IDENTIFICATION

PRODUCT CODE: AC-E006C-MC
PRODUCT NAME: CVOVCCO DLV11-F OFFLINE TEST
PRODUCT DATE: AUGUST 1984
AUTHOR: ODIE CHOEAT
MAINTAINER: DIAGNOSTIC ENGINEERING GROUP

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSIDERED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION DISCLAIMS ANY RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE OR EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1977,1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL PDP UNIBUS MASSBUS
DEC DECUS DECTAPE
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>GENERAL PROGRAM INFORMATION.</td>
</tr>
<tr>
<td>1.1</td>
<td>PROGRAM PURPOSE (ABSTRACT).</td>
</tr>
<tr>
<td>1.2</td>
<td>SYSTEM REQUIREMENTS.</td>
</tr>
<tr>
<td>1.3</td>
<td>RELATED DOCUMENTS AND STANDARDS.</td>
</tr>
<tr>
<td>1.4</td>
<td>DIAGNOSTIC HIERARCHY PREREQUISITES.</td>
</tr>
<tr>
<td>1.5</td>
<td>ASSUMPTIONS.</td>
</tr>
<tr>
<td>2.0</td>
<td>OPERATING INSTRUCTIONS.</td>
</tr>
<tr>
<td>2.1</td>
<td>LOADING AND STARTING PROCEDURES.</td>
</tr>
<tr>
<td>2.2</td>
<td>SPECIAL ENVIRONMENTS.</td>
</tr>
<tr>
<td>2.3</td>
<td>OPERATIONAL SWITCH SETTINGS</td>
</tr>
<tr>
<td>2.4</td>
<td>PROGRAM OPTIONS.</td>
</tr>
<tr>
<td>2.5</td>
<td>EXECUTION TIMES.</td>
</tr>
<tr>
<td>3.0</td>
<td>ERROR INFORMATION.</td>
</tr>
<tr>
<td>3.1</td>
<td>ERROR REPORTING PROCEDURE.</td>
</tr>
<tr>
<td>3.2</td>
<td>ERROR HIALTS.</td>
</tr>
<tr>
<td>4.0</td>
<td>PERFORMANCE AND PROGRESS REPORTS.</td>
</tr>
<tr>
<td>4.1</td>
<td>PERFORMANCE REPORTS.</td>
</tr>
<tr>
<td>5.0</td>
<td>DEVICE INFORMATION TABLES.</td>
</tr>
<tr>
<td>6.0</td>
<td>SUMMARY OF TESTS AND SPECIAL SUBROUTINES</td>
</tr>
</tbody>
</table>
1.0 GENERAL PROGRAM INFORMATION.

-----------

1.1 PROGRAM PURPOSE (ABSTRACT).

THIS DIAGNOSTIC IS A LOGIC TEST TO VERIFY THE OPERATION OF THE
DLVII-F SERIAL LINE INTERFACE. THE USER CAN SELECTIVELY
ENABLE AND DISABLE TESTING OF THE OPTIONS BY ALTERING THE
CONTENTS OF 'HUSER'. THE DIAGNOSTIC IS DESIGNED TO TEST AND
DETECT FAULTS IN THE INTERFACE LEVEL (NOT THE CHIP LEVEL).
THIS TEST OPERATES ON UP TO SIXTEEN (16) IDENTICALLY CONFIGURED
DLVII-F SERIAL LINE INTERFACES. THE DEFAULT ADDRESSES ARE:

  177560 - CONSOLE INTERFACE DEVICE ADDRESS
  175610 - FIRST SERIAL LINE ADDRESS OF 15 CONSECUTIVE
          SERIAL LINE DEVICES.

60 - VECTOR FOR CONSOLE DEVICE INTERFACE.
300 - VECTOR FOR FIRST OF 15 DEVICES.

THIS PROGRAM IS DESIGNED TO RUN ON ANY PDP-11 WITH 4K OF
MEMORY AND A DLVII-F (LSI-BUS) MODULE. IT CAN RUN UNDER XIXDP,
APT, AND ACT MONITORS, AND ON PROCESSORS WITH NO HARDWARE
SWITCH REGISTER. A POWER FAILURE WILL CAUSE THE DIAG-
NOSTIC TO RESTART.

1.2 SYSTEM REQUIREMENTS.

    HARDWARE REQUIREMENTS:

    ANY PDP-11 FAMILY PROCESSOR
    4K MEMORY - MINIMUM
    A SPECIAL WRAP CONNECTOR OR EQUIVALENT (OPTIONAL)

    SOFTWARE REQUIREMENTS:

    THIS DIAGNOSTIC IS DESIGNED TO RUN IN ANY OF THE
    FOLLOWING WAYS:
    STAND ALONE
    WITH APT MONITOR
    WITH ACT MONITOR
    WITH XIXDP MONITOR (CHAINABLE)

1.3 RELATED DOCUMENTS AND STANDARDS.

    DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS 175-003-009-02
    APT
    ACT
    SYSPAC

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES.
NO SPECIAL DIAGNOSTICS ARE REQUIRED TO RUN BEFORE THIS, BUT THE PROCESSOR, MEMORY, AND BUS ARE ASSUMED TO BE FULLY OPERATIONAL.

1.5 ASSUMPTIONS.

THIS DIAGNOSTIC ASSUMES THAT THE OPERATOR HAS INITIALIZED LOCATION '#USR#' AND '#DEV#' TO THE PROPER VALUES.

2.0 OPERATING INSTRUCTIONS.

---------------

2.1 LOADING AND STARTING PROCEDURES.

USE STANDARD PROCEDURE FOR POP-11 ABSOLUTE BINARY FORMATTED MEDIA.

THIS DIAGNOSTIC HAS ONLY ONE (1) STARTING ADDRESS, 200 FOR START AND RESTART.

THE USER CAN SELECT A SPECIFIC TEST TO BE EXECUTED BY SETTING SWITCH # IN THE SWITCH REGISTER AND THE TEST NUMBER (IN OCTAL) IN THE LOWER BYTE. (NOTE: ALL TESTS PREVIOUS TO THE SELECTED ONE ARE EXECUTED WITHOUT ITERATIONS.)

2.2 SPECIAL ENVIRONMENTS.

THIS DIAGNOSTIC FOLLOWS THE STANDARD PROCEDURE FOR RUNNING USER AP, ACT, JOCTOL MONITORS, AS DESCRIBED IN THEIR RESPECTIVE PROCEDURES MANUAL AND SYSPAC PACKAGE.

2.3 OPERATIONAL SWITCH SETTINGS

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

CONTROL:

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

1) TYPE CONTROL C <T>: THIS WILL ALLOW THE TTY TO ENTER DATA INTO LOC. 176 AT SELECTED POINTS WITHIN THE
PROGRAM.

2) THE MACHINE WILL THEN TYPE: ' SWR=XXXXX NEW=' (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER.)

3) AFTER THE 'NEW=' HAS BEEN TYPED THEN THE OPERATOR CAN DO ONE OF THE FOLLOWING AT THE TTY:

   A) TYPE A NUMBER TO BE LOADED INTO LOC. 176 FOLLOWED BY A <CR>. (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED). LEADING ZEROS NEED NOT BE TYPED, AND IF MORE THAN 6 DIGITS ARE TYPED THE LAST 6 WILL BE USED. IF A <CR> IS THE FIRST KEY DEPRESSED THE SOFTWARE SWITCH REGISTER CONTENTS WILL NOT BE CHANGED.

   B) IF A CONTROL U <TI> IS DEPRESSED THEN THE PROGRAM WILL SEND YOU BACK TO STEP 3.

   C) IF THE INPUT CHARACTER IS NOT ONE OF THE CHARACTERS MENTIONED ABOVE THEN A QUESTION MARK (?) WILL BE TYPED FOLLOWED BY A CARRIAGE RETURN AND A LINE FEED SEQUENCE THEN PROCEED FROM STEP 3 (ERASING ALL PREVIOUS INPUT).

DYNAMIC SWITCH REGISTER

---

BIT 15 - HALT ON ERROR
14 - LOOP ON TEST
13 - INHIBIT ERROR TYPEOUTS
12 - (UNUSED)
11 - INHIBIT ITERATIONS
10 - BELL ON ERROR
9 - LOOP ON ERROR
8 - LOOP ON TEST IN SWR<7:0>
7:0 - TEST NUMBER TO LOOP ON (USED WITH BIT 8)

2.4 PROGRAM OPTIONS.

THIS PROGRAM WILL SUPPORT TESTING OF MULTIPLE DLV11-F'S. IT REQUIRES THE ADDRESS OF THE FIRST RCSR (STORED AT 'IBASE'), AND ITS INTERRUPT VECTOR (STORED AT 'IVECT1') AND WILL BE ABLE TO ADDRESS ANY DLV11-F STARTING AT THE SPECIFIED BASE ADDRESS UP TO 16 CONSECUTIVE DEVICES.

