
TST1: SCOPE
```

```
MOV #1,$STNM ;LOAD THE NUMBER OF THIS TEST
MOV #TST2,NEXT ;POINT TO THE START OF THE NEXT TEST
MOV #5$,4 ;SET TRAP VECTOR
MOV #MASK,6 ;SET PRIORITY TO HIGH(MASK INTERRUPTS)
MOV #1$,LOCK ;SET RETURN IF SW09=11
1$: MOV DZVCSR,R0 ;SET ADDRESS TO TEST
MOV (R0),R1 ;READ THE ADDRESS
NOP ;WASTE TIME
CLR (R0) ;WRITE THE ADDRESS
NOP ;WASTE TIME
MOV #2$,LOCK ;SET RETURN ADDRESS FOR SW09
2$: MOV DZVRBUF,R0 ;SET ADDRESS TO TEST
MOV (R0),R1 ;READ THE ADDRESS
NOP ;WASTE TIME
CLR (R0) ;WRITE THE ADDRESS
NOP ;WASTE TIME
MOV #3$,LOCK ;SET RETURN ADDRESS FOR SW09
3$: MOV DZVTCR,R0 ;SET ADDRESS TO TEST
MOV (R0),R1 ;READ THE ADDRESS
NOP ;WASTE TIME
CLR (R0) ;WRITE THE ADDRESS
NOP ;WASTE TIME
MOV #4$,LOCK ;SET RETURN ADDRESS
4$: MOV DZVMSR,R0 ;SET ADDRESS TO TEST
MOV (R0),R1 ;READ FROM ADDRESS
NOP ;WASTE TIME
CLR (R0) ;WRITE THE ADDRESS
NOP ;WASTE TIME
MOV #6,4 ;SET TRAP CATCHER BACK TO NORMAL
CLR 6 ;
ADVANCE ;SCOPE THIS TEST
5$: MOV (SP),R1 ;SAVE PC OF TRAP
POP2SP ;POP TRAP OFF STACK
ERROR 1 ;*NO BUS REPLY RESPONSE.
SCOPE1 ;SW09=1?
JMP (R1) ;RTI
```

```
***** TEST 2 *****
*THIS TEST PROVES THAT BIT 'DCLR'
*CAN BE SET AND THAT IT WILL CLEAR
*BY ITSELF
```

```
::* TEST 2
```

```
*****
```

```
TST2: SCOPE
```

```
MOV #2,$STNM ;LOAD THE NUMBER OF THIS TEST
MOV #TST3,NEXT ;POINT TO THE START OF THE NEXT TEST
MOV DZVCSR,R0 ;SET POINTER
```

```

8718 012236 012710 000020      MOV    #DCLR,(R0)      ;SET DCLR
8719 012242 005005              CLR    R5              ;SET EXPECTED TO 0
8720 012244 005003              CLR    R3              ;DUAL LOOP COUNTER
8721 012246 011004      2$:  MOV    (R0),R4        ;IS DCLR CLEAR?
8722 012250 001403              BEQ    3$              ;IF YES, GO TO THE NEXT TEST
8723 012252 105203              INCB   R3              ;IF NO,COUNT 1 OF 256 TICKS
8724 012254 001374              BNE    2$              ;HAS THE TIME EXPIRED? IF NO, GO TEST BIT AGAIN
8725 012256 104002              ERROR  2              ;*DCLR FAILED TO CLEAR
8726 012260      3$:
8727      ;***** TEST 3 *****
      ;*TEST TO VERIFY THAT THE R/W BITS OF THE
      ;*DZVCSR REGISTER CAN BE SET. THEN VERIFY THAT
      ;*THESE BITS CAN BE CLEARED. AND FINALLY, VERIFY
      ;*THAT AFTER BEING SET AGAIN THEY CAN BE
      ;*CLEARED BY A 'DEVICE CLEAR'.
      ;*THE BITS TESTED ARE: MAINT, MSENAB, SILOEN,
      ;*RIE, AND TIE.
      ;:* TEST 3
      ;*****
TST3: SCOPE
(1) 012260 000004
(3) 012262 012737 000003 001246      MOV    #3,$STSTNM      ;LOAD THE NUMBER OF THIS TEST
(3) 012270 012737 012436 001362      MOV    #TST4,NEXT      ;POINT TO THE START OF THE NEXT TEST
(1) 012276 013700 002010              MOV    DZVCSR,R0       ;GET BASE ADDRESS
(1) 012302 012703 012416              MOV    #5$,R3          ;SET R3 TO TOP OF TABLE
(1) 012306 011305      1$:  MOV    (R3),R5          ;SET BIT
(1) 012310 012737 012316 001364      MOV    #11$,LOCK       ;SETUP FOR TIGHT SCOPE LOOP
(1) 012316 010510      11$: MOV    R5,(R0)          ;SET BIT IN DEVICE
(1) 012320 011004              MOV    (R0),R4         ;READ THE BIT FROM DEVICE
(1) 012322 020504              CMP    R5,R4           ;WAS BIT SET?
(1) 012324 001401              BEQ    2$              ;BR IF YES
(1) 012326 104002              ERROR  2              ;*BIT R/W FAILURE
(1) 012330 104401      2$:  SCOPE1              ;IS SWITCH 9 SET?
(1) 012332 012737 012340 001364      MOV    #12$,LOCK       ;SET FOR NEXT TIGHT SCOPE LOOP
(1) 012340 040510      12$: BIC    R5,(R0)          ;CLEAR THE BIT.
(1) 012342 011004              MOV    (R0),R4         ;READ DEVICE
(1) 012344 001403              BEQ    3$              ;BR IF BITS WERE CLEARED.
(1) 012346 005005              CLR    R5              ;CLEAR FOR ERROR PRINTOUT
(1) 012350 104002              ERROR  2              ;*BIT FAILED TO CLEAR
(1) 012352 011305      3$:  MOV    (R3),R5          ;RESTORE THE BIT.
(1) 012354 104401              SCOPE1              ;SW09 SET?
(1) 012356 012737 012364 001364      MOV    #13$,LOCK       ;SET UP FOR NEXT TIGHT SCOPE
(1) 012364 010510      13$: MOV    R5,(R0)          ;SET THE BIT AGAIN
(1) 012366 104413              DEVICE.CLR           ;ISSUE DEVICE CLEAR
(1) 012370 011004              MOV    (R0),R4         ;READ THE BIT.
(1) 012372 001403              BEQ    4$              ;BR IF BIT CLEARED BY INIT (DEVICE CLEAR)
(1) 012374 005005              CLR    R5              ;SET EXPECTED TO ZERO
(1) 012376 104002              ERROR  2              ;*BIT NOT CLEARED BY DEVICE CLEAR
(1) 012400 011305      4$:  MOV    (R3),R5          ;RESTORE BIT AGAIN
(1) 012402 104401              SCOPE1              ;SW09 SET?
(1) 012404 062703 000002      ADD    #2,R3           ;POP R3
(1) 012410 005713              TST    (R3)           ;IS THIS THE END OF TABLE?
(1) 012412 001407              BEQ    6$              ;IF YES GET OUT
(1) 012414 000734              BR     1$             ;OTHERWISE TEST NEXT BIT
(1) 012416 000010      5$:  #MAINT              ;CSR BIT: INTERNAL MAINTENANCE
(1) 012420 000040              #MSENAB              ;CSR BIT: MASTER SCAN ENABLE
  
```

```
(1) 012422 010000      #SILOEN      ;CSR BIT: SILO ENABLE
(1) 012424 000100      #RIE         ;CSR BIT: RECEIVER INTER. ENABLE
(1) 012426 040000      #TIE         ;CSR BIT: TRANS. INTER. ENABLE
(1) 012430 000000      #0           ;END OF TABLE
(1) 012432 005037 001364 6$: CLR      LOCK      ;ZERO LOCK INDICATOR
8728
(1)                      ;***** TEST 4 *****
(1)                      ;*THIS TESTS THAT ALL OF THE TCR BITS
(1)                      ;*CAN BE: SET, CLEARED, AND CLEARED BY A DEVICE CLEAR.
(1)                      ;*THIS TEST ALSO DETERMINES IF THE DTR BITS CAN
(1)                      ;*BE SET, CLEARED, AND CLEARED BY A RESET.
(3)                      ;:* TEST 4
(6)                      ;*****
(5) 012436 000004      TST4:  SCOPE
(5)
(3) 012440 012737 000004 001246  MOV      #4, $TSTNM      ;LOAD THE NUMBER OF THIS TEST
(3) 012446 012737 012642 001362  MOV      #TST5, NEXT    ;POINT TO THE START OF THE NEXT TEST
(1) 012454 013700 002024          MOV      DZVTCR, R0     ;SET DEVICE ADDRESS
(1) 012460 012703 012546          MOV      #5$, R3       ;SET R3 POINTER TO TOP OF TABLE
(1) 012464 012737 012474 001364 1$: MOV      #11$, LOCK     ;SET LOCK FOR SW09 SCOPE LOOP
(1) 012472 011305          MOV      (R3), R5      ;SET EXPECTED RESULTS
(1) 012474 010510 11$: MOV      R5, (R0)      ;SET THE BIT
(1) 012476 011004          MOV      (R0), R4      ;READ THE BIT FROM THE DEVICE
(1) 012500 020504          CMP      R5, R4        ;DID THE BIT SET?
(1) 012502 001401          BEQ      2$           ;BR IF YES
(1) 012504 104002          ERROR   2           ;*BIT FAILED TO SET.
(1) 012506 104401          SCOPE1          ;SW09 SET?
(1) 012510 012737 012516 001364 2$: MOV      #3$, LOCK     ;SET UP FOR NEXT TIGHT SCOPE LOOP
(1) 012516 040510 3$: BIC      R5, (R0)      ;CLEAR THE BIT
(1) 012520 011004          MOV      (R0), R4      ;READ THE REGISTER
(1) 012522 001403          BEQ      4$           ;BR IF YES
(1) 012524 005005          CLR      R5           ;SET EXPECTED TO 0
(1) 012526 104002          ERROR   2           ;*REPORT BIT NOT CLEAR
(1) 012530 011305          MOV      (R3), R5      ;RESTORE R5
(1) 012532 104401          SCOPE1          ;SW09 SET?
(1) 012534 062703 000002 4$: ADD      #2, R3        ;POP POINTER TO NEXT TABLE ENTRY
(1) 012540 005713          TST      (R3)         ;END OF TABLE?
(1) 012542 001412          BEQ      6$           ;IF YES JUMP OVER TABLE
(1) 012544 000747          BR       1$           ;START TESTING NEXT BIT
(1) 012546 000001 5$: #TCR0      ;TCR BIT FOR LINE 0
(1) 012550 000002          #TCR1      ;TCR BIT FOR LINE 1
(1) 012552 000004          #TCR2      ;TCR BIT FOR LINE 2
(1) 012554 000010          #TCR3      ;TCR BIT FOR LINE 3
(1) 012556 000400          #DTR0      ;DTR BIT FOR LINE 0
(1) 012560 001000          #DTR1      ;DTR BIT FOR LINE 1
(1) 012562 002000          #DTR2      ;DTR BIT FOR LINE 2
(1) 012564 004000          #DTR3      ;DTR BIT FOR LINE 3
(1) 012566 000000          #0         ;END OF TABLE
(1) 012570 005037 001364 6$: CLR      LOCK      ;CLEAR TIGHT SCOPE LOOP INDIC.
(1) 012574 012710 177777          MOV      #-1, (R0)     ;SET ALL BITS IN TCR REGISTER
(1) 012600 012705 007400          MOV      #007400, R5   ;SET EXPECTED
(1) 012604 104413          DEVICE.CLR ;SET DCLR BIT IN CSR
(1) 012606 011004          MOV      (R0), R4      ;READ REGISTER
(1) 012610 020504          CMP      R5, R4        ;TCR BITS CLEARED?
(1) 012612 001401          BEQ      7$           ;IF YES BRANCH
(1) 012614 104002          ERROR   2           ;TCR BITS NOT CLEARED!
(1) 012616 005005 7$: CLR      R5           ;SET EXPECTED TO ZERO
```


(1) 012620 005227 000000
(1) 012624 001375
(1) 012626 012710 177777
(1) 012632 000005
(1) 012634 011004
(1) 012636 001401
(1) 012640 104002
(1) 012642
8729
(1)
(1)
(1)
(1)
(1)
(3)
(6)
(5) 012642 000004
(5)
(3) 012644 012737 000005 001246
(3) 012652 012737 012744 001362
(1) 012660 013700 002010
(1) 012664 104413
(1) 012666 005005
(1) 012670 012710 121600
(1)
(1) 012674 011004
(1) 012676 001401
(1) 012700 104002
(1) 012702 012705 100040
(1) 012706 052777 000017 167110
(1) 012714 052710 000040
(1) 012720 005002
(1) 012722 011004
(1) 012724 042704 001400
(1) 012730 020504
(1) 012732 001404
(1) 012734 104414
(1) 012736 005202
(1) 012740 001370
(1) 012742 104002
(1) 012744
8730
8731
8732
8733
8734
8736
(5)
(4) 012744 000004
(4)
(2) 012746 012737 000006 001246
(2) 012754 012737 013074 001362
8737 012762 104413
8738 012764 013700 002010
8739 012770 012710 177757
8740 012774 012705 050150