EXAMPLES:  IBASE: 175610
           IVECT1: 300

THE PROGRAM WILL BE ABLE TO TEST ANY DLV11-F WITHIN THE ADDRESS RANGE 175610 --> 176000

IBASE AND IVECT1 DEFAULT TO 175610 AND 300 RESPECTIVELY.
THE PROGRAM ASSOCIATES UNIT NUMBERS AS FOLLOWS: (NUMBERS IN
PARENTHESES ARE OCTAL)

UNIT#0 -- BASE ADDRESS STORED AT '#BASE'
ASSOCIATED BASE VECTOR STORED AT '#VECT1'

UNIT#1 -- BASE ADDRESS + (10)
BASE VECTOR + (10)
UP TO

UNIT#14 -- BASE ADDRESS + (160)
BASE VECTOR + (160)

LOCATION '#DEVH' IS USED AS A BIT MAP TO INDICATE WHICH UNIT
NUMBERS ARE PRESENT AND WILL BE TESTED.

BIT 15 BIT 14 BIT 13 BIT 12

! CON- ! UNIT ! ! UNIT ! UNIT !
! SOL- ! 14 ! ! 01 ! 00 !

A BIT MAP CAN BE ENTERED AT '#DEVH' PRIOR TO STARTING THE
PROGRAM.

EXAMPLE:

#BASE: 175610
#VECTOR: 250
#DEVH: 100013

THE PROGRAM WILL TEST-

UNIT#0 175610 300
UNIT#1 175620 310
UNIT#3 175640 330
CONSOLE 177560 60

OPTIONS

LOCATION '#USER' CONTAINS ALL THE USER SELECTABLE OPTIONS. THE
VALUES IN THIS WORD MUST CONFORM TO THE ACTUAL BOARD
CONFIGURATION.
THE DEFAULT VALUE OF USWR IS AS FOLLOWS:

<table>
<thead>
<tr>
<th>BIT POSITION</th>
<th>DEFINITION</th>
<th>DEFAULT VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-3</td>
<td># OF DATA BITS</td>
<td>10(8) = 0</td>
</tr>
<tr>
<td>4</td>
<td>PARITY ENABLED - (SEE NOTE)</td>
<td>0 = NO</td>
</tr>
<tr>
<td>5</td>
<td>EVEN ODD PARITY - (SEE NOTE)</td>
<td>0 = ODD</td>
</tr>
<tr>
<td>6</td>
<td>COMMON SPEED</td>
<td>1 = YES</td>
</tr>
<tr>
<td>7</td>
<td>PROGRAMMABLE BAUD RATE</td>
<td>0 = NO</td>
</tr>
<tr>
<td>8-11</td>
<td>BAUD RATE OFFSET (SEE FOLLOWING NOTE)</td>
<td>02(8) = 110 BAUD</td>
</tr>
<tr>
<td>12</td>
<td>BREAK GENERATION ENABLED</td>
<td>1 = YES</td>
</tr>
<tr>
<td>13</td>
<td>WRAP CONNECTOR INSTALLED</td>
<td>0 = NO</td>
</tr>
<tr>
<td>14</td>
<td>MAINT JUMPER (SEE NOTE)</td>
<td>0 = NO</td>
</tr>
<tr>
<td>15</td>
<td>ERROR BITS ENABLED</td>
<td>0 = NO</td>
</tr>
</tbody>
</table>

NOTE ON BITS <4:5>

THIS DIAGNOSTIC DOES NOT TEST THE PARITY LOGIC.

NOTE ON BITS <7:11>

WHEN THE PROGRAMMABLE BAUD RATE OPTION IS ENABLED, THE PROGRAMMABLE BAUD RATE TEST WILL EXIT WITH THE BAUD RATE SET TO THE SELECTED VALUE. TO CHANG THE DEFAULT VALUE OF 110 BAUD REPLACE BITS <11:8> WITH THE OFFSET INDICATED IN THE TABLE AT THE END OF THE PBR TEST. (TEST #16)

NOTE ON BIT 14

THIS SWITCH WHEN ON WILL ALLOW THE DIAGNOSTIC TO TEST IN MAINTAINCE MODE. IT IS ASSUMED THAT THE MAINTAINCE JUMPER IS INSTALLED ON ALL OF THE DLVII-F MODULES WHEN THIS BIT IS SET.

DLVII-F INDIVIDUAL TEST REQUIREMENTS TABLE

<table>
<thead>
<tr>
<th>TEST #</th>
<th>CONSOLE DEVICE</th>
<th>APT ENVIRONMENT</th>
<th>(MAINT) BIT SET</th>
<th>(WRAP CON) BIT SET</th>
<th>(ERROR BITS) BIT SET</th>
<th>(COM SPD) BIT SET</th>
<th>(BREAK) BIT SET</th>
<th>(PROG BAUD RATE) BIT SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST 1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>10</td>
</tr>
<tr>
<td>TEST 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEST 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** TEST WILL NOT RUN IF THIS CONDITION IS TRUE.
-- TEST WILL NOT RUN IF THIS CONDITION IS FALSE.
** TEST WILL NOT RUN IF ALL OF THE CONDITIONS IN THIS COLUMN ARE FALSE.
2.5 EXECUTION TIMES.

EXECUTION TIMES ARE FOR AN LSI-11 PROCESSOR WITH ALL OPTIONS
ENABLED ON THE DLV11-F (EXCEPT FOR PROGRAMMABLE BAUD RATE), AT
110 BAUD, AND NOT AT THE CONSOLE ADDRESS.

FIRST PASS - 90 SECONDS
ADDITIONAL PASSES 95 SECONDS
ADDITIONAL DEVICES 95 SECONDS

THE TEST TIME IS BAUD RATE DEPENDANT; HIGHER BAUD GIVES
SHORTER PASS TIMES.

IF THE DIAGNOSTIC IS RUN AT THE CONSOLE
ADDRESS THE RUNNING TIME IS 5 SECONDS PER PASS.

3.0 ERROR INFORMATION.

3.1 ERROR REPORTING PROCEDURE.

SINCE THIS DIAGNOSTIC WAS DESIGNED TO FIT IN 4-K OF MEMORY THE
ERROR TYPEOUT IS VERY BRIEF. THE FORMAT OF THE ERROR TYPEOUT
IS AS FOLLOWS:

TEST#_____,ERROR#_____,PC=_____,ADDRESS=______,VECTOR=_____

WHERE ALL VALUES TYPED ARE OCTAL.

THE ADDRESS AND VECTOR REFER TO THE FAILING SLU'S.

FOR FURTHER INFORMATION THE LISTING MUST BE CONSULTED.

BITS 15, 13, 10 AND 9 OF THE SWITCH REGISTER CONTROL THE
SEQUENCE OF EVENTS AFTER AN ERROR IS CAUGHT.

BIT 15 - CAUSES THE PROGRAM TO HALT IN THE ERROR
ROUTINE. CONTINUEING THE PROGRAM CAUSES IT TO
PROCEED.

BIT 13 - DISABLES THE PRINTING OF THE ERROR MESSAGE.

BIT 10 - CAUSES THE BELL TO RING ON ERROR.

BIT 9 - CAUSES THE DIAGNOSTIC TO LOOP FROM BEGINNING
OF TST TO ERROR.

THE ERROR ROUTINE SUPPORTS THE CONTROL G FUNCTION.

3.2 ERROR HALTS.

THE ONLY HALT IN THIS DIAGNOSTIC IS IN THE ERROR ROUTINE, AND
IS EXECUTED ONLY IF BIT 15 OF THE SWITCH REGISTER IS A ONE
WHEN AN ERROR OCCURS.
PERFORMANCE AND PROGRESS REPORTS.

PERFORMANCE REPORTS.

AS EACH DEVICE COMPLETES ONE PASS OF THE DIAGNOSTIC THE FOLLOWING WILL BE TYPED:

CSR: ___ VECTOR: ___ ERRORS: ___

WHERE 'CSR: ___' IS THE DEVICE CSR UNDER TEST
'VECTOR: ___' IS THE ASSOCIATED VECTOR
AND 'ERRORS: ___' IS THE TOTAL NUMBER OF ERRORS ON THIS DEVICE ON THIS PASS.

NOTE

THIS IS TYPED AFTER THE DEVICE HAS COMPLETED ITS PASS.

AFTER ALL DEVICES HAVE BEEN EXERCISED AN END PASS STATEMENT IS TYPED:

"ENDPASS..."

DEVICE INFORMATION TABLES.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

------------------------------------------------------------------------


6.0 SUMMARY OF TESTS AND SPECIAL SUBROUTINES.

TEST 1 ADDRESSABILITY

This test verifies that the address as placed in the hardware p-table to be correct and the DLVII-F responds to that address space.

TEST 2 BREAK - TCSR0 SET, CLEAR, RESET

TEST 3 MAINT - TCSR2 SET, CLEAR, RESET

TEST 4 XMITIE - TCSR6 SET, CLEAR, RESET

TEST 5 RCVRIE - RCSR6 SET, CLEAR, RESET

The following 4 tests verify that reset (INIT) initializes read only bits.

TEST 6 RCVRDONE - RCSR 7 - IS CLEARED BY INIT

TEST 7 RCVRACT - RCSR 11 - 15 CLEARED BY INIT

TEST 10 XMITREG - TCSR 7 - IS SET BY INIT

TEST 11 XMIT RDY - TCSR 7 - CLEARS WHEN TBUF IS LOADED WITH A CHARACTER AND THAT IT SETS WITHIN A REASONABLE AMOUNT OF TIME.

TEST 12 OUTPUTTING A (CHR FROM TBUF (WITH MAINT SET)

RESULTS IN RCVRDONE SETTING WITHIN A REASONABLE AMOUNT OF TIME AND THAT RESET CLEARS THE BIT.
TEST 13 RCVROONE IS CLEARED BY READING RBUF

TEST 14 RCVRCLT - RCSR 11 - SETS WHEN A START BIT IS
RECEIVED AND CLEARS WHEN RCVRONE - RCSR 7 -
SETS

TEST 15 OVERRUN BIT - RBUF 14

TEST 16 PROGRAMMABLE BAUD RATE TEST TEST AT ALL SPEEDS
AVAILABLE A COMPARISON WILL BE MADE TO SEE IF
NEW TIME IS LESS THAN PREVIOUS.

TEST 17 TRANSMITTER INTERRUPT LOGIC TEST
LOGICALLY THIS IS 4 SEPARATE TESTS
A) DOES TRANSMITTER INTERRUPT LOGIC WORK
B) AT PRIORITY OF 0
C) AND ONLY ONCE
D) BUT NOT WITH INTERRUPT ENABLE CLEAR

TEST 20 RECEIVER INTERRUPT LOGIC TEST THIS TEST COVERS ALL
OF THE RECEIVER SIDE OF THE INTERRUPT LOGIC IN
CHARACTER MODE.

TEST 21 TEST ACTUAL DATA TRANSFERED NON-_INTERRUPT
MAINTENANCE BIT SET

TEST 22 TEST DATA THROUGH WRAP

TEST 23 FULL DATA TRANSFER WITH INTERRUPTS AND MAINTENANCE
MODE.

TEST 24 TEST BREAK GENERATION LOGIC TRANSMIT KNOWN CHAR
WITH BREAK SET AND COMPARE RECEIVED WITH 0.

TEST 25 NOT A TEST - SEND BACK TO LOOP
NOTE

FOR ALL OF THE FOLLOWING ROUTINES THE USE
OF (R5) IS PART OF THE LINKAGE MECHANISM
BETWEEN THE CALLER AND THE CALLED.

ROUTINE: TIMER
-------------

THIS ROUTINE IS USED TO TEST THE STATUS OF
ANY BIT IN ANY REGISTER.

INPUTS:
- HOWLONG THE MAXIMUM AMOUNT OF TIME TO
  SPEND IN THIS ROUTINE.
- WHICHBIT A MASK WITH THE BIT(S) SET THAT
  ARE TO BE CHECKED
- REG A POINTER TO THE REGISTER TO BE
  CHECKED
- SETCLR THE DESIRED RESULTS -- EITHER SET
  OR CLEAR

OUTPUT:
- THE 'C' BIT IS SET TO INDICATE AN ERROR BUT IT
  IS TESTED BY THE IF ERROR STATEMENT.

ROUTINE: DATLNG
-------------

THIS ROUTINE SETS UP A MASK FOR DATA, WITH -

INPUT:
- NOTHING IS PASSED TO THIS ROUTINE BUT GLOBAL
  INFORMATION IS ASSUMED TO EXIST:
  #USWR -- THE WORD FOR SOFTWARE PARAMETERS
  DATA -- A MASK FOR THE LOCATION OF THE OCTAL
  NUMBER OF DATA BITS

OUTPUT:--
- MASK -- A MASK OF BINARY ZEROS RIGHT-JUSTIFIED
  THE NUMBER OF WHICH IS DEFINED IN #USWR WORD.

ROUTINE: WAIT
-------------

THIS ROUTINE IS USED TO DELAY EXECUTION OF THE
MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME.
THIS IS ACCOMPLISHED BY INCREMENTING A
REGISTER UP TO A LIMIT. THE INNER LOOP IS SET
TO APPROXIMATE 1 MICRO SEC.

SERVICE ROUTINE: INTSRV
----------

THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT
'INTFLAG' each time it is called. It assumes that the main calling routine will know what to look for.

**ROUTINE: CYCLE**

This routine causes ADRS to point to the address of DLV11-F under test. ADRS +2 to point to the vector of the DLV11-F under test. It keeps track of the current device and bit masks.
.TITLE MAINDEC-11-DVDVC.C
.COPYRIGHT (C) 1977
*DIGITAL EQUIPMENT CORP.
*Maynard, Mass. 01754
*
*PROGRAM BY ODES CHOATE
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*
.SBTTL OPERATIONAL SWITCH SETTINGS

<table>
<thead>
<tr>
<th>SWITCH</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>HALT ON ERROR</td>
</tr>
<tr>
<td>14</td>
<td>LOOP ON TEST</td>
</tr>
<tr>
<td>13</td>
<td>INHIBIT ERROR TYPEOUTS</td>
</tr>
<tr>
<td>11</td>
<td>INHIBIT ITERATIONS</td>
</tr>
<tr>
<td>10</td>
<td>BELL ON ERROR</td>
</tr>
<tr>
<td>9</td>
<td>LOOP ON ERROR</td>
</tr>
<tr>
<td>8</td>
<td>LOOP ON TEST IN SHR&lt;7:0&gt;</td>
</tr>
</tbody>
</table>

.SBTTL BASIC DEFINITIONS

*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
EQUIV EXT.ERROR : BASIC DEFINITION OF ERROR CALL
EQUIV 107,SCOPE  : BASIC DEFINITION OF SCOPE CALL

*MISSCIRISEQUENCE DEFINITIONS

MT=  11  : CODE FOR HORIZONTAL TAB
LF=  12  : CODE FOR LINE FEED
CR=  15  : CODE FOR CARRIAGE RETURN
CRLF= 200  : CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776  : PROCESSOR STATUS WORD
EQUIV PS,PSM
STKLMT= 177774  : STACK LIMIT REGISTER
PSRM= 177772  : PROGRAM INTERRUPT REQUEST REGISTER
DSRM= 177570  : HARDWARE SWITCH REGISTER
DOISP= 177570  : HARDWARE DISPLAY REGISTER

*GENERAL PURPOSE REGISTER DEFINITIONS

R0=  #0  : GENERAL REGISTER
R1=  #1  : GENERAL REGISTER
R2=  #2  : GENERAL REGISTER
R3=  #3  : GENERAL REGISTER
R4=  #4  : GENERAL REGISTER
R5=  #5  : GENERAL REGISTER
R6=  #6  : GENERAL REGISTER
R7=  #7  : GENERAL REGISTER
SP=  #6  : STACK POINTER
PC=  #7  : PROGRAM COUNTER

*PRIORITY LEVEL DEFINITIONS
PRO=  0  : PRIORITY LEVEL 0
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>672</td>
<td>000040</td>
</tr>
<tr>
<td>673</td>
<td>000100</td>
</tr>
<tr>
<td>674</td>
<td>000140</td>
</tr>
<tr>
<td>675</td>
<td>000200</td>
</tr>
<tr>
<td>676</td>
<td>000240</td>
</tr>
<tr>
<td>677</td>
<td>000300</td>
</tr>
<tr>
<td>678</td>
<td>000340</td>
</tr>
</tbody>
</table>

### PRIORITY LEVELS
- PRI.1: 40
- PRI.2: 100
- PRI.3: 140
- PRI.4: 200
- PRI.5: 240
- PRI.6: 300
- PRI.7: 340

---

```
/* SWITCH REGISTER */

<table>
<thead>
<tr>
<th>Switch</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW15</td>
<td>100000</td>
</tr>
<tr>
<td>SW14</td>
<td>40000</td>
</tr>
<tr>
<td>SW13</td>
<td>20000</td>
</tr>
<tr>
<td>SW12</td>
<td>10000</td>
</tr>
<tr>
<td>SW11</td>
<td>4000</td>
</tr>
<tr>
<td>SW10</td>
<td>2000</td>
</tr>
<tr>
<td>SW9</td>
<td>1000</td>
</tr>
<tr>
<td>SW8</td>
<td>400</td>
</tr>
<tr>
<td>SW7</td>
<td>200</td>
</tr>
<tr>
<td>SW6</td>
<td>100</td>
</tr>
<tr>
<td>SW5</td>
<td>40</td>
</tr>
<tr>
<td>SW4</td>
<td>20</td>
</tr>
<tr>
<td>SW3</td>
<td>10</td>
</tr>
<tr>
<td>SW2</td>
<td>4</td>
</tr>
<tr>
<td>SW1</td>
<td>2</td>
</tr>
<tr>
<td>SW0</td>
<td>1</td>
</tr>
</tbody>
</table>