```
8$: INC #0 ;DELAY FOR ACT
    BNE 8$ ;
    MOV #-1,(R0) ;SET ALL POSSIBLE BITS
    RESET ;DO BUS INIT
    MOV (R0),R4 ;DID REGISTER CLEAR?
    BEQ 9$ ;IF YES GET OUT
    ERROR 2 ;REGISTER DID NOT CLEAR!

9$: ;***** TEST 5 *****
    ;*THIS TEST VERIFIES THAT
    ;*BITS 'RDONE,TRDY, BIT9, BIT8,
    ;*AND SILOAL'' ARE READ ONLY AND THAT TRDY IS
    ;*ZERO UNTIL A LINE IS SELECTED AND MSENAB IS SET.
    ;*
    ;:* TEST 5
    ;*****
TST5: SCOPE
    MOV #5,$STSTNM ;LOAD THE NUMBER OF THIS TEST
    MOV #TST6,NEXT ;POINT TO THE START OF THE NEXT TEST
    MOV DZVCSR,R0 ;SET ADDRESS TO R0
    DEVICE.CLR ;DO A DEVICE CLEAR
    CLR R5 ;SET EXPECTED TO 0
    MOV #RDONE+TRDY+BIT9+BIT8+SILOAL,(R0) ;WRITE THE BITS
    MOV (R0),R4 ;READ BACK THE BITS
    BEQ 2$ ;BR IF NONE ARE SET.
    ERROR 2 ;*BITS WERE SET.
2$: MOV #TRDY+MSENAB,R5 ;SET EXPECTED BIT
    BIS #17,@DZVTCR ;SET TCR BITS FOR ALL LINES
    BIS #MSENAB,(R0) ;SET SCAN ENABLE
    CLR R2 ;SET COUNTER TO ZERO
3$: MOV (R0),R4 ;READ THE REGISTER
    BIC #BIT9!BIT8,R4 ;MASK OUT LINE NO.
    CMP R5,R4 ;BIT SET?
    BEQ 4$ ;BR IF YES
    DELAY ;STALL TIME
    INC R2 ;UPDATE COUNTER
    BNE 3$ ;BR IF COUNTER NOT DONE.
    ERROR 2 ;*TRDY NOT SET!

4$: ;***** TEST 6 *****
    ;*THIS TEST VERIFIES THAT:
    ;*TIE,SILOEN,RIE,MSENAB,AND MAINT ARE THE
    ;*ONLY R/W BITS IN THE DZVCSR AND THAT
    ;*SETTING 'DCLR'' IN THE CSR WILL CLEAR THESE BITS.
    ;:* TEST 6
    ;*****
TST6: SCOPE
    MOV #6,$STSTNM ;LOAD THE NUMBER OF THIS TEST
    MOV #TST7,NEXT ;POINT TO THE START OF THE NEXT TEST
    DEVICE.CLR ;SET DCLR IN CSR
    MOV DZVCSR,R0 ;SET UP FOR ERROR MESSAGE
    MOV #^C<DCLR>,(R0) ;TRY TO SET ALL BITS EXCEPT DCLR
    MOV #TIE!SILOEN!RIE!MSENAB!MAINT,R5 ;MAKE EXPECTED
```

```
8741 013000 011004      MOV      (R0),R4      ;ACTUAL
8742 013002 020405      CMP      R4,R5       ;CMP EXPECTED VS ACTUAL
8743 013004 001401      BEQ      1$          ;YES
8744 013006 104002      ERROR   2           ;*NO
8745 013010 105010      1$: CLRB   (R0)       ;CLEAR LOW BYTE OF CSR
8746 013012 105005      CLRB   R5           ;CLEAR LOW BYTE OF EXPECTED DATA
8747 013014 011004      MOV      (R0),R4     ;READ CSR
8748 013016 020405      CMP      R4,R5       ;DOES CSR COMPARE WITH EXPECTED?
8749 013020 001401      BEQ      3$          ;BRANCH IF YES
8750 013022 104002      ERROR   2           ;IF NOT PRINT ERROR
8751 013024 012710 177757 3$: MOV      #*C<DCLR>,(R0) ;SET ALL CSR BITS POSSIBLE
8752 013030 105077 166756 CLRB   @HDZVCSR     ;CLEAR HIGH BYTE OF CSR
8753 013034 012705 000150 MOV      #RIE!MSENAB!MAINT,R5 ;SET EXPECTED IN R5
8754 013040 011004      MOV      (R0),R4     ;READ CSR REGISTER
8755 013042 020405      CMP      R4,R5       ;DOES ACTUAL=EXPECTED
8756 013044 001401      BEQ      4$          ;IF YES CONTINUE
8757 013046 104002      ERROR   2           ;IF NO PRINT ERROR
8758 013050 012710 177757 4$: MOV      #*C<DCLR>,(R0) ;SET ALL POSSIBLE CSR BITS
8759 013054 025005      CLR      R5          ;SET R5 TO EXPECTED RESULTS
8760 013056 052710 000020 BIS     #DCLR,(R0)   ;DEVICE MASTER RESET
8761 013062 000240      NOP
8762 013064 011004      MOV      (R0),R4     ;ACTUAL
8763 013066 020405      CMP      R4,R5       ;CMP ACTUAL VS EXPECTED
8764 013070 001401      BEQ      2$          ;YES
8765 013072 104002      ERROR   2           ;*NO
8766 013074
8767
(1)
(1)
(1)
(3)
(6)
(5) 013074 000004
(5)
(3) 013076 012737 000007 001246 MOV      #7,$STSTM    ;LOAD THE NUMBER OF THIS TEST
(3) 013104 012737 013160 001362 MOV      #TST10,NEXT ;POINT TO THE START OF THE NEXT TEST
(1) 013112 104413 DEVICE.CLR          ;CLEAR DZV11
(1) 013114 013700 002014 MOV      DZVRBUF,R0 ;SET UP FOR ERROR MESSAGE
(1) 013120 011005 MOV      (R0),R5    ;COPY PRESENT CONTENTS
(1) 013122 042705 106000 BIC     #DVALID!BIT11!BIT10,R5 ;CLEAR ILLEGAL BITS
(1) 013126 012777 177777 166664 MOV      #-1,@DZVLPR ;TRY TO WRITE ALL 1'S
(1) 013134 011004 MOV      (R0),R4    ;ACTUAL
(1) 013136 020405 CMP      R4,R5      ;CMP ACTUAL VS EXPECTED
(1) 013140 001401 BEQ      1$         ;IF YES,GO CONTINUE PROCESSING
(1) 013142 104002 ERROR   2           ;*ERROR- BIT PATTERN NOT CORRECT
(1) 013144 005077 166650 1$: CLR     @DZVLPR ;TRY TO WRITE ALL ZEROES
(1) 013150 011004 MOV      (R0),R4    ;READ REGISTER
(1) 013152 020405 CMP      R4,R5      ;CMP ACTUAL VS. EXPECTED
(1) 013154 001401 BEQ      2$         ;BRANCH IF EQUAL
(1) 013156 104002 ERROR   2           ;VALUES DID NOT COMPARE
8768
(1)
(1)
(1)
(3)
::* TEST 7
*****
TST7: SCOPE
*****
::* TEST 10
*****
*THIS TEST PERFORMS RESET TESTING AND
*TESTING OF READ ONLY REGISTER DZVMSR
*AND TESTING OF WRITE ONLY REGISTER DZVTDR
::* TEST 10
```

```
(6)
(5) 013160 000004
(5)
(3) 013162 012737 000010 001246      MOV #10,$TSTNM      ;LOAD THE NUMBER OF THIS TEST
(3) 013170 012737 013244 001362      MOV #TST11,NEXT    ;POINT TO THE START OF THE NEXT TEST
(1) 013176 104413                      DEVICE.CLR          ;CLEAR DZV11
(1) 013200 013700 002030              MOV DZVMSR,R0      ;SET UP FOR ERROR MESSAGE
(1) 013204 011005                      MOV (R0),R5         ;COPY PRESENT CONTENTS
(1) 013206 042705 170360              BIC #170360,R5     ;CLEAR ILLEGAL BITS
(1) 013212 112777 177777 166614      MOVB #-1,@DZVTDR   ;TRY TO WRITE ALL 1'S
(1) 013220 011004                      MOV (R0),R4         ;ACTUAL
(1) 013222 020405                      CMP R4,R5           ;CMP ACTUAL VS EXPECTED
(1) 013224 001401                      BEQ 1$              ;IF YES,GO CONTINUE PROCESSING
(1) 013226 104002                      ERROR 2             ;*ERROR- BIT PATTERN NOT CORRECT
(1) 013230 005077 166600 1$:         CLR @DZVTDR ;TRY TO WRITE ALL ZEROES
(1) 013234 011004                      MOV (R0),R4         ;READ REGISTER
(1) 013236 020405                      CMP R4,R5           ;CMP ACTUAL VS. EXPECTED
(1) 013240 001401                      BEQ 2$              ;BRANCH IF EQUAL
(1) 013242 104002                      ERROR 2             ;VALUES DID NOT COMPARE
(1) 013244
```

8769
8770
8771
8772
8773
8774
8775
8776
8777
8778
8780
8781