```

---

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

---

```
;DATA BIT DEFINITIONS (BIT00 TO BIT15)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIT15</td>
<td>100000</td>
</tr>
<tr>
<td>BIT14</td>
<td>40000</td>
</tr>
<tr>
<td>BIT13</td>
<td>20000</td>
</tr>
<tr>
<td>BIT12</td>
<td>10000</td>
</tr>
<tr>
<td>BIT11</td>
<td>4000</td>
</tr>
<tr>
<td>BIT10</td>
<td>2000</td>
</tr>
<tr>
<td>BIT09</td>
<td>1000</td>
</tr>
<tr>
<td>BIT08</td>
<td>400</td>
</tr>
<tr>
<td>BIT07</td>
<td>200</td>
</tr>
<tr>
<td>BIT06</td>
<td>100</td>
</tr>
<tr>
<td>BIT05</td>
<td>40</td>
</tr>
<tr>
<td>BIT04</td>
<td>20</td>
</tr>
<tr>
<td>BIT03</td>
<td>10</td>
</tr>
<tr>
<td>BIT02</td>
<td>4</td>
</tr>
<tr>
<td>BIT01</td>
<td>2</td>
</tr>
<tr>
<td>BIT00</td>
<td>1</td>
</tr>
</tbody>
</table>
```

---

```
.EQUIV BIT09, BIT9
.EQUIV BIT08, BIT8
.EQUIV BIT07, BIT7
```
.EQUIV BIT0, BIT6
.EQUIV BIT0, BIT5
.EQUIV BIT0, BIT4
.EQUIV BIT0, BIT3
.EQUIV BIT0, BIT2
.EQUIV BIT0, BIT1
.EQUIV BIT0, BIT0

; BASIC "CPU" TRAP VECTOR ADDRESSES
ENTRY = 4 ; TIME OUT AND OTHER ERRORS
RESEV = 10 ; RESERVED AND ILLEGAL INSTRUCTIONS
TBITVEC = 14 ; "T" BIT
TNTVEC = 14 ; TRACE TRAP VECTOR
BPTVEC = 14 ; BREAKPOINT TRAP (BPT)
IOTVEC = 20 ; INPUT/OUTPUT TRAP (IOT) ** SCOPE **
PAUSE = 24 ; POWER FAIL
EMTVEC = 30 ; EMULATOR TRAP (EMT) ** ERROR **
TRAPVEC = 34 ; "TRAP" TRAP
TKVEC = 60 ; ITTY KEYBOARD VECTOR
TPVEC = 64 ; ITTY PRINTER VECTOR
PINTVEC = 240 ; PROGRAM INTERRUPT REQUEST VECTOR
ILMEM = 4
ADRS = R1
GOOD = R2
BAD = R3
REGISTER = R1
BIT = R2
Funct = R3
LEAD = R2
FOLLOW = R4
DLADDR = 175610

; THE FOLLOWING DEFINITIONS APPLY TO THE GLOBAL SUBS
SET = -1
CLR = 0

;******************************
; RCSR REGISTER BIT NAMES
;******************************
;UNUSED BIT15
;UNUSED BIT14
;UNUSED BIT13
;UNUSED BIT12
;RCVRACT = BIT11 ; RECEIVER ACTIVE INDICATOR
;UNUSED BIT10
;UNUSED BIT9
;UNUSED BIT8
;RCVRDONE = BIT07 ; RECEIVER DONE
;UNUSED BIT06 ; RECEIVER INTERRUPT ENABLE
;UNUSED BIT05
;UNUSED BIT04
;UNUSED BIT03
;UNUSED BIT02
;UNUSED BIT01
;RDURUN = BIT00 ; READER RUN
;UNUSED BIT0
### BUF REGISTER BIT NAMES

| 784  | 100000 |
| 785  | 040000 |
| 786  | 020000 |
| 787  | 010000 |
| 788  | 000000 |
| 789  | 000010 |
| 790  | 000020 |
| 791  | 000040 |
| 792  | 000000 |
| 793  | 000040 |
| 794  | 000000 |
| 795  | 000000 |
| 796  | 000020 |
| 797  | 000010 |
| 798  | 000000 |
| 799  | 000000 |
| 800  | 000000 |
| 801  | 000000 |
| 802  | 000000 |
| 803  | 000001 |

### TCSR REGISTER BIT NAMES

| 804  | 100000 |
| 805  | 040000 |
| 806  | 020000 |
| 807  | 010000 |
| 808  | 004000 |
| 809  | 000000 |
| 810  | 000000 |
| 811  | 000000 |
| 812  | 000000 |
| 813  | 000000 |
| 814  | 000000 |
| 815  | 000000 |
| 816  | 000000 |
| 817  | 000000 |
| 818  | 000000 |
| 819  | 000000 |
| 820  | 000000 |
| 821  | 000000 |
| 822  | 000000 |
| 823  | 000000 |
| 824  | 000000 |
| 825  | 000000 |
| 826  | 000000 |
| 827  | 000000 |
| 828  | 000000 |
| 829  | 000000 |
| 830  | 000000 |
| 831  | 000000 |
| 832  | 000000 |
| 833  | 000000 |
| 834  | 000000 |
| 835  | 000000 |
| 836  | 000000 |
| 837  | 000000 |
| 838  | 000000 |
| 839  | 000000 |
BASIC DEFINITIONS

840 000000  TDATA5= BIT05
841 000020  TDATA4= BIT04
842 000010  TDATA3= BIT03
843 000004  TDATA2= BIT02
844 000002  TDATA1= BIT01
845 000001  TDATA0= BIT00

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

; FLAG BITS TO BE USE OR CLEARED IN SUSWR.

; DATA = 17
851 000017  EVENODD = 40
852 000020  COMSPD = 100
853 000040  PBR = 200
854 000100  ; BAUDE MUST BE ON THE UPPER
855 002000  ; BYTE BOUNDARY OF SUSWR,-4 BITS
856 007400  BND = 7400
857 010000  BRK = 10000
858 020000  WRAP = 20000
859 040000  MAINT= 40000
860 100000  EPAD= 100000

; ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A "..2,MAL'T"
; SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
; LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
; =174
861 000174  DISPREG = MORD 0
862 000176  SWREG = MORD 0

; SBTTL STARTING ADDRESS(ES)
; JMP @START, JUMP TO STARTING ADDRESS OF PROGRAM
863 000200  000137 001336

.SBTTL TRAP CATCHER
MAINDC 11 OVDVC-C  MACY 11 30A(1052) 12-SEP-84 15:41 PAGE 20

CVQVCC.P11  12-SEP-84 08:35  ACT11 HOOKS

.SBTTL ACT11 HOOKS

**********************************************************************

.HOOKS REQUIRED BY ACT11

;SVPC=. ;SAVE PC

.46  ;1) SET LOC .46 TO ADDRESS OF $ENDAD IN .EOP

.52  ;2) SET LOC .52 TO ZERO

.1000 ; RESTORE PC

=1000

.SBTTL APT PARAMETER BLOCK

************************************************************

;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

**************************************************************

.0X=  ; SAVE CURRENT LOCATION

.24  ; SET POWER FAIL TO POINT TO START OF PROGRAM

.200 ; FOR APT START UP

.44  ; POINT TO APT INDIRECT ADDRESS PTR.

.$APTHDR ; POINT TO APT HEADER BLOCK

.1X  ;RESET LOCATION COUNTER

.Mvc .99

;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC

;INTERFACE SPEC.

.$APTHDR:

.$MBITS: .WORD 0 ; TWO HIGH BITS OF 16 BIT MAILBOX ADDR.

.$MADDR: .WORD 0 ; ADDRESS OF APT MAILBOX (BITS 0-15)

.$TSMT: .WORD 5 ; RUN TIM OF LONGEST TEST

.$RTIME: .WORD 45 ; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

.$UNITM: .WORD 30 ; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

.2 $ETEND-$MAIL/2 ; LENGTH MAILBOX-ETABLE(WORDS)
.SBTTL COMMON TAGS

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

; .CNTAG: .1100
; .STMM: .WORD 0
; .ERFLG: .BYTE 0
; .ICNT: .WORD 0
; .LPMAD: .WORD 0
; .IEVTI: .BYTE 0
; .ITEM: .BYTE 0
; .ERMAX: .BYTE 1
; .ERPPC: .WORD 0
; .GDADR: .WORD 0
; .BDAT: .WORD 0
; .AUTOB: .BYTE 0
; .INTAG: .BYTE 0
; .SNR: .WORD 0
; .DISP: .WORD 0
; .TKS: .WORD 0
; .TPS: .WORD 0
; .TPB: .WORD 0
; .NULL: .BYTE 0
; .FILL: .BYTE 0
; .TPFLG: .BYTE 0
; .TIMES: 0
; .BELL: .ASCIZ "<207><377>"
; .ASCI: /?/
; .CRLF: .ASCIZ "<15>"
; .LF: .ASCIZ "<12>"

; .SBTTL APT MAILBOX-EТАBLE

; .EVEN
; .MAIL: .WORD APT MAILBOX
; .MTY: .WORD AMSGTY
; .TAL: .WORD AFTAL
; .FAL: .WORD ATZFN
; .TESTN: .WORD ATEN
; .APASS: .WORD APASS
; .PASS: .WORD TEF
; .ADECT: .WORD TEEVCT
; .DEVC: .WORD TDEVCT
; .UNIT: .WORD AMUNIT
; .AIM: .WORD AMUNIT
; .MAY: .WORD AMGAD
; .MESSAGE: Address
I2

MAINEDEC-11-DVDVC-C   MACY11 30A(1052)  12-SEP-84  15:41  PAGE 22
CVDVC.C  19-SEP-84  08:55  APT MAILBOX-ETABLE

965 001212 000000  MSGLG: .WORD AMSGL \ MESSAGE LENGTH
966 001214 000000  ETABLE: \ APT ENVIRONMENT TABLE
967 001214 000000  ENV: .BYTE AEENV \ ENVIRONMENT BYTE
968 001215 000000  ENVYM: .BYTE AEENVY \ ENVIRONMENT MODE BITS
969 001216 000000  ASREG: .WORD ASWREG \ APT SWITCH REGISTER
970 001220 011110  USR: .WORD AUSR \ USER SWITCHES
971 001222 000000  CPUOP: .WORD APCPU \ CPU TYPE.OPTIONS
972 001224 000000  BITS 15-11=CPU TYPE
973 001225 000000  \ 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
974 001226 000000  BIT 10=REAL TIME CLOCK
975 001227 000000  BIT 9=FLOATING POINT PROCESSOR
976 001228 000000  BIT 8=MEMORY MANAGEMENT
977 001229 000000  \ HIGH ADDRESS,M.S. BYTE
978 001230 000000  \ HIGH ADDRESS,BLK#1
979 001231 000000  \ MEM.TYPE BYTE -- (HIGH BYTE)
980 001232 000000  \ 900 NSEC CORE=001
981 001233 000000  \ 300 NSEC BIPOLAR=002
982 001234 000000  \ 500 NSEC HDS=003
983 001235 000000  \ HIGH ADDRESS,BLK#1
984 001236 000000  \ MEM.LAST ADDR.+3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
985 001237 000000  \ HIGH ADDRESS,M.S. BYTE
986 001238 000000  \ HIGH ADDRESS,M.S. BYTE
987 001239 000000  \ MEM.TYPE,BLK#2
988 001240 000000  \ MEM.LAST ADDRESS,BLK#2
989 001241 000000  \ HIGH ADDRESS,M.S.BYTE
990 001242 000000  \ HIGH ADDRESS,M.S.BYTE
991 001243 000000  \ MEM.TYPE,BLK#3
992 001244 000000  \ MEM.LAST ADDRESS,BLK#3
993 001245 000000  \ HIGH ADDRESS,M.S.BYTE
994 001246 000000  \ HIGH ADDRESS,M.S.BYTE
995 001247 000000  \ MEM.TYPE,BLK#4
996 001248 000000  \ MEM.LAST ADDRESS,BLK#4
997 001249 000000  \ INTERRUPT VECTOR#1.BUS PRIORITY#1
998 001250 175610  \ INTERRUPT VECTOR#2 BUS PRIORITY#2
999 001251 000000  \ BASE: .WORD ABASE \ BASE ADDRESS OF EQUIPMENT UNDER TEST
1000 001252 000000  \ DEVM: .WORD ADEVM \ DEVICE MAP

.SBTTL ERROR POINTER TABLE

* THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
* THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
* LOCATION ITEM. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
* NOTE1: IF ITEM 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

#ERRTB:

GLOBAL DATA

DLADD: DLADDR
DLVEC: 300
RCSR: DLADDR + 0
RBUFF: DLADDR + 2
TCSR: DLADDR + 4
TCSRHI: DLADDR + 5
TBUFF: DLADDR + 6
I: 0

.BLKWM 20 ; FOR R5 STACK
RSSTACK: .WORD 0
START:

.SBTL INITIALIZE THE COMMON TAGS

.CLEAR THE COMMON TAGS (ICMTAG) AREA

MOV #ICMTAG, R6

FIRST LOCATION TO BE CLEARED

CLR (R6),

CLEAR MEMORY LOCATION

CMP $SR,R6

DONE?

BNE -.6

LOOP BACK IF NO

MOV #STACK, SP

SETUP THE STACK POINTER

.INITIALIZE A FEW VECTORS

MOV @SCAPE, #FTOTVEC

IOT VECTOR FOR SCOPE ROUTINE

MOV #340, #FTOTVEC+2

LEVEL 7

MOV #340, #FINTVEC+2

LEVEL 7

MOV #TRAP, #FTRAPVEC

TRAP VECTOR FOR TRAP CALLS

MOV #TRAP, #FTRAPVEC+2

LEVEL 7

MOV #PHDRN, #FPHRVEC

POWER FAILURE VECTOR

MOV #340, #FPHRVEC+2

LEVEL 7

MOV #HENDCT, #EOCT

SETUP END-OF-PROGRAM COUNTER

CLR #TIMES

INITIALIZE NUMBER OF ITERATIONS

CLR #ESCAPE

CLEAR THE ESCAPE ON ERROR ADDRESS

MOV #1, #EHMAX

ALLOW ONE ERROR PER TEST

MOV #1, #ERRORVEC

SET ERROR VECTOR

SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS

EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.

MOV #ERRORVEC-.SP

SAVE ERROR VECTOR

MOV #641, #ERRORVEC

SET UP ERROR VECTOR

MOV #DSMR, SHR

SETUP FOR A HARDWARE SWICH REGISTER

MOV #DDISP, DISPLAY

AND A HARDWARE DISPLAY REGISTER

CMP 0-1, #DSMR

TRY TO REFERENCE HARDWARE SR

BNE 660

BRANCH IF NO TIMEOUT TRAP OCCURRED

AND THE HARDWARE SR IS NOT = -1

BRANCH IF NO TIMEOUT

SET UP FOR TRAP RETURN

RTI

MOV #SRREG, SHR

POINT TO SOFTWARE SR

MOV #DISPREG, DISPLAY

MOV (SP)+, #ERRORVEC

RESTORE ERROR VECTOR

CLR #PASS

CLEAR PASS COUNT

TEST USER SIZE UNDER APT

YES, USE NON APT SWITCH

TYPE ASCII STRING

GET VALUE FOR SOFTWARE SWITCH REGISTER

TST #042

ARE WE RUNNING UNDER XXDP/ACT?

BNE 700

BRANCH IF YES

CMPB $ENV, #1

ARE WE RUNNING UNDER APT?

BEQ 700

BRANCH IF YES
GET VALUE FOR SOFTWARE SWITCH REGISTER

CMP  SWR,#SWREG \ SOFTWARE SWITCH REG SELECTED?
BNE  715 \ BRANCH IF NO
GET SWR SETTINGS
BR  715
MOV  #1, #AUTOB \ SET AUTO-MODE INDICATOR
177244
715:
800001
112767
001662
1087
1088
1089
1090
1091
001670
001670
001670
001670
001670
026727
001646
001654
001656
001660
001662
001662
001662
001662
001662
177266
000176
000105
104406
000040
000001
177244
681
681
681
681
681
681
681
681
681
681
ASCIZ <CRLF>*MD-11-DVUVC-C<CRLF>
MATMDC-11-DVDC-C  MACY11 30A(1052) 12-SEP-84  15:41  PAGE 26
SNOWC.P11  12-SEP-84  08:55  GET VALUE FOR SOFTWARE SWITCH REGISTER

1092  001712
1093  001712
1094  001712  005767  177334
1095  001716  001101
1096  001720
1097  001762
1098  002050
1099  002116
1100  002120
1101  002120  000674
1102  002122
1103  002122
1104  002122
1105  002130
1106  002130  012767  000001  006760
1107  002136
1108  002136
1109  002136  004767  006524
1110  002136
1111  002142
1112  002142  012167  177106
1113  002146
1114  002146
1115  002146  011167  177104
1116  002152
1117  002152
1118  002152  016701  177076
1119  002156
1120  002156
1121  002156  016767  177072  177074
1122  002164
1123  002164  016767  177064  177070
1124  002172  062767  000002  177062
1125  002200
1126  002200  016767  177050  177056
1127  002206  062767  000004  177050
1128  002214
1129  002214  016767  177034  177044
1130  002222  062767  000005  177036
1131  002230
1132  002230  016767  177020  177032
1133  002236  062767  000006  177024
1134  002244
1135  002244  012705  001334
1136  002250  000005
1137  002250

WHILE #DEV1 EQ #0 DO

TSI  #DEV1
BNE  50001:

TYPITXT <<CRLF>> I HAVE NO DEVICE TO TEST.

TYPITXT <<CRLF>> SET UP #DEV1 TO INDICATE ACTUAL CONFIGURATION.

TYPITXT <<CRLF>> TYPE PROCEED (P) TO CONTINUE.