```
***** TEST 11 *****
*VERIFY THAT SETTING 'DTR' FOR A LINE WILL
*BRING UP 'CO' AND 'RING' FOR:
*THE SAME LINE IF IN EXTERNAL MODE
*THE STAGGERED LINE IF IN STAGGERED MODE.
*LINES ARE STAGGERED AS FOLLOWS:
*LINE0 WITH LINE1; LINE2 WITH LINE3.
*THIS TEST IS ONLY RUN IF AN H325,OR H329
*IS CONNECTED ON THE DZV UNDER TEST.
```

::* TEST 11

```
(5)
(4) 013244 000004
(4)
(2) 013246 012737 000011 001246      MOV #11,$TSTNM      ;LOAD THE NUMBER OF THIS TEST
(2) 013254 012737 013440 001362      MOV #TST12,NEXT    ;POINT TO THE START OF THE NEXT TEST
8782 013262 005737 001372              TST MODE           ;TEST TO SEE IF TESTING WITH
8783 013266 001001                      BNE 8$             ;CONNECTOR
8784 013270 104400                      ADVANCE            ;IF NO, GO TO NEXT TEST
8785 013272 012737 013362 001364 8$:  MOV #10$,LOCK      ;SET FOR TIGHT SCOPE LOOP
8786 013300 104413                      DEVICE.CLR         ;SET DCLR IN CSR TO ZERO DEVICE
8787 013302 013700 002030              MOV DZVMSR,R0      ;SET REGISTER
8788 013306 005003                      CLR R3             ;ZERO LINE NUMBER
8789 013310 012702 000001              MOV #1,R2          ;SET POINTER
8790 013314 130237 001366 1$:        BITB R2,LINE        ;TEST THIS LINE?
8791 013320 001003                      BNE 3$             ;YES
8792 013322 005203                      INC R3             ;LINE #
8793 013324 104420                      SHIFT              ;GET NEXT LINE
8794 013326 000772                      BR 1$              ;TEST NEXT LINE
8795 013330 010204 001372 3$:        MOV R2,R4           ;SAVE BINARY BIT FOR LINE #
8796 013332 105737                      TSTB MODE          ;RUNNING IN EXTERNAL MODE?
8797 013336 100406                      BMI 5$             ;IF YES SKIP STAGGERED SETUP
8798 013340 032703 000001              BIT #BIT0,R3       ;IF EVEN LINE
8799 013344 001402                      BEQ 4$             ;GO GET ODD PARTNER
8800 013346 006204                      ASR R4             ;OTHERWISE GET EVEN COMPANION
```

```
8801 013350 000401 BR 5$ ;GO SETUP EXPECTED RESULTS
8802 013352 006304 4$: ASL R4 ;FIND ODD PARTNER
8803 013354 010405 5$: MOV R4,R5 ;LOAD R5 FOR EXPECTED
8804 013356 000305 SWAB R5 ;PLACE IN UPPER BYTE
8805 013360 150405 BISB R4,R5 ;SET FOR RING BITS
8806 013362 150277 166440 10$: BISB R2,@HDZVTCR ;SET DTR BIT
8807 013366 104414 DELAY ;DELAY FOR CABLE LAG
8808 013370 011004 MOV (R0),R4 ;MOVE RESULTS OF MSR REGISTER TO R4
8809 013372 020504 CMP R5,R4 ;RESULTS=EXPECTED?
8810 013374 001401 BEQ 6$ ;IF YES CONTINUE
8811 013376 104002 ERROR 2 ;IF NOT PRINT ERROR RESULTS
8812 013400 104401 6$: SCOPI ;IS SW09 SET?
8813 013402 012737 013410 001364 MOV #11$,LOCK ;SET UP FOR NEXT TIGHT SCOPE
8814 013410 140277 166412 11$: BICB R2,@HDZVTCR ;CLEAR DTR BIT FOR LINE UNDER TEST
8815 013414 104414 DELAY ;DELAY FOR CABLE LAG
8816 013416 011004 MOV (R0),R4 ;LOAD MSR REGISTER INTO R4
8817 013420 001402 BEQ 7$ ;IF CO AND RING CLEARED CONTINUE
8818 013422 005005 CLR R5 ;OTHERWISE SET EXPECTED FOR ERROR
8819 013424 104002 ERROR 2 ;PRINTOUT
8820 013426 104401 7$: SCOPI ;IS SW09 SET?
8821 013430 012737 013362 001364 MOV #10$,LOCK ;RESET TIGHT SCOPE LOOP
8822 013436 000731 BR 2$ ;GET NEXT LINE
8823
8824 ;***** TEST 12 *****
(1) ;* THIS TEST VERIFIES THAT TRDY IS SET WHEN A LINE
(1) ;* IS READY TO BE LOADED, AND THAT THE LINE SPECI-
(1) ;* FIED IN BITS 8-9 OF DZVCSR CORRESPOND
(1) ;* TO THE LINE SELECTED IN DZVTCR
(3) ;:* TEST 12
(6) ;*****
(5) 013440 000004 TST12: SCOPE
(5)
(3) 013442 012737 000012 001246 MOV #12,$TSTNM ;LOAD THE NUMBER OF THIS TEST
(3) 013450 012737 013572 001362 MOV #TST13,NEXT ;POINT TO THE START OF THE NEXT TEST
(1) 013456 104413 DEVICE.CLR ;ISSUE A 'DEVICE CLEAR' (RESET)
(1) 013460 012737 013514 001364 MOV #2$,LOCK ;SET UP FOR TIGHT SCOPE LOOP
(1) 013466 005037 011374 CLR SAVLIN ;INITIALIZE FOR ERROR PRINTOUT
(1) 013472 013700 002010 MOV DZVCSR,R0 ;SET POINTER
(1) 013476 012705 100040 MOV #MSENAB!TRDY,R5 ;START THE EXPECTED LINE NUMBER AT 0
(1) 013502 012702 000001 MOV #1,R2 ;USING R2 AS A BIT POINTER, POINT TO LINE 0
(1) 013506 130237 001366 1$: BITB R2,LINE ;IS THIS LINE SELECTED?
(1) 013512 001421 BEQ 6$ ;IF NO, SKIP THE STARTUP
(1) 013514 050277 166304 2$: BIS R2,@DZVTCR ;SET THE GO BIT FOR THIS LINE
(1) 013520 052710 000040 BIS #MSENAB,(R0) ;START THE SCANNER
(1) 013524 005004 CLR R4 ;SET FOR DELAY
(1) 013526 005710 3$: TST (R0) ;TX READY?
(1) 013530 100404 BMI 4$ ;BR IF YES
(1) 013532 104414 DELAY ;DELAY
(1) 013534 005204 INC R4 ;COUNTER
(1) 013536 001373 BNE 3$ ;BR IF <>0!
(1) 013540 104003 ERROR 3 ;*TX NOT READY!
(1) 013542 011004 4$: MOV (R0),R4 ;GET THE LINE POINTED TO BY THE SCANNER
(1) 013544 020405 CMP R4,R5 ;IS THE LINE NUMBER WHAT IT SHOULD BE?
(1) 013546 001401 BEQ 5$ ;IF YES,GO WORK ON THE NEXT LINE
(1) 013550 104002 ERROR 2 ;*LINE NUMBER DID NOT MATCH TCR BIT
(1) 013552 104401 5$: SCOPI ;IS SW09 SET?
```

```
(1) 013554 104413          DEVICE.CLR          ;SET DCLR IN CSR;SETUP FOR NEXT LINE
(1) 013556 062705 000400 6$: ADD #400,R5      ;POINT TO THE NEXT EXPECTED LINE
(1) 013562 104420          SHIFT              ;POINT TO THE NEXT LINE.ARE ALL LINES TESTED?
(1) 013564 005237 001374  INC SAVLIN         ;ADJUST FOR ERROR PRINTOUT
(1) 013570 000746          BR 1$             ;IF NOT, GO DO THE NEXT LINE
8825          ;***** TEST 13 *****
8826          ;*TEST TO TRANSMIT ONE CHAR AND
8827          ;*RECEIVE ONE CHAR ON ONE LINE
8828          ;*AT A TIME. THE CHAR IS "252" AND
8829          ;*ALL SELECTED LINES WILL BE TURNED ON .
8830          ;*THIS IS THE FIRST TIME ANY
8831          ;*DATA IS CHECKED IN THE RECEIVER.
8832          ;*USING SWITCH NINE WITH THIS TEST CREATES A TIGHT SCOPE LOOP
8833          ;*WHICH TRANSMITS A STEADY STREAM OF CHARACTERS.
8835          ;:* TEST 13
          ;*****
          TST13: SCOPE
(5)
(4) 013572 000004
(4)
(2) 013574 012737 000013 001246  MOV #13,$STSTM      ;LOAD THE NUMBER OF THIS TEST
(2) 013602 012737 014062 001362  MOV #TST14,NEXT    ;POINT TO THE START OF THE NEXT TEST
(1) 013610 012737 014044 001364  MOV #16$,LOCK      ;USE THIS ADDRESS IF A TIGHT SCOPE LOOP IS SELECTED
8836 013616 104417          DCLASM            ;SET DCLR IN CSR AND SET MAINT MODE
8837 013620 104421          LPRSET           ;LOAD LPR REGISTER FOR ALL LINES
8838 013622 005037 001374  CLR SAVLIN         ;INIT. FOR ERROR PRINTOUT
8839 013626 105037 001425  CLRB DONFLG       ;INIT FOR TCR BIT HANDLER
8840 013632 012702 000001  MOV #1,R2          ;LINE POINTER
8841 013636 012701 000252  MOV #252,R1        ;SAVE CHARACTER TO BE TRANSMITTED
8842 013642 052777 000040 166140  BIS #MSENAB,@DZVCSR ;START SCANNER
8843 013650 030237 001366 3$: BIT R2,LINE     ;VALID LINE ?
8844 013654 001467          BEQ 15$           ;NO SET UP NEXT LINE
8845 013656 010277 166142  MOV R2,@DZVTDR    ;SET TCR BIT
8846 013662 005005 5$: CLR R5              ;SET R5 FOR A DELAY LOOP
8847 013664 105777 166120  TSTB @DZVCSR      ;IS REC DONE = 0 ?
8848 013670 100001          BPL 6$            ;IF YES, ALLOW TIME FOR TRDY TO SET
8849 013672 104020          ERROR 20          ;*REC DONE SHOULD = 0
8850 013674 005777 166110 6$: TST @DZVCSR     ;TRDY SET?
8851 013700 100404          BMI 7$           ;IF YES BRANCH
8852 013702 104414          DELAY            ;IF NO THEN WAIT FOR IT
8853 013704 005205          INC R5              ;DELAY LOOP
8854 013706 001372          BNE 6$            ;BRANCH BACK AND TEST AGAIN
8855 013710 104003          ERROR 3          ;*TRDY FAILED TO SET!
8856 013712 105737 001425 7$: TSTB DONFLG     ;HAVE WE ALREADY SENT CHARAC.
8857 013716 001041          BNE 13$          ;IF YES GO CLEAR TCR BIT
8858 013720 105237 001425  INCB DONFLG       ;IF NOT INDICATE HAVING BEEN HERE
8859 013724 110177 166104  MOVB R1,@DZVTDR   ;LOAD CHARACTER
8860 013730 013705 001374  MOV SAVLIN,R5     ;MAKE EXPECTED LINE #
8861 013734 005737 001372  TST MODE          ;IS THIS TEST IN STAGGERED MODE?
(1) 013740 100006          BPL 10$          ;IF NOT, SKIP STAGGERED SETUP
(1)
(1)
(1)
(1) 013742 006205          ASR R5            ;GET THE LAST BIT INTO THE CARRY BIT
(1) 013744 103402          BCS 8$           ;IF IT IS SET, GO CLEAR IT
(1) 013746 000261          SEC            ;IF IT IS CLEAR SET IT HERE
(1) 013750 000401          BR 9$           ;SKIP THE CLEARING
(1) 013752 000241 8$: CLC            ;CLEAR THE CARRY BIT (INVERSION OF LINE PARITY)
```