HALT

B 50000:

ENDDO

50001:

LET INITFLAG := #1
LET BITMASK := #BIT15 ; START AT CONSOLE

LOOP:

CALL CYCLE ; NO ARGUMENTS--ADDRS -> NEXT ADDRESS

JSR PC,CYCLE

; ADDRS+2 -> NEXT VECTOR

LET DLADD := (ADRS)+

; GET UNIT ADDRESS

LET DLADD,RCSR

; GET UNIT VECTOR

LET DLADD,RCRFB

; ADD #2,RCRFB

LET TCSR := DLADD + #4

; DLADD := DLADD + 0

LET TCSR := DLADD + #5

; DLADD := DLADD + #6

LET R5 := #RSSTACK

; RSSTACK

; DSERR

; BRESET
TST1: SCOPE

1143 002252 000004
1144 002254 012767 000002 176676
1145 002256 012767 000001 176710
1146 002272
1147 002270
1148 002270 016701 176760
1149 002270
1150 002274
1151 002274 010146
1152 002276 012701 000004
1153 002302 012721 010636
1154 002306 012711 000340
1155 002312 012601
1156 002314
1157 002314 005067 176752
1158 002320
1159 002320
1160 002320
1161 002320 012767 002326 176562
1162 002324
1163 002324 005067 006332
1164 002326
1165 002326
1166 002326
1167 002332 005711
1168 002332
1169 002332 005767 006324
1170 002340 001401
1171 002342
1172 002342 104001
1173 002344
1174 002344
1175 002344
1176 002344
1177 002344
1178 002344
1179 002352
1180 002352 016701 176676
1181 002356 066701 176710
1182 002362
1183 002362 026727 176704 000010
1184 002370 001353
1185 002372
1186 002372 010146

MOV $2, @TIMES
MOV $1, @TESTN
SET TEST NUMBER IN APT MAIL BOX
LET ADRS := DLADD
SETUP INTERRUPT

MOV @ILLMEM, R1
MOV @INPSVR.(R1), R1
MOV (SP)++, R1
LET I := 00
REPEAT

CLR I

BGNSUB

MOV #644, @LPERR
CLR INTF 'G'
LET INTFLAG := 00

READ FLAG
IF INTFLAG NE 00 THEN
FATAL ERROR
ERRDF 1,,NODL
ENDIF

ENDSUB

LET I := I + 02
LET ADRS := DLADD + I
UNTIL I EQ 08.

MOV R1, -(SP)

CLRVEC ILLMEM
1187 002374 010246
1188 002376 012701 000004
1189 002402 010102
1190 002604 062702 000002
1191 002410 010221
1192 002412 005011
1193 002414 012602
1194 002416 012601
1195
1196 002420

MOV  R2,(SP)
MOV  @ILLMEM,R1
MOV  R1,R2
ADD  @2,R2
MOV  R2,(R1)+
CLR  (R1)
MOV  (SP)+,R2
MOV  (SP)+,R1

;PUSH R2 ON STACK
;POP STACK INTO R2
;POP STACK INTO R1

ENDTST
;END OF TEST
THE FOLLOWING 8 TESTS TEST ALL 'READ WRITE' BITS

TEST 2
BREAK, TCSRA SET, CLEAR, RESET
THE BREAK BIT IS USUALLY USED ON THE CONSOLE DEVICE. IF ADDITIONAL DNL OPTIONS ARE USED
IT IS RECOMMENDED TO REMOVE THE 'BIG JUMPER AND CLEAR BIT 12 IN UNSR WHICH WILL CAUSE THIS TEST TO BE SKIPPED.

TST2:  SCPE  MOV 0104T  TIMES  DD 10 ITERATIONS
       MOV 024T  TESTN  SET TEST NUMBER IN APT MAIL BOX

          BIT  BREAK, UISR
          BEQ  500041
          BIT  APTENV, ENV
          BEQ  500051

          500041:
          EXIT TEST, BREAK NOT INSTALLED
          ENDIF

          500051:
          ; SEE IF IT IS CLEAR
          BANGSUB

          500061:
          ; TRY TO SET BREAK BIT
          BANGSUB

          500071:
          LET TCSR := TCSR SET BY BREAK
MAINDEC-11-DVDVC-C  MACY11 30A(1052)  12-SEP-84  13:41  PAGE 30
CVDVC.C111  12-SEP-84  08:55  T2  BREAK - TCSR0 SET, CLEAR, RESET

1253  002534  500071:
1254  002534  ENDSUB
1255  002534
1256  002534  : TRY TO CLEAR A SET BIT
1257  002534  BGNSUB
1258  002534  MOV  @664,$ILPERR
1259  012767  002542  176346  LET  BCSR := BCSR CLR.BY BREAK
1260  002542  042777  000001  176514  IF  @BREAK SET IN BCSR THEN
1261  002542  BIT  @BREAK,BCSR
1262  002550  032777  000001  176506  ; BREAK DID NOT CLEAR IN TCSR
1263  002550  001401  ERRNO 4, DIDNOT
1264  002556  500101  ENDIF
1265  002560  500101:
1266  002560  ENDSUB
1267  002560  : NOW SEE IF RESET CLEARS IT
1268  002560  BGNSUB
1269  002560  ERROR 4
1270  002562  500101:
1271  002562  ENDSUB
1272  002562  ERROR 5
1273  002562  ENDSTST
1274  002562 
1275  002562  MOV  @671,$ILPERR
1276  012767  002570  176320  LET  BCSR := BCSR SET.BY BREAK
1277  002570  052777  000001  176466  ISSUE BUS RESET
1278  002570  BIS  @BREAK,BCSR
1279  002576  RESET
1280  002576  000005  IF  @BREAK SET IN BCSR THEN
1281  002576  BIT  @BREAK,BCSR
1282  002600  032777  000001  176456  ; BREAK DID NOT RESET IN TCSR
1283  002600  001401  ERRNO 5, DIDNOT
1284  002606  500111:
1285  002606  ENDIF
1286  002606  ENDSUB
1287  002606  ENDSTST
1288  002612
1289  002612
1290  002612
1291  002612
1292  002612
1293  002612
E3

MAINDCC: DODVC-C
MACY1 30A(1052) 12-SEP-84 15:41 PAGE 31
CVDCC.P11 12-SEP-84 08:55

BREAK - TCSR0 SET, CLEAR, RESET

TEST 3: MAINT - TCSR2 SET, CLEAR, RESET

TST3: SCOPE

1294 002612 000004
1295 002614 012767 000010 176336
1300 002622 012767 000003 176350

MOV #10, TIMES
1296 1297 ; DO 10 ITERATIONS
1301
1302
1303 002630 032767 040000 176362
1304 002636 001404
1305 002640 126727 006261 000001
1306 002646 001004
1307 002650
1308 002650
1309 002650 012767 000001 176302
1310 002656 000452
1311 002660
1312 002660

1313
1314
1315 002660
1316 002660 012767 002666 176222
1317
1318
1319 002666 032777 000004 176370
1320 002674 001401
1321
1322 002676
1323 002676 104006
1324 002700
1325 002700
1326 002700
1327
1328
1329 002700
1330 002700 012767 002706 176202
1331
1332 002706 052777 000004 176350
1333
1334 002714
1335 002714 032777 000004 176342
1336 002722 001001
1337
1338 002724
1339 002724 104007
1340 002726
1341 002726
1342 002726
1343
1344
1345 002726
1346 002726 012767 002734 176154
1347
1348 002734
1349 002734 042777 000004 176322

IF @MAINTJUMP NOTSETIN @USWR ORB CONSOLE EQ @TRUE

TST4: EXIT TEST

EXIT THIS TEST

50013:

50012:

50012:

50013:

50014:

50014:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:

50015:
MAINDOC-11-OVDVC-C  MACY113OA(1052) 12-SEP-84 15:41 PAGE 32

CVDVC_CP11  12-SEP-84  08:55  T3  MAINT - TCSR2 SET, CLEAR, RESET

1350  002754
1351  002742  002742  032777  000004  176314  BIT  #MAINT, #TCSR
1353  002750  001401  BEQ  500164
1354
1355  002752
1356  002752  104010  ERROR  10
1358  002754
1359  002754  500164:
1360
1361
1362  002754
1363  002754  012767  002762  176126  MOV  #678, #LPERR
1364
1365  002762
1366  002762  052777  000004  176274  BIS  #MAINT, #TCSR
1367
1368  002770
1369  002770  000005  RESET
1370  002772
1371  002772  032777  000004  176264  BIT  #MAINT, #TCSR
1372  003000  001401  BEQ  500170
1373
1374  003002
1375  003002  104011  ERROR  11
1376  003004
1377  003004
1378  003004
1379  003004
1380
1381
1382
MAINDEC-11-DYDVC-C   MACYII 30A(1052)  12-SEP-84 15:41 PAGE 33

**T3** MAINT - TCS2 SET, CLEAR, RESET

1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438

**TST4:** SCOPE

```
1387: MOV 003004 000004
           012767 000010 176144
1388: MOV 003006 012767 000004 176156
1389: MOV @PR7,-(SP) ; PUT NEW PS ON STACK
1390: MOV @PR7,-(SP) ; PUT NEW PC ON STACK
1391: MOV 003032 000002
1392: RTI
644:
```