```
(1) 013754 006105          9$:  ROL      R5          ;GET THE NEW BIT BACK INTO R5
8862 013756 000305        10$:  SWAB     R5          ;MOVE THE LINE NUMBER TO THE UPPER BYTE
8863 013760 150105          BISB   R1,R5          ;ADD CHARACTER
8864 013762 052705 100000  BIS    #DVALID,R5    ;ADD DATA VALID
8865 013766 005003          CLR    R3
8866 013770 105777 166014  11$:  TSTB    @DZVCSR     ;IS RDONE SET?
8867 013774 100404          BMI    12$          ;IF YES GO GET CHAR.
8868 013776 104414          DELAY          ;IF NOT THEN WAIT
8869 014000 005203          INC    R3          ;DELAY LOOP
8870 014002 001372          BNE    11$          ;DELAY DONE?
8871 014004 104004          ERROR  4          ;*RDONE FAILED TO SET!
8872 014006 017704 166002  12$:  MOV     @DZVRBUF,R4  ;LOAD THE VALUE ACTUALLY RECEIVED
8873 014012 020405          CMP    R4,R5       ;COMPARE ACTUAL VS EXPECTED. ARE THEY THE SAME?
8874 014014 001722          BEQ    5$          ;IF YES, GO DO THE NEXT LINE
8875 014016 104006          ERROR  6          ;*NO DATA/CONTENTS DID NOT COMPARE
8876 014020 000720          BR    5$          ;GO BACK AND WAIT TO CLEAR TCR BIT
8877 014022 104401          13$:  SCOP1          ;CHECK TO SEE IF SWITCH NINE IS SET
8878 014024 105037 001425  CLRB   DONFLG       ;SET UP FOR NEXT LINE
8879 014030 005077 165770  CLR    @DZVTCCR     ;CLEAR PREVIOUS TCR BIT
8880 014034 005237 001374  15$:  INC     SAVLIN      ;SET LINE INDICATOR FOR NEXT LINE
8881 014040 104420          SHIFT
8882 014042 000702          BR    3$          ;CALCULATE NEXT LINE
8883                                     ;GET GET STARTED
8884                                     ;TIGHT SCOPE LOOP FOR THIS TEST. LOOP TRANSMITS CHARACTERS ONLY
8885
8886 014044 005777 165740  16$:  TST     @DZVCSR     ;IS TRANSMITTER READY?
8887 014050 100375          BPL    16$         ;IF NOT, WAIT FOR IT
8888 014052 110177 165756  MOVB   R1,@DZVTDR   ;LOAD THE CHARACTER
8889 014056 104401          SCOP1          ;LOOP AGIN IF SW09=1
8890 014060 000760          BR    13$        ;OTHERWISE, GO PICK UP THE TEST NORMALLY
8891
8892                                     ;***** TEST 14 *****
8893                                     ;*THIS TEST VERIFIES THAT EACH RECEIVING LINE CAN BE
8894                                     ;*DISABLED BY SETTING RCVON (BIT12 IN THE LPR REGISTER)
8895                                     ;*TO ZERO FOR EACH LINE.
8896                                     ;*THIS TEST ALSO VERIFIES THAT THE SILO CAN BE
8897                                     ;*EMPTIED BY ISSUING A DEVICE MASTER CLEAR.
8898
8899                                     ;:* TEST 14
9000                                     ;*****
(4) 014062 000004        TST14: SCOPE
(2) 014064 012737 000014 001246  MOV    #14,$TSTNM   ;LOAD THE NUMBER OF THIS TEST
(2) 014072 012737 014404 001362  MOV    #TST15,NEXT  ;POINT TO THE START OF THE NEXT TEST
8900 014100 105037 001425          CLRB   DONFLG       ;CLEAR TEST CONTROL FLAG
8901 014104 005037 001374          CLR    SAVLIN       ;CLEAR LINE INDICATOR
8902 014110 104417          DCLASM          ;ISSUE A DEVICE MASTER CLEAR
8903                                     ;AND SET MAINT BIT IF NECESSARY
8904 014112 013701 001370          MOV    PAR,R1       ;SAVE DEFAULT PARAMETERS
8905 014116 042737 010000 001370  BIC    #RCVON,PAR   ;DISABLE RECEIVER IN DEFAULT PAR.
8906 014124 104421          LPRSET          ;LOAD PARAMETERS IN LPR REGISTER
8907 014126 010137 001370          MOV    R1,PAR       ;RESTORE DEFAULT PARAMETERS
8908 014132 012701 000252          MOV    #252,R1      ;LOAD A CHARAC. INTO R1
8909 014136 013702 001366          MOV    LINE,R2      ;COPY AN IMAGE OF THE ACTIVE LINES
8910 014142 010277 165656          MOV    R2,@DZVTCCR  ;SET TCR BITS FOR ALL ACTIVE LINES
8911 014146 052777 000040 165634  BIS    #MSENAB,@DZVCSR ;SET MASTER SCAN ENABLE
8912 014154 005005          CLR    R5          ;INIT DELAY COUNTER
```

8913	014156	005777	165626	2\$:	TST @DZVCSR	: IS TRANS READY SET?
8914	014162	100404			BMI 3\$: BRANCH IF YES
8915	014164	104414			DELAY	: WAIT FOR TRDY TO SET
8916	014166	005205			INC R5	: INCREMENT DELAY COUNTER
8917	014170	001372			BNE 2\$: RETURN TO CHECK TRDY
8918	014172	104003			ERROR 3	: TRDY FAILED TO SET!
8919	014174	117705	165612	3\$:	MOVB @HDZVCSR,R5	: MOVE LINE NO. TO R5
8920	014200	012703	000001		MOV #1,R3	: INIT TCR POINTER
8921	014204	042705	177774		BIC #^C<3>,R5	: ISOLATE LINE NO.
8922	014210	001403			BEQ 31\$: IF LINE 0 BRANCH
8923	014212	106303		30\$:	ASLB R3	: SHIFT R3 POINTER TO NEXT LINE
8924	014214	005305			DEC R5	: DECREMENT LINE NO.
8925	014216	001375			BNE 30\$: WHEN R5=0, R3 POINTS TO LINE TCR
8926	014220	030302		31\$:	BIT R3,R2	: HAS CHARACTER BEEN SENT?
8927	014222	001007			BNE 4\$: BRANCH IF NO
8928	014224	140377	165574		BICB R3,@DZVTDR	: IF YES THEN CLEAR TCR BIT
8929	014230	001351			BNE 1\$: IF ALL CHARAC. SENT DROP THROUGH
8930	014232	105737	001425		TSTB DONFLG	: IF NO MORE ACTIVE IS THIS SECOND
8931						: TIME HERE?
8932	014236	001037			BNE 10\$: IF YES SKIP TO SECOND PART OF TEST
8933	014240	000404			BR 5\$: IF FIRST TIME HERE GO ZERO TCR BITS
8934	014242	110177	165566	4\$:	MOVB R1,@DZVTDR	: LOAD CHAR. INTO BUFFER
8935	014246	040302			BIC R3,R2	: INDICATE CHARAC. SENT ON THIS LINE
8936	014250	000741			BR 1\$: GO BACK AND WAIT FOR TRDY TO SET
8937	014252	005077	165546	5\$:	CLR @DZVTDR	: CLEAR OUT TCR BITS
8938	014256	005005			CLR R5	: INIT DELAY COUNTER
8939	014260	105777	165524	6\$:	TSTB @DZVCSR	: IS RECEIV. DONE SET?
8940	014264	100002			BPL 7\$: IF NOT THEN WAIT TO SEE IF IT WILL
8941	014266	104020			ERROR 20	: REC DONE SHOULD NOT SET!
8942	014270	000403			BR 8\$: GO FIND WHICH LINE RECEIVED
8943	014272	104414		7\$:	DELAY	: STALL FOR RECEIVER
8944	014274	005205			INC R5	: INCREMENT DELAY COUNTER
8945	014276	001370			BNE 6\$: IF NOT DONE GO RETEST REC DONE
8946	014300	017704	165510	8\$:	MOV @DZVRBUF,R4	: READ REC. BUFFER
8947	014304	100007			BPL 9\$: IS DVALID SET?
8948	014306	000304			SWAB R4	: IF YES GET LINE NO.
8949	014310	042704	177774		BIC #^C<3>,R4	: ISOLATE LINE NO.
8950	014314	010437	001374		MOV R4,SAVLIN	: SET UP LINE NO. FOR ERROR REPORT
8951	014320	104017			ERROR 17	: DVALID SHOULD NOT BE SET
8952	014322	000766			BR 8\$: GO CHECK FOR ANY OTHER CHAR. IN SILO
8953	014324	105237	001425	9\$:	INCB DONFLG	: INDICATE THAT FIRST PART OF TEST IS DONE
8954	014330	013701	001370		MOV PAR,R1	: SAVE DEFAULT LINE PARAM.
8955	014334	000673			BR 100\$: NOW GO RELOAD LPR REGISTER TO
8956						: TURN RECEIVERS ON
8957	014336	005005		10\$:	CLR R5	: ZERO DELAY COUNTER
8958	014340	104414		11\$:	DELAY	: WAIT FOR ALL CHARAC. TO BE RECEIVED
8959	014342	005205			INC R5	: INCREASE DELAY COUNT
8960	014344	001375			BNE 11\$: CONT. DELAY IF NOT FINISHED
8961	014346	104413			DEVICE.CLR	: ISSUE A MASTER CLEAR
8962	014350	000240			NOP	
8963	014352	000240			NOP	
8964	014354	105777	165430		TSTB @DZVCSR	: NOW IS RECEIV. DONE SET?
8965	014360	100003			BPL 12\$: BRANCH IF NO
8966	014362	005037	001374		CLR SAVLIN	: CLEAR LINE NO FOR ERROR REPORT
8967	014366	104020			ERROR 20	: REC. DONE SHOULD NOT BE SET!
8968	014370	017704	165420	12\$:	MOV @DZVRBUF,R4	: READ REC. BUFFER

3

CNDZA-A MACY11 30(1046) 15-DEC-82 14:36
CNDZAA.P11 15-DEC-82 14:31

PAGE 81-57
DZV11 DEVICE DIAGNOSTICS.

L 6

COPYRIGHT 1982 DIGITAL EQUIP. CORP.

SEQ 0076