**TEST 4** XMITIE - TCSR SET, CLEAR, RESET

```
; SEE IF IT IS CLEAR
BGNSUB

dmv 0654,1LPErr

if xmitie setin @tcsr then

bit xmitie,@tcsr

beq 50020:

xmitie did not reset in tcsr
errhard 12,,didnot
endif

endsub

; TRY TO SET XMITIE BIT
BGNSUB

mov 0664,1LPErr

let @tcsr := @tcsr set.by xmitie

bis xmitie,@tcsr

if stuck to 0

xmitie notsetin @tcsr then

bit xmitie,@tcsr

bne 50021:

xmit did not reset in tcsr
errhard 13,,didnot
endif

endsub

; TRY TO CLEAR A SET BIT
BGNSUB

mov 0674,1LPErr

let @tcsr := @tcsr clr.by xmitie

bic xmitie,@tcsr

if should have cleared

xmit setin @tcsr then

bit xmitie,@tcsr

beq 50022:

xmit did not clear in tcsr
errhard 14,,didnot
endif
```

ENDIF

ERROR 13

ENDIF

ERROR 14

ENDIF
1439 003130  
1440 003130  
1441 003130  
1442  
1443  
1444 003130 012767 003136 175752  
1445 003144 052777 000100 176120  
1446 003136  
1447 003136  
1448 003136 052777 000100 176110  
1449 003144 000005  
1450 003144  
1451 003146 0001401  
1452 003146  
1453 003146 032777 001000 176100  
1454 003154  
1455  
1456 003156  
1457 003156 104015  
1458 003160  
1459 003160  
1460 003160  
1461 003160  
1462  
1463  
1464  

500221:  
ENDIF  
ENDSUB  

; NOW SEE IF RESET CLEARS IT  
BGNSUB  

LET @TCSR := @TCSR SET, @XMITIE  
ISSUE BUS RESET  
RESET  
IF @XMITIE SET IN @TCSR THEN  
ERROR 15

500231:  
ENDIF  
ENDSUB  
ENDSYST
; TEST 5: RCVRIE - RCSR Set, Clear, Reset

; TST5: SCOPE

003160 000004
003162 012767 000010 175770
003170 012767 000005 176002
003176 012767 003204 175704
003204 032777 000100 176046
003212 001401
003214 104035

MOV $10, TIMES 10 10 ITERATIONS
MOV $5, TIMES 10 SET TEST NUMBER IN APT MAIL BOX
; SEE IF IT IS CLEAR
BGNSUB

003204
003204 032777 000100 176046
BIT RCVRIE, RCSR
BEQ 500241

; RCVRIE DID NOT RESET IN RCSR ERR# 35, DIDNOT
ENDIF
500241:

; TRY TO SET RCVRIE BIT
BGNSUB

003216
003216 012767 003224 175664
MOV $651, ILPErr
LET RCSR := RCSR SET.BY RCVRIE

003224 052777 000100 176026
BIS RCVRIE, RCSR

003232
003232 032777 000100 176020
BIT RCVRIE, RCSR
BNE 500251

; RCVRIE DID NOT SET IN RCSR ERR# 36, DID NOT
ENDIF
500251:

; TRY TO CLEAR A SET BIT
BGNSUB

003244
003244 012767 003252 175636
MOV $661, ILPErr
LET RCSR := RCSR CLR.BY RCVRIE

003244
003244 012767 003252 175636
MOV $661, ILPErr
LET RCSR := RCSR CLR.BY RCVRIE

003252
003252 042777 000100 176000
BIC RCVRIE, RCSR

003260
003260 032777 000100 175772
BIT RCVRIE, RCSR
BEQ 500261

; RCVRIE DID NOT CLEAR IN RCSR ERR# 37, DID NOT
ENDIF
500261:

; NOW SEE IF RESET CLEARS IT
BGNSUB
MOV #67H, @LPERR

LEA @ACRSR := @ACRSR SET.BY @ACVRIE

LET @ACRSR := @ACRSR SET.BY @ACVRIE

ISSUE BUS RESET

IF @ACVRIE SET IN @ACRSR THEN

: RCVRIE DID NOT RESET IN @ACSR

ERROR 40

ENDIF

ENDIF

ERRMARD 40., DID NOT

ENDSUB

ENDTST

50027H:
; THE FOLLOWING 4 TESTS VERIFY
; THAT RESET (INIT) Initializes READ ONLY BITS.
;*****************************************************************************
;*****************************************************************************
; TEST 6  TEST THAT RCVRDONE - RCSR 7 - IS CLEARED BY INIT
;*****************************************************************************

TST6:  SCOPE

BGNSUB

N 003322 000004
N 003324 012767 000010 175626
N 003332 012767 000006 175640

N 003332
N 012767
N 000006
N 175640

N 003340
N 003340 012767 003346 175542
N 003346
N 032777 000200 175704
N 003354 001402

MDV #10,1TIMES ;DO 10 ITERATIONS
MDV #6,1TESTN ;SET TEST NUMBER IN APT MAIL BOX

;DO 10 ITERATIONS

MOV #641,1LPERR
BIS #RCVRDONE,1RCSR
BEQ 50030:

; RCVRDONE SHOULD HAVE CLEARED BY INIT
; RCVRDONE DID NOT CLEAR IN RCSR
; ERRAH 41, MRESET, DID NOT

; REISSUE RESET
BRESER

ENDIF

; ALLOW LOOPING AFTER ERROR
CLOOP
ENDSUB
ENDTST

RESET

50030:

ERROR 41
; TEST THAT RCVDONE - RCSR 7 - IS CLEARED BY INIT

; TEST 7: TEST THAT RCVRACK - RCSR 11 - IS CLEARED BY INIT

; Do 10 iterations
; Set test number in map mail box

; If B console eq @TRUE then
; Execute test
; Else
; If @WRAP set in USHR then
; Execute test
; Else
; If @MAINT set in USHR then
; Execute test
; Else
; Exit test: line must be terminated

; Exit this test

; If @RCVRACK set in BRCSR then
; Reset should have cleared RCVRACK
; Let @BCSR := @BCSR CLR. BY @MAINT
; Error 44, MReset, DidNot
; Testing effect of reset on bit
N3

TST10: SCOPE

*TEST 10  TEST THAT XMITRDY - TCSR 7 - IS SET BY INIT

1648

1649

1650

1651

1652

1653

1654

1655

1656

1657

1658

1659

1660

1661

1662

1663

1664

1665

1666

1667

1668

1669

1670

1671

1672

1673

1674

1675

1676

1677

1678

1679

1680

1681

MOV @010, @THERM
MOV @010, @TESTN
SET TEST NUMBER IN APT MAIL BOX

MOV @0641, @LPERR

IF XMITRDY NOTSETIN @TCSR THEN

;RESET SHOULD HAVE SET BIT.
;XMITRDY DID NOT SET IN TCSR (AFTER RESE
ERRHAND 42,RESET,DIDN'T

;ISSUE ANOTHER RESET
RESET

ENDIF

;ALLOW LOOPING ON ERROR

ENDSUB

ENDTST

17

TEST THAT RCVRACT - RCSR 11 - IS CLEARED BY INIT

MOV @010, @THERM
MOV @010, @TESTN
SET TEST NUMBER IN APT MAIL BOX

MOV @0641, @LPERR

IF XMITRDY NOTSETIN @TCSR THEN

;RESET SHOULD HAVE SET BIT.
;XMITRDY DID NOT SET IN TCSR (AFTER RESE
ERRHAND 42,RESET,DIDN'T

;ISSUE ANOTHER RESET
RESET

ENDIF

;ALLOW LOOPING ON ERROR

ENDSUB

ENDTST
**MAINDDEC-11-DVDVC-C**

**MACY11 30A(1052) 12-SEP-84 15:41 PAGE 41**

**CV0VC.C.P11 12-SEP-84 08:55**

**T100** TEST THAT XMITRDY - TCSR 7 - IS SET BY INIT

```
1682
1683
1684
1685
1686
1687
1688 003336 000004
1689 003340 012767 000001 175412
1690 003354 012767 000011 175424
1691
1692 003354
1693 003354 126727 005345 000001
1694 003362 001404
1695 003364 052767 000001 175422
1696 003372 001404
1697 003374
1698 003374
1699 003374 012767 000001 175356
1700 003362 000513
1701 003364
1702 003364
1703
1704 003364
1705 003364 012767 000001 000212
1706 003364
1707 003364
1708
1709 003364
1710 003364 012767 000000 000206
1711 003364
1712 003364
1713
1714
1715
1716
1717
1718
1719 003362
1720 003362 105077 175436
1721
1722
1723
1724
1725 003362
1726 003362 010546
1727 003364 012745 177777
1728 003364 012745 175420
1729 003364 012745 000200
1730 003354 012745 000500
1731 003354 004767 004440
1732
1733
1734 003362
1735 003362 103001
1736
1737 003364
```

Note: The text appears to be a program listing in assembly language, detailing tests and operations related to the initialization of certain registers and variables.
MAINDEC-11 DVDC-C MACTJ 50A(1052) 12-SEP-84 15:41 PAGE 42
CVDCC.P11 12-SEP-84 08:55 T11 TEST THAT XMIT RDY TCSR 7 CLEARS