```

8969 014374 100003          BPL      13$          ;IS DVALID SET? IT SHOULDN'T BE
8970 014376 005037 001374  CLR      SAVLIN      ;DEVICE. CLR DID NOT ZERO SILO
8971 014402 104017          ERROR    17          ;PRINT OUT THE ERROR.(LINE NO. IS IRRELEVANT)
8972 014404          13$:
8973
8974          ;***** TEST 15 *****
8975          ;* THIS TEST PROVES THAT THE TRANSMITTER TRANSMITS
8976          ;*CHARACTERS (FLAG MODE)AND THE RECEIVER RECEIVES (FLAG MODE)
8977          ;*(ONE LINE AT A TIME  BASED UPON VALID LINES)
8978          ;*THIS IS THE FIRST TIME THAT ALL DATA IS CHECKED
8980          ;:* TEST 15
(5)          ;*****
(4) 014404 000004          TST15: SCOPE
(4)
(2) 014406 012737 000015 001246  MOV      #15,$STNM    ;LOAD THE NUMBER OF THIS TEST
(2) 014414 012737 014674 001362  MOV      #TST16,NEXT  ;POINT TO THE START OF THE NEXT TEST
(1) 014422 012737 014510 001364  MOV      #5$,LOCK     ;USE THIS ADDRESS IF A TIGHT SCOPE LOOP IS SELECTED
8981 014430 104417          DCLASM          ;SET DCLR AND SET MNTFLG
8982 014432 104421          LPRSET         ;LOAD LPR REGISTER FOR ALL LINES
8983 014434 005037 001374  CLR      SAVLIN      ;INIT FOR FIRST LINE
8984 014440 104422          BUFSET         ;ZERO BUFFER AREA
8985 014442 105037 001425  CLRB     DONFLG      ;ZERO TCR BIT HANDLER FLAG
8986 014446 012702 000001  MOV      #1,R2       ;LINE POINTER
8987 014452 052777 000040 165330  BIS      #MSENAB,@DZVCSR ;START SCANNER
8988 014460 030237 001366          BIT      R2,LINE     ;VALID LINE ?
8989 014464 001477          BEQ      15$        ;NO SET UP NEXT LINE
8990 014466 010277 165332  MOV      R2,@DZVTCR  ;SET TCR BIT
8991 014472 013700 001374  MOV      SAVLIN,R0   ;ADJUST BUFFER POINTER
8992 014476 006300          ASL      R0         ;OFFSET
8993 014500 105777 165304          4$: TSTB     @DZVCSR    ;IS REC DONE = 0 ?
8994 014504 100001          BPL      5$        ;IF YES, ALLOW TIME FOR TRDY TO SET
8995 014506 104020          ERROR    20        ;*REC DONE SHOULD = 0
8996 014510 005005          5$: CLR      R5        ;USE R5 AS TIMER WAITING FOR TRDY TO SET
8997 014512 005777 165272          6$: TST      @DZVCSR  ;IS THE TRANSMITTER READY?
8998 014516 100404          BMI      7$        ;IF SO, GO TRANSMIT A CHARACTER
8999 014520 104414          DELAY          ;WAIT A LITTLE BIT
9000 014522 005205          INC      R5        ;UP THE LOCAL COUNTER.TIME EXCEEDED?
9001 014524 001372          BNE     6$        ;IF NOT, GO TRY AGAIN
9002 014526 104003          ERROR    3         ;*TRDY FAILED TO SET!
9003 014530 105737 001425          7$: TSTB     DONFLG   ;ALL CHARAC. TRANS.?
9004 014534 001047          BNE     14$       ;IF YES GO ZERO TCR BIT
9005 014536 116077 001426 165270  MOVB    TD0(R0),@DZVTDR ;LOAD CHARACTER
9006 014544 013705 001374  MOV      SAVLIN,R5   ;MAKE EXPECTED LINE #
9007 014550 005737 001372  TST      MODE       ;IS THIS TEST IN STAGGERED MODE?
(1) 014554 100006          BPL      10$       ;IF NOT, SKIP STAGGERED SETUP
(1)
(1)
(1)          ;WE MUST NOW INVERT THE LAST BIT OF THE LINE NUMBER
(1) 014556 006205          ASR      R5        ;GET THE LAST BIT INTO THE CARRY BIT
(1) 014560 103402          BCS     8$        ;IF IT IS SET, GO CLEAR IT
(1) 014562 000261          SEC          ;IF IT IS CLEAR SET IT HERE
(1) 014564 000401          BR      9$        ;SKIP THE CLEARING
(1) 014566 000241          8$: CLC          ;CLEAR THE CARRY BIT (INVERSION OF LINE PARITY)
(1) 014570 006105          9$: ROL      R5        ;GET THE NEW BIT BACK INTO R5
9008 014572 000305          10$: SWAB     R5      ;MOVE THE LINE NUMBER TO THE UPPER BYTE
9009 014574 156005 001426  BISB    TD0(R0),R5   ;ADD CHARACTER

```



```

9010 014600 052705 100000      BIS      #DVALID,R5      ;ADD DATA VALID
9011 014604 005003              CLR      R3              ;
9012 014606 105777 165176    11$:     TSTB     @DZVCSR      ;REC DONE?
9013 014612 100404              BMI      12$            ;IF YES GO CHECK CHAR.
9014 014614 104414              DELAY   ;IF NOT WAIT FOR REC.
9015 014616 005203              INC      R3              ;DELAY LOOP TIMER
9016 014620 001372              BNE     11$             ;DELAY FINISHED?
9017 014622 104004              ERROR   4               ;*RDONE FAILED TO SET!
9018 014624 017704 165164    12$:     MOV      @DZVRBUF,R4    ;LOAD THE VALUE ACTUALLY RECEIVED
9019 014630 020405              CMP     R4,R5           ;COMPARE ACTUAL VS EXPECTED. ARE THEY THE SAME?
9020 014632 001401              BEQ     13$            ;IF YES, GO DO THE NEXT LINE
9021 014634 104006              ERROR   6               ;*NO DATA/CONTENTS DID NOT COMPARE
9022 014636 104401              SCOPI  ;CHECK TO SEE IF SWITCH NINE IS SET
9023 014640 105260 001426    13$:     INCB    TDO(RO)      ;INCREMENT BINARY PATTERN FOR THIS LINE
9024 014644 001315              BNE     4$              ;GO 'ROUND AGAIN FOR NEXT CHARACTER
9025 014646 105237 001425    14$:     INCB    DONFLG     ;INDICATE ALL CHAR. SENT
9026 014652 000712              BR      4$              ;BRANCH TO CLEAR TCR BIT
9027 014654 005077 165144    14$:     CLR      @DZVTCR     ;CLEAR TCR REGISTER
9028 014660 105037 001425    15$:     CLRB   DONFLG     ;INIT FOR NEXT LINE
9029 014664 005237 001374    15$:     INC     SAVLIN     ;INC EXPECTED LINE
9030 014670 104420              SHIFT  ;SHIFT THE LINE POINTER. ARE WE ALL DONE?
9031 014672 000672              BR      3$              ;IF NO, GO AROUND AGAIN FOR NEXT LINE

```

```

9032
9033
9034      ;***** TEST 16 *****
9035      ;*THIS TEST WILL PROVE THAT:
9036      ;* 1) THE TRANSMITTER 'BREAK BIT' WORKS
9037      ;* 2) THE RECEIVER CAN FLAG 'FRAMING ERRORS'
9038      ;* 3) THE RECEIVER CAN FLAG 'PARITY ERRORS'
9039      ;*ONLY ONE LINE AT A TIME WILL BE EXERCISED.

```

```

9041      ;:* TEST 16
9042      ;*****
(5)
(4) 014674 000004      TST16: SCOPE
(4)
(2) 014676 012737 000016 001246      MOV      #16,$STNM      ;LOAD THE NUMBER OF THIS TEST
(2) 014704 012737 015076 001362      MOV      #TST17,NEXT   ;POINT TO THE START OF THE NEXT TEST
9042 014712 012737 015022 001364      MOV      #5$,LOCK      ;SET FOR LOOP
9043 014720 005037 001374              CLR      SAVLIN        ;INIT LINE INDIC. FOR ERROR PRINTOUT
9044 014724 012702 000001              MOV     #1,R2          ;LINE POINTER
9045 014730 030237 001366    1$:     BIT      R2,LINE      ;VALID LINE?
9046 014734 001454              BEQ     9$              ;IF NOT SET FOR NEXT LINE
9047 014736 104417              DCLASM ;SET DCLR IN CSR AND SET MNTFLG
9048 014740 013701 001370              MOV     PAR,R1         ;PICK UP PARAMETERS
9049 014744 052737 000300 001370      BIS     #ODDPAR!PARITY,PAR ;FORCE ODD PARITY
9050 014752 104421              LPRSET ;LOAD LPR REGISTER
9051 014754 010137 001370              MOV     R1,PAR         ;RESET PAR TO ORIGINAL VALUE
9052 014760 052777 000040 165022      BIS     #MSENAB,@DZVCSR ;START SCANNER
9053 014766 013705 001374              MOV     SAVLIN,R5     ;MAKE EXPECTED DATA
9054 014772 005737 001372              TST    MODE           ;IS THIS TEST IN STAGGERED MODE?
(1) 014776 100006              BPL     4$             ;IF NOT, SKIP STAGGERED SETUP
(1)
(1)
(1)
(1) 015000 006205              ASR     R5              ;GET THE LAST BIT INTO THE CARRY BIT
(1) 015002 103402              BCS    2$              ;IF IT IS SET, GO CLEAR IT
(1) 015004 000261              SEC     ;IF IT IS CLEAR SET IT HERE

```

```

(1) 015006 000401 BR 3$ ;SKIP THE CLEARING
(1) 015010 000241 2$: CLC ;CLEAR THE CARRY BIT (INVERSION OF LINE PARITY)
(1) 015012 006105 3$: ROL R5 ;GET THE NEW BIT BACK INTO R5
9055 015014 000305 4$: SWAB R5 ;PUT LINE NUMBER IN UPPER BYTE
9056 015016 052705 130000 BIS #DVALID!PARER!FRMERR,R5 ;ADD EXPECTED
9057 015022 005003 5$: CLR R3 ;INIT DELAY ACCUMULATOR
9058 015024 110277 165006 MOVB R2,@HDZVTDR ;SET BREAK BIT
9059 015030 105777 164754 6$: TSTB @DZVCSR ;RECEIVER DONE?
9060 015034 100404 BMI 7$ ;BRANCH IF YES
9061 015036 104414 DELAY ;WAIT FOR REC DONE TO SET
9062 015040 005203 INC R3 ;INC DELAY LOOP
9063 015042 001372 BNE 6$ ;DELAY FINISHED?
9064 015044 104004 ERROR 4 ;*R DONE FAILED TO SET!
9065 015046 017704 164742 7$: MOV @DZVRBUF,R4 ;ACTUAL
9066 015052 020405 CMP R4,R5 ;CMP ACTUAL VS EXPECTED. DO THEY MATCH?
9067 015054 001401 BEQ 8$ ;IF YES, GO CLEAN UP
9068 015056 104006 ERROR 6 ;*DATA/CONTENTS FAILED TO COMPARE
9069 015060 105077 164752 8$: CLRB @HDZVTDR ;CLEAR BREAK BITS
9070 015064 104401 SCOP1 ;LOOP?
9071 015066 005237 001374 9$: INC SAVLIN ;INC LINE #
9072 015072 104420 SHIFT ;SET R2 TO NEXT LINE
9073 015074 000715 BR 1$ ;GO BACK AND TEST NEXT LINE
9074 ***** TEST 17 *****
(1) ;* THIS TEST VERIFIES THAT THE DEVICE DOES NOT INTERRUPT
(1) ;*WHILE THE PROCESSOR STATUS DOES NOT ALLOW INTERRUPTS
(1) ;*BUT WILL INTERRUPT IF THE PROCESSOR STATUS
(1) ;*ALLOWS INTERRUPTS.
(3) ;: * TEST 17
(6) ;: *****
(5) 015076 000004 TST17: SCOPE
(5)
(3) 015100 012737 000017 001246 MOV #17,$STNM ;LOAD THE NUMBER OF THIS TEST
(3) 015106 012737 015472 001362 MOV #TST20,NEXT ;POINT TO THE START OF THE NEXT TEST
(1) 015114 104417 DCLASM ;SET DCLR IN CSR AND SET MAINT BIT
(1) ;IF NECESSARY (INTERNAL MODE)
(1) 015116 104421 LPRSET ;SET UP LPR REGISTER
(1) 015120 005037 001374 CLR SAVLIN ;INIT LINE INDIC. FOR ERROR
(1) 015124 105037 001425 CLRB DONFLG ;INIT TCR BIT HANDLER FLAG
(1) 015130 113777 001366 164666 MOVB LINE,@DZVTCR ;SET ALL VALID TCR BITS
(1) 015136 106427 000200 MTPS #MASK ;SET CPU STATUS TO DZV11 PRIO,
(1) 015142 012777 000200 164672 MOV #MASK,@DZVRIS ;SET RECEIVER STATUS
(1) 015150 012777 000200 164670 MOV #MASK,@DZVTIS ;SET TRANSMITTER STATUS
(1) 015156 1$:
(2) 015156 012777 015244 164660 MOV #6$,@DZVTIV ;SET UP THE TRANSMITTER INTERRUPT VECTOR
(2) 015164 012777 015266 164646 MOV #7$,@DZVRIV ;SET UP THE RECEIVER INTERRUPT VECTOR
(2) 015172 012777 000200 164642 MOV #MASK,@DZVRIS ;SET THE INTERRUPT VECTOR STATUS
(2) 015200 012777 000200 164640 MOV #MASK,@DZVTIS ;SET TRANSMITTER INTERRUPT PRIORITY
(2) 015206 052777 040040 164574 BIS #TIE!MSENAB,@DZVCSR ;ENABLE THE DEVICE
(1) 015214 005005 CLR R5 ;INIT DELAY COUNTER
(1) 015216 005777 164566 4$: TST @DZVCSR ;TRDY SET?
(1) 015222 100003 BPL 5$ ;IF NOT GO DO DELAY
(1) 015224 000240 NOP ;WAIT FOR INTERRUPT
(1) 015226 000240 NOP
(1) 015230 000420 BR 8$ ;GO CLEAR TIE BIT
(1) 015232 104414 5$: DELAY ;DELAY ROUTINE CALL
(1) 015234 005205 INC R5 ;INC DELAY COUNTER

```

```
(1) 015236 001367 BNE 4$ ;DELAY FINISHED?  
(1) 015240 104003 ERROR 3 ;*TRDY NOT SET!  
(1) 015242 000413 BR 8$ ;GO CLEAR TIE  
(1) 015244 022626 POP2SP ;REMOVE THE INTERRUPT FROM THE STACK  
(1) 015246 042777 040000 164534 6$: BIC #TIE,@DZVCSR ;DON'T LET ANY MORE INTERRUPTS OCCUR  
(1) 015254 105737 001425 TSTB DONFLG ;PROCESSOR ALLOWING INTER?  
(1) 015260 001013 BNE 10$ ;IF YES NO ERROR  
(1) 015262 104010 ERROR 10 ;IF NOT PRINT ERROR  
(1) 015264 000413 BR 9$ ;RETURN TO THE NORMAL FLOW  
(1) 015266 104012 7$: ERROR 12 ;*RECEIVER SHOULD NOT INTERRUPT  
(1) 015270 022626 POP2SP ;POP FOR FAKE RTI  
(1) 015272 042777 040000 164510 8$: BIC #TIE,@DZVCSR ;RESET TRANSMITTER INTERRUPT ENABLE  
(1) 015300 105737 001425 TSTB DONFLG ;INTERRUPTS ENABLED?  
(1) 015304 001403 BEQ 9$ ;IF NOT GET OUT  
(1) 015306 104007 ERROR 7 ;IF YES TRANS FAILED TO INTER.  
(1) 015310 106427 000000 10$: MTPS #CLEAR ;ALLOW INTERRUPTS  
(1) 015314 9$:  
(2) 015314 012777 015420 164522 MOV #11$,@DZVTIV ;SET UP THE TRANSMITTER INTERRUPT VECTOR  
(2) 015322 012777 015424 164510 MOV #12$,@DZVRIV ;SET UP THE RECEIVER INTERRUPT VECTOR  
(2) 015330 012777 000200 164504 MOV #MASK,@DZVRIS ;SET THE INTERRUPT VECTOR STATUS  
(2) 015336 012777 000200 164502 MOV #MASK,@DZVTIS ;SET TRANSMITTER INTERRUPT PRIORITY  
(2) 015344 052777 000140 164436 BIS #RIE!MSENAB,@DZVCSR ;ENABLE THE DEVICE  
(1) 015352 113777 001426 164454 MOV#B TDO,@DZVTDR ;LOAD BUFFER WITH ANY CHAR.  
(1) 015360 005005 CLR R5 ;INIT DELAY ACCUMULATOR  
(1) 015362 105777 164422 13$: TSTB @DZVCSR ;REC. DONE?  
(1) 015366 100003 BPL 14$ ;IF NOT DELAY  
(1) 015370 000240 NOP ;WAIT FOR INTERRUPT  
(1) 015372 000240 NOP  
(1) 015374 000404 BR 18$  
(1) 015376 104414 14$: DELAY ;DELAY FOR INTERRUPT  
(1) 015400 005205 INC R5 ;INCREMENT DELAY COUNTER  
(1) 015402 001367 BNE 13$ ;DELAY FINISHED?  
(1) 015404 104004 ERROR 4 ;*NO RX DONE! (NOT SET)  
(1) 015406 105737 001425 18$: TSTB DONFLG ;PROCESSOR ALLOWING INTERRUPTS?  
(1) 015412 001411 BEQ 15$ ;IF NOT DON'T PRINT ERROR  
(1) 015414 104011 ERROR 11 ;RECEIVER FAILED TO INTERRUPT  
(1) 015416 000407 BR 15$ ;CONTINUE TEST  
(1) 015420 104010 11$: ERROR 10 ;TRANSMITTER SHOULD NOT INTER.  
(1) 015422 000404 BR 16$ ;CONT TEST  
(1) 015424 105737 001425 12$: TSTB DONFLG ;PROCESSOR ALLOWING INTERRUPTS?  
(1) 015430 001001 BNE 16$ ;IF YES DON'T PRINT ERROR  
(1) 015432 104012 ERROR 12 ;*RECEIVER SHOULD NOT INTERRUPT  
(1) 015434 022626 POP2SP ;POP FOR FAKE RTI  
(1) 015436 042777 040100 164344 15$: BIC #RIE!TIE,@DZVCSR ;CLEAR INTERRUPTS  
(1) 015444 105737 001425 TSTB DONFLG ;SECOND TIME THROUGH?  
(1) 015450 001005 BNE 17$ ;IF YES LEAVE TEST  
(1) 015452 105237 001425 INCB DONFLG ;IF NO INDICATE SECOND TEST PASS  
(1) 015456 106427 000000 MTPS #CLEAR ;ALLOW INTERRUPTS  
(1) 015462 000635 BR 1$ ;RESTART TEST  
(1) 015464 106427 000200 17$: MTPS #MASK ;DON'T ALLOW INTERRUPTS  
(1) 015470 104413 DEVICE.CLR ;CLEAR DEVICE, LEAVE TEST
```

9075
9076
9077
9078
9079

```
***** TEST 20 *****  
;*THIS TEST VERIFIES THAT THE RECEIVER WILL  
;*INTERRUPT BEFORE THE TRANSMITTER EVEN  
;*THOUGH THE TRANSMITTER WAS ENABLED
```

```

9080 ;*FIRST. SET PS TO HIGH (MASK INTERRUPTS);
9081 ;*GET RDONE AND TRDY TO SET;
9082 ;*SET TX IE AND RX IE;
9083 ;*CLEAR PS AND EXPECT RX TO INTERRUPT FIRST
9085 ;:* TEST 20
(5) ;*****
(4) 015472 000004 TST20: SCOPE
(4)
(2) 015474 012737 000020 001246 MOV #20,$STSTNM ;LOAD THE NUMBER OF THIS TEST
(1) 015502 012737 004154 001362 MOV #$EOP,NEXT ;POINT TO THE END-OF-PASS HANDLER
9086 015510 104417 DCLASM ;SET DCLR IN CSR AND MNTFLG
9087 015512 104421 LPRSET ;LOAD PAR REGISTER FOR ALL LINES
9088 015514 005037 001374 CLR SAVLIN ;INIT. ERROR LINE INDIC.
9089 015520 012777 015730 164312 MOV #8,@DZVRIV ;SETUP INTERRUPT STUFF
9090 015526 012777 000200 164306 MOV #MASK,@DZVRIS
9091 015534 012777 016016 164302 MOV #12,@DZVTIV
9092 015542 012777 000200 164276 MOV #MASK,@DZVTIS
9093 015550 052777 000040 164232 BIS #MSENAB,@DZVCSR
9094 015556 012702 000001 MOV #1,R2 ;LINE POINTER
9095 015562 030237 001366 3$: BIT R2,LINE ;VALID LINE ?
9096 015566 001515 BEQ 14$ ;IF NOT GO TO NEXT LINE
9097 015570 106427 000200 4$: MTPS #MASK
9098 015574 110277 164224 MOVB R2,@DZVTCR ;SET TCR BIT
9099 015600 005777 164210 TST @DZVRBUF ;VALID DATA?
9100 015604 100001 BPL .+4 ;IT BETTER NOT BE SET
9101 015606 104017 ERROR 17 ;DATA VALID SHOULD NOT BE SET
9102 015610 105777 164174 5$: TSTB @DZVCSR ;RECEIVER DONE ?
9103 015614 100001 BPL .+4
9104 015616 104020 ERROR 20 ;RECEIVER DONE BIT SHOULD NOT BE SET
9105 015620 005005 CLR R5
9106 015622 005004 CLR R4
9107 015624 005777 164160 99$: TST @DZVCSR ;WAIT FOR TRDY
9108 015630 100404 BMI 100$ ;BR IF READY
9109 015632 104414 DELAY ;STALL TIME
9110 015634 005204 INC R4
9111 015636 001372 BNE 99$
9112 015640 104003 ERROR 3 ;TRDY FAILED TO SET
9113 015642 105077 164166 100$: CLRB @DZVTDR ;SEND A ZERO CHARACTER
9114 015646 005004 CLR R4
9115 015650 105777 164134 6$: TSTB @DZVCSR ;IS RDONE SET?
9116 015654 100404 BMI 7$
9117 015656 104414 DELAY
9118 015660 005204 INC R4
9119 015662 001372 BNE 6$
9120 015664 104004 ERROR 4 ;*RDONE FAILED TO SET!
9121 015666 005777 164116 7$: TST @DZVCSR ;TRANS DONE BIT = 1 ?
9122 015672 100401 BMI .+4 ;YES
9123 015674 104003 ERROR 3 ;*NO TRANS DONE FAILED TO SET
9124 ;NOW THAT BOTH TRANSMITTER AND RECEIVER DONE BIT =1
9125 ;SET INTERRUPT ENABLES
9126 015676 052777 040000 164104 BIS #TIE,@DZVCSR
9127 015704 052777 000100 164076 BIS #RIE,@DZVCSR
9128 015712 106427 000000 MTPS #CLEAR ;ALLOW THE INTERRUPTS
9129 015716 000240 NOP
9130 015720 000240 NOP
9131 015722 104007 ERROR 7 ;*TRANSMITTER FAILED TO INTERRUPT

```

```

9132 015724 104011          ERROR 11          ;*RECEIVER FAILED TO INTERRUPT
9133 015726 000435          BR      14$         ;GET OUT
9134
9135          ;RECEIVER INTERRUPT ROUTINE
9136 015730 017704 164060    8$:  MOV     @DZVRBUF,R4    ;ACTUAL
9137 015734 010403          MOV     R4,R3
9138 015736 000303          SWAB   R3
9139 015740 042703 177770    BIC     #^C<7>,R3    ;STRIP JUNK
9140 015744 005737 001372    TST     MODE         ;IS THIS TEST IN STAGGERED MODE?
(1) 015750 100006          BPL     11$         ;IF NOT, SKIP STAGGERED SETUP
(1)
(1)          ;WE MUST NOW INVERT THE LAST BIT OF THE LINE NUMBER
(1) 015752 006203          ASR     R3           ;GET THE LAST BIT INTO THE CARRY BIT
(1) 015754 103402          BCS     9$          ;IF IT IS SET, GO CLEAR IT
(1) 015756 000261          SEC          ;IF IT IS CLEAR SET IT HERE
(1) 015760 000401          BR      10$        ;SKIP THE CLEARING
(1) 015762 000241          9$:  CLC          ;CLEAR THE CARRY BIT (INVERSION OF LINE PARITY)
(1) 015764 006103          10$: ROL     R3           ;GET THE NEW BIT BACK INTO R3
9141 015766 020337 001374    11$: CMP     R3,SAVLIN   ;IS THIS A VALID LINE
9142 015772 001401          BEQ     +4         ;YES
9143 015774 104015          ERROR   15         ;*INVALID LINE
9144 015776 042704 177400    BIC     #^C<377>,R4 ;STRIP JUNK
9145 016002 120504          CMPB   R5,R4       ;DATA COMPARE ?
9146 016004 001401          BEQ     +4         ;YES
9147 016006 104005          ERROR   5          ;*DATA DOES NOT COMPARE
9148 016010 040277 164010    BIC     R2,@DZVTCCR ;CLEAR TCR BIT
9149 016014 000401          BR      13$        ;GO GET OUT OF INTERRUPT MODE
9150          ;TRANSMITTER INTERRUPT SVC ROUTINE
9151 016016 104011          12$: ERROR 11      ;THE RECEIVER INTERRUPT FAILED
9152          ;TO OVERRIDE THE TRANSMITTER
9153 016020 022626          13$: POP2SP        ;REMOVE THE INTERRUPT VECTOR FROM THE STACK
9154 016022 005237 001374    14$: INC     SAVLIN  ;ADJUST FOR NEXT LINE
9155 016026 104420          SHIFT          ;GET THE NEXT POINTER. IF DONE, ADVANCE
9156 016030 000137 015562    JMP     3$         ;OTHERWISE GO DO THE NEXT LINE

```

			:ERROR TABLE	
			.ERRTAB:	
9158				
9159	016034	000000	0	:ERROR 0
9160	016036	000000	0	
9161	016040	000000	0	
9162				
9163	016042	016202	EM1	:ERROR
9164	016044	017020	DH1	
9165	016046	017140	DT1	
9166				
9167	016050	016255	EM2	:ERROR 2
9168	016052	017044	DH2	
9169	016054	017152	DT2	
9170				
9171	016056	016303	EM3	:ERROR 3
9172	016060	017077	DH3	
9173	016062	017170	DT3	
9174				
9175	016064	016342	EM4	:ERROR 4
9176	016066	017077	DH3	
9177	016070	017170	DT3	
9178				
9179	016072	016371	EM5	:ERROR 5
9180	016074	017111	DH4	
9181	016076	017176	DT4	
9182				
9183	016100	016420	EM6	:ERROR 6
9184	016102	017111	DH4	
9185	016104	017176	DT4	
9186				
9187	016106	016457	EM7	:ERROR 7
9188	016110	017077	DH3	
9189	016112	017170	DT3	
9190				
9191	016114	016520	EM10	:ERROR 10
9192	016116	017077	DH3	
9193	016120	017170	DT3	
9194				
9195	016122	016562	EM11	:ERROR 11
9196	016124	017077	DH3	
9197	016126	017170	DT3	
9198				
9199	016130	016620	EM12	:ERROR 12
9200	016132	017077	DH3	
9201	016134	017170	DT3	
9202				
9203	016136	000000	0	
9204	016140	000000	0	
9205	016142	000000	0	
9206				
9207	016144	000000	0	
9208	016146	000000	0	
9209	016150	000000	0	
9210				
9211	016152	016657	EM15	:ERROR 15
9212	016154	000000	0	
9213	016156	000000	0	

9214				
9215	016160	000000	0	
9216	016162	000000	0	
9217	016164	000000	0	
9218				
9219	016166	016721	EM17	;ERROR 17
9220	016170	017077	DH3	
9221	016172	017170	DT3	
9222				
9223	016174	016757	EM20	
9224	016176	017077	DH3	
9225	016200	017170	DT3	

```

9227 ;ERROR MESSAGES
9231 016202 047200 020117 052502 EM1: .ASCIZ <200>/NO BUS REPLY RESPONSE FROM DZV11 REGISTER/
9232 016255 200 042522 044507 EM2: .ASCIZ <200>/REGISTER R/W FAILURE?
9233 016303 200 051124 047101 EM3: .ASCIZ <200>/TRANSMIT READY (TRDY) NOT SET/
9234 016342 051200 041505 044505 EM4: .ASCIZ <200>/RECEIVER DONE NOT SET/
9235 016371 200 040504 040524 EM5: .ASCIZ <200>/DATA COMPARISON ERROR/
9236 016420 042200 053132 030461 EM6: .ASCIZ <200>/DZV11 *RECEIVER BUFFER* ERROR/
9237 016457 200 051124 047101 EM7: .ASCIZ <200>/TRANSMITTER FAILED TO INTERRUPT/
9238 016520 052600 042516 050130 EM10: .ASCIZ <200>/UNEXPECTED TRANSMITTER INTERRUPT/
9239 016562 051200 041505 044505 EM11: .ASCIZ <200>/RECEIVER FAILED TO INTERRUPT/
9240 016620 052600 042516 050130 EM12: .ASCIZ <200>/UNEXPECTED RECEIVER INTERRUPT/
9241 016657 200 041501 044524 EM15: .ASCIZ <200>/ACTION DETECTED ON INVALID LINE./
9242 016721 200 040504 040524 EM17: .ASCIZ <200>/DATA VALID SHOULD NOT BE SET/
9243 016757 200 042522 042503 EM20: .ASCIZ <200>/RECEIVER DONE SHOULD NOT BE SET/
  
```

```

9244
9245 017020 052200 040522 020120 DH1: .ASCIZ <200>/TRAP PC DZV11 REG/
9246 017044 042600 050130 041505 DH2: .ASCIZ <200>/EXPECTED FOUND REGISTER/
9247 017077 200 044514 042516 DH3: .ASCIZ <200>/LINE NO./
9248 017111 200 054105 042520 DH4: .ASCIZ <200>/EXPECTED FOUND LINE/
  
```

```

9249
9250 .EVEN
9254 ;DATA TABLES FOR ERROR MESSAGES
  
```

```

9255 017140 000002 DT1: 2
9256 017142 006 003 .BYTE 6,3
9257 017144 001330 $REG1
9258 017146 006 001 .BYTE 6,1
9259 017150 001326 $REG0
9260
9261 017152 000003 DT2: 3
9262 017154 006 004 .BYTE 6,4
9263 017156 001340 $REG5
9264 017160 006 001 .BYTE 6,1
9265 017162 001336 $REG4
9266 017164 006 001 .BYTE 6,1
9267 017166 001326 $REG0
9268
9269 017170 000001 DT3: 1
9270 017172 003 001 .BYTE 3,1
9271 017174 001374 SAVLIN
9272
9273 017176 000003 DT4: 3
9274 017200 006 004 .BYTE 6,4
9275 017202 001340 $REG5
9276 017204 006 001 .BYTE 6,1
9277 017206 001336 $REG4
9278 017210 003 001 .BYTE 3,1
9279 017212 001374 SAVLIN
  
```

;TABLE OF DELAY TIMES FOR INDIVIDUAL BAUD RATES

```

9280
9288 ;-----
9289
9290
9291 017214 002450 DLYTBL: 2450 ;TIME FOR 50 BAUD
9292 017216 001560 1560 ;TIME FOR 75 BAUD
9293 017220 001120 1120 ;TIME FOR 110 BAUD
9294 017222 000750 750 ;TIME FOR 134 BAUD
9295 017224 000660 660 ;TIME FOR 150 BAUD
  
```


9296	017226	000330	330	:TIME FOR 300 BAUD
9297	017230	000150	150	:TIME FOR 600 BAUD
9298	017232	000060	60	:TIME FOR 1200 BAUD
9299	017234	000040	40	:TIME FOR 1800 BAUD
9300	017236	000030	30	:TIME FOR 2000 BAUD
9301	017240	000020	20	:TIME FOR 2400 BAUD
9302	017242	000010	10	:TIME FOR 3600 BAUD
9303	017244	000001	1	:TIME FOR 4800 BAUD
9304	017246	000001	1	:TIME FOR 7200 BAUD
9305	017250	000001	1	:TIME FOR 9600 BAUD
9306	017252	000001	1	:TIME OF DELAY FOR 19200 BAUD
9307				
9308				
9309				

:DELAYS WERE COMPUTED TO ALLOW MAXIMUM TIME AT EACH BAUD RATE
:FOR ALL TESTS TO FUNCTION CORRECTLY ON A LSI11.

```
9392
9393      ; INIT $BASE AND $VECT1 AND TWEAK THE '$GETPAR' CALLING
9394      ; SEQUENCE TO ACCEPT THE VALID FALCON RANGE.
9395
9396 017254 023727 001174 160010 FALCINI: CMP $BASE,#ABASE ; IS $BASE VIRGIN ??      ;;GPA
9397 017262 001003                BNE 1$ ; SKIP NEXT IF NOT      ;;GPA
9398 017264 012737 174040 001174    MOV #174040,$BASE ; YES, SET ENGINEERING DEFAULT ;;GPA
9399 017272 023727 001170 000300 1$: CMP $VECT1,#AVECT1 ; IS $VECT1 VIRGIN ??      ;;GPA
9400 017300 001003                BNE 2$ ; SKIP NEXT IF NOT      ;;GPA
9401 017302 012737 000370 001170    MOV #370,$VECT1 ; YES, SET ENGINEERING DEFAULT ;;GPA
9402 017310 012737 017354 002370 2$: MOV #3$,GETCSR+2 ; SUBSTITUE CSR TEXT... ;;GPA
9403 017316 012737 174000 002374    MOV #174000,GETCSR+6 ; ;GPA
9404 017324 012737 177770 002376    MOV #177770,GETCSR+10 ;...AND VALID RANGE. ;GPA
9405 017332 012737 017416 002414    MOV #4$,GETVFC+2 ; SUBSTITUE VECTOR TEXT... ;GPA
9406 017340 005037 002420                CLR GETVEC+6 ; ;GPA
9407 017344 012737 000370 002422    MOV #370,GETVEC+10 ;...AND VALID RANGE. ;GPA
9408 017352 000207                RETURN ; RETURN TO CALLER. ;GPA
9409
9410 017354 030600 052123 041440 3$: .ASCIZ <200>'1ST CSR ADDRESS (174000:177770) ' ;;GPA
    017362 051123 040440 042104
    017370 042522 051523 024040
    017376 033461 030064 030060
    017404 030472 033467 033467
    017412 024460 000040
9411 017416 030600 052123 053040 4$: .ASCIZ <200>'1ST VECTOR ADDRESS (000:370) ' ;;GPA
    017424 041505 047524 020122
    017432 042101 051104 051505
    017440 020123 030050 030060
    017446 031472 030067 020051
    017454 020040 000040
    .EVEN ;;GPA
9412
9413
9426
9427
9428      ; THE FOLLOWING .INIT CALL WAS ADDED TO INIT BRKVEC AND LKVEC
    017460 POINT=. ;SAVE POINTER
    (1) 000100 000100 .=100
    (1) 000102 017460 $CLKVEC ;LKVEC HANDLER
    (1) 000140 000300 300 ;INTERRUPT HANDLER PH.
    (1) 000142 000140 .=140 ;BRKVEC
    (1) 000144 170000 170000 ;ODT START ADDRESS
    (1) 000146 000300 300 ;PRIORITY
    (1) 017460 017460 .=POINT ;RESTORE POINTER
    (1) 017460 104402 017466 $CLKVEC: TYPE,CLKMES
    (1) 017464 000000 HALT
    (1) 017466 005015 045514 042526 CLKMES: .ASCIZ <15><12>/LKVEC INTERRUPT - DISCONNECT LTC /
    (1) 017474 020103 047111 042524
    (1) 017502 051122 050125 020124
    (1) 017510 020055 044504 041523
    (1) 017516 047117 042516 052103
    (1) 017524 046040 041524 000040
9429 017532 000001 KXTFLAG:1 ;TO INDICATE THE PROCESSOR IS FALCON
9430 000001 .END
```

ABASE = 160010	8360#	8703	9396	
ACDW1 = 000017	8365#	8703		
ACDW2 = 000000	8703			
ACPUOP= 000000	8703			
ACTIVE 001420	8703#*	8705*		
ADDW0 = 017470	8363#	8703		
ADDW1 = 017470	8363#	8703		
ADDW10= 017470	8363#	8703		
ADDW11= 017470	8363#	8703		
ADDW12= 017470	8363#	8703		
ADDW13= 017470	8363#	8703		
ADDW14= 017470	8363#	8703		
ADDW15= 017470	8363#	8703		
ADDW2 = 017470	8363#	8703		
ADDW3 = 017470	8363#	8703		
ADDW4 = 017470	8363#	8703		
ADDW5 = 017470	8363#	8703		
ADDW6 = 017470	8363#	8703		
ADDW7 = 017470	8363#	8703		
ADDW8 = 017470	8363#	8703		
ADDW9 = 017470	8363#	8703		
ADEVCT= 000000	8703			
ADEVN = 000001	8366#	8703		
ADRCNT 005751	8705#*			
ADVANC= 104400	8703#	8705	8710	8784
AENV = 000000	8703			
AENVN = 000000	8703			
AFATAL= 000000	8703			
AMADR1= 000000	8703			
AMADR2= 000000	8703			
AMADR3= 000000	8703			
AMADR4= 000000	8703			
AMAMS1= 000000	8703			
AMAMS2= 000000	8703			
AMAMS3= 000000	8703			
AMAMS4= 000000	8703			
AMSGAD= 000000	8703			
AMSGLG= 000000	8703			
AMSGTY= 000000	8703			
AMTYP1= 000000	8703			
AMTYP2= 000000	8703			
AMTYP3= 000000	8703			
AMTYP4= 000000	8703			
APASS = 000000	8703			
APRIOR= 000000	8703			
APTCU= 000040	8705#			
APTENV= 000001	8705#			
APTSIZ= 000200	8705#			
APTSP0= 000100	8705#			
ASWREG= 000000	8703			
ATESTN= 000000	8703			
AUNIT = 000000	8703			
AUSWR = 000000	8703			
AUTO.S 011320	8703	8705#		
AVECT1= 000300	8364#	8703	9399	
AVECT2= 000000	8703			

SW04	=	000020	8703#						
SW05	=	000040	8703#						
SW06	=	000100	8703#						
SW07	=	000200	8703#						
SW08	=	000400	8703#	8705					
SW09	=	001000	8703#	8705					
SW1	=	000002	8703#						
SW10	=	002000	8703#	8705					
SW11	=	004000	8703#						
SW12	=	010000	8703#	8705					
SW13	=	020000	8703#	8705					
SW14	=	040000	8703#						
SW15	=	100000	8703#						
SW2	=	000004	8703#						
SW3	=	000010	8703#						
SW4	=	000020	8703#						
SW5	=	000040	8703#						
SW6	=	000100	8703#						
SW7	=	000200	8703#						
SW8	=	000400	8703#						
SW9	=	001000	8703#						
S110	=	001000	8703#						
S1200	=	003400	8703#						
S134	=	001400	8703#						
S150	=	002000	8703#						
S1800	=	004000	8703#						
S19200	=	007400	8703#						
S2000	=	004400	8703#						
S2400	=	005000	8703#						
S300	=	002400	8703#						
S3600	=	005400	8703#						
S4800	=	006000	8703#						
S50	=	000000	8703#						
S600	=	003000	8703#						
S7200	=	006400	8703#						
S75	=	000400	8703#						
S9600	=	007000	8703#						
TBITVE	=	000014	8703#						
TCR0	=	000001	8703#	8728					
TCR1	=	000002	8703#	8728					
TCR2	=	000004	8703#	8728					
TCR3	=	000010	8703#	8728					
TD0	=	001426	8703#	8705	9005	9009	9023*	9074	
TD1	=	001430	8703#						
TD2	=	001432	8703#						
TD3	=	001434	8703#						
TEIGHT	=	002106	8703#						
TFIVE	=	002114	8703#						
TIE	=	040000	8703#	8727	8740	9074	9126		
TKVEC	=	000060	8703#						
TLAST	=	015472	8705	9285#					
TLO	=	000000	8703#						
TL1	=	000400	8703#						
TL2	=	001000	8703#						
TL3	=	001400	8703#						
TMTBL	=	002050	8703#						

TPVEC =	000064	8703#		
TRAPVE=	000034	8703#		
TRDY =	100000	8703#	8729	8824
TRTVEC=	000014	8703#		
TRO	001436	8703#		
TR1	001440	8703#		
TR2	001442	8703#		
TR3	001444	8703#		
TSEVEN	002110	8703#		
TSIX	002112	8703#		
TST1	012024	8705	8710#	
TST10	013160	8767	8768#	
TST11	013244	8768	8781#	
TST12	013440	8781	8824#	
TST13	013572	8824	8835#	
TST14	014062	8835	8899#	
TST15	014404	8899	8980#	
TST16	014674	8980	9041#	
TST17	015076	9041	9074#	
TST2	012214	8710	8716#	
TST20	015472	9074	9085#	9285
TST21 =	***** U	9085		
TST3*	012260	8716	8727#	
TST4	012436	8727	8728#	
TST5	012642	8728	8729#	
TST6	012744	8729	8736#	
TST7	013074	8736	8767#	
TTST	004402	8703*	8705#	
TWOSTO=	000040	8703#		
TYPDAT	006706	8705#		
TYPE =	104402	8703#	8705	9428
TYPMSG	006576	8705#		
T110	002054	8703#		
T1200	002066	8703#		
T134	002056	8703#		
T150	002060	8703#		
T1800	002070	8703#		
T2000	002072	8703#		
T2400	002074	8703#		
T300	002062	8703#		
T3600	002076	8703#		
T4800	002100	8703#		
T50	002050	8703#		
T600	002064	8703#		
T7200	002102	8703#		
T75	002052	8703#		
T9600	002104	8703#		
VECMAP	011634	8705#		
WRDCNT	006220	8705#*		
WTBS.F	006674	8705#		
XBX	006464	8705#		
XCSR	004340	8705#		
XERR	004362	8705#		
XHEAD	010062	8703	8705#	
XMTCNT	001400	8703#		
XMTLIN	001376	8703#		

.DELAY	006302	8703	8705#
.DEVIC	006250	8703	8705#
.ERRTA	016034	8705	9159#
.INSTE	005532	8703	8705#
.INSTR	005426	8703	8705#
.INST1	005446	8705#	
.LPRSE	006352	8703	8705#
.MSG	005450	8705#*	
.PARAM	005552	8703	8705#
.PARMD	011170	8703	8705#
.PAWCH	010304	8703	8705#
.RESOS	006012	8703	8705#
.SAVOS	005752	8703	8705#
.SCOPE	004370	8703	8705#
.SCOPI	004634	8703	8705#
.SETFL	010164	8703	8705#
.SHIFT	006334	8703	8705#
.START	002116	8703#	
.TRPSR	006226	8703	8705#
.TRPTA	001742	8703#	8705
.TYPE	004660	8703	8705#
.\$ASTA=	***** U	8705	
.\$X =	001446	8703#	

ERRORS DETECTED: 0

CNDZAA,CNDZAA/CR/NL:TOC=CNMAC2.SML,CNDZAA.P11
RUN-TIME: 18 19 1 SECONDS
RUN-TIME RATIO: 133/39=3.3
CORE USED: 50K (100 PAGES)