1738 003666 104066
1739 003666
1740 003666
1741 500461:

ENDIF

1742

1743

LOAD TBUF WITH A SECOND CHARACTER

1744

; CHECK IMMEDIATELY THAT XMIT RDY IS CLEAR

1745

; AND THEN WAIT FOR IT TO SET

1746

1747 003666
1748 003666 105077 175376
1749 003672 000240
1750

CLRD @TBUF NOP

1751

GIVE IT TIME TO CLEAR

1752 003674
1753 003674 032777 000200 175362
1754 003702 001404
1755

BIT @XMITRDY,TCSR

1756 003704
1757 003704 012767 177777 000114
1758

MOV @SET,ERRORFLAG

1759

SEND SECOND CHARACTER

1760 003712
1761 003712 000416
1762 003714
1763

BR 500471

1764

ELSE

1765

1766 003714
1767 003714 010546
1768 003716 012745 177777
1769 003722 016745 175336
1770 003726 012745 000200
1771 003732 012745 000500
1772 003736 004787 004356
1773 003742 012605
1774 003744
1775 003744 010001
1776

MOV RS,-(SP)

1777 003746
1778 003746 104070
1779

ERROR 70

1780

ENDIF

1781

1782 003750
1783 003750
1784 003750 026727 000052 177777
1785 003756 001011
1786

500511:

1787 003760
1788 003760 026727 000040 000001
1789 003766 003404
1790

CMP ERRORFLAG,@SET

1791 003770
1792 003770 0104067
1793 003772 012767 177777 000030
1794

ERROR 67

1795

MOV @SET,EXITFLAG

1796

ENDIF

1797

1798

CALL TIMER IN <500,0XMITRDY,TCSR,@SET>

1799

IF ERROR THEN

1800

XMIT RDY DID NOT SET IN TCSR

1801

ERROR 70, DIDNOT

1802

ENDIF

1803

1804

1805 003756
1806

ENDIF; OF DEFERRED ERROR CALL

1807

1808

IF ERRORFLAG EQ @SET THEN

1809

CALL ERROR IF 2ND TRY

1810

ERROR 67, DIDNOT

1811

LET EXITFLAG := @SET
ENDIF

#elif

ELSE

; NO ERROR

LET EXITFLAG := #SET

ENDIF

ENDIF

EXITFLAG EQ #SET

EXIT

EXIT / SKIP AROUND FLAG WORDS

EXIT / EXIT THIS TEST

ENDTST
E4

PAGE 44

1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870

TST 12: SCOPE

004032  000004
004034  012767  000001  175116
004042  012767  00012   175130
004050
004050  032767  040000  175142
004056  001404
004060  126727  005041  000001
004066  001004
500541:
004070
004070
004070  012767  000001  175062
004076  000442
004100
500551:
004100
004100  052777  000004  175156
004114
004114  105077  175150
004120  010546
004122  012745  177777
004126  016745  175126
004132  012745  000200
004136  012745  000500
004142  004767  004152
004146  012605
004150
004150  103004
004152
004152  042777  000004  175104
004156
004160
1040"1
004162
500561:

MDV #01,#TIMES 1:DO 1  ITERATION
IF #MAINTJUMP NOTSET IN  #USMR ORB Console EQ #TRUE
EXIT TEST
SET TEST NUMBER IN APT MAIL BOX

MDV #12,#TEST:N

BR TST13

MDV #01,#TIMES

BIS #MAINT,#BTCSR

BGNSUB

SEND A CHARACTER AND LET IT WRAP AROUND

LET #TBUF :B= #0

SEND A CHARACTER AND LET IT WRAP AROUND

CALL TIMER IN <#500,#RCVRDONE,#CRSR,#SET>

; DIDN'T SET IN TIME

IF ERROR THEN

; RCVRDONE DID NOT SET IN RCSR

LET #BTCSR := #BTCSR CLR.BY #MAINT

ERROR 71

ENDIF
1871          ENDSUB
1872 004162
1873
1874 004162
1875 004162 012767 004170 174720        MOV     #651,0,LPERR
1876        ; NOW THAT IT IS SET SEE IF IT CAN BE RESET
1877        ; THIS ALSO WILL CLEAR THE MAINT. BIT
1878 004170
1879 004170 000005                  RESET
1880
1881 004172
1882 004172 032777 000200 175060        BIT    #RCVDONE,0,BCSR
1883 004200 001401                BEQ     500571
1884
1885 004202
1886 004202 104072              ERROR  72
1887 004204
1888 004204                500571:
1889 004204
1890 004204                  ENDSUB
1891
1892 ENDTST
1891 004204 000004
1892 004206 012767 000010 174744
1893 004214 012767 000013 174756
1894 MOV @10, #TIMES
1895 MOV @13, #TESTN
1896 ;DO 10 ITERATIONS
1897 ;SET TEST NUMBER IN APT MAIL BOX
1898 TST13: SCOPE
1899 BIT @MAINT,JUMP, #USER
1900 BEQ 500601
1901 JMP 500601
1902 CMP CONSOLE, #TRUE
1903 BNE 500610
1904 EXIT TEST
1905
1906 004242 012767 000001 174710
1907 MOV @1, #TIMES
1908 BR TST14
1909 ;EXIT THIS TEST
1910 ENDIF
1911
1912 004252 052777 000004 175004
1913 BIS @MAINT, #BCSR
1914 BGSUB
1915 MOV @24.4, #PERR
1916 ;OUTPUT A CHARACTER WITH MAINTENANCE
1917 ;SET, AND WAIT FOR XMITRTY TO SET.
1918
1919 004266
1920 CLRB @TBUF
1921 004266 105077 174776
1922 ;WAIT MAXIMUM OF 500 MSEC
1923 ;FOR RCVRDONE TO SET IN
1924 ;RCSR
1925 CALL TIMER IN <5000, #RCVRDONE, #RCSR, #SET>
1926 004272 010946
1927 004274 012745 177777
1928 004290 016745 174754
1929 004304 012745 000200
1930 004310 012745 000500
1931 JSR PC,TIMER
1932 MOV (SP)+, R5
1933 004320 12605
1934 BIC @MAINT, #BCSR
1935 LET BCSR := BCSR CLR.BY @MAINT
1936 ;DID IT BECOME READY?
1937 IF .ERROR THEN
1938 
1939 RCSR DID NOT SET IN RCSR
1940 ERROR 73
1941 ENDF
1942 ENDSUB
1943
1944 004330
1945 004350 103001
1946 004362 042777 000004 174734
1947 LET BCSR := BCSR CLR.BY @MAINT
1948 ;DID IT BECOME READY?
1949 IF .ERROR THEN
1950 RCSR DID NOT SET IN RCSR
1951 ERROR 73
1952 ENDF
1953 ENDSUB
1954
; READ BUFFER
LET RO ; B = BRBUF

IF @RCVRDONE SET IN @RCSR THEN

; RCVRDONE DID NOT CLEAR IN RCSR
ERRHAD 74 .DIDNOT

ENDIF

ENDTST
**MAINDEC-11-OVDCC-C**  
**MACY11 30A(1052) 12-SEP-84 15:41 PAGE 48**  
**CVODCC.P11 12-SEP-84 06:55**

**T13** TEST THAT RCVRDONE IS CLEARED BY READING RBUF

---

**TST14:** SCOPE

**1961**
**1962**
**1963**
**1964**
**1965**
**1966**
**1967** 04352 000004
**1968** 04354 012767 000010 174576
**1969** 04362 012767 000014 174610
**1970** 04370
**1971** 04370 126727 004531 000001
**1972** 04376 001404
**1973** 04400 032767 040000 174612
**1974** 04406 001004
**1975** 04410
**1976** 04410
**1977** 04410 012767 000001 174542
**1978** 04416 000526
**1979** 04420
**1980** 04420
**1981** 04420
**1982** 04420 032767 000001 174566
**1983** 04426 001404
**1984** 04430
**1985** 04430 012767 000001 174522
**1986** 04436 000516
**1987** 04440
**1988** 04440
**1989**
**1990**
**1991**
**1992** 04440 052777 000004 174616
**1993** 04440
**1994** 04446
**1995** 04446 012700 000000
**1996** 04452
**1997** 04452 005001
**1998**
**1999**
**2000**
**2001**
**2002** 04454
**2003** 04454 105077 174610
**2004** 04460
**2005** 04460
**2006** 04460
**2007** 04460 032777 004000 174572
**2008** 04466 001403
**2009** 04470
**2010** 04470 012700 177777
**2011** 04474
**2012** 04474 000401
**2013** 04474
**2014** 04474
**2015** 04476 005201
**2016** 044500

---

**MOV @10.1 TIMES 1:00 10 ITERATIONS**

**MOV @14.1 TESTN 1:SET TEST NUMBER IN APT MAIL BOX**

**CMPB CONSOLE. @TRUE**

**BEG 500641**

**BIT @MAINTJUMP.1USWR**

**BNE 500651**

**500641:**

**EXIT TEST**

**BR TST15 1:EXIT THIS TEST**

**ENDIF**

**500651:**

**IF @APTENV SETIN $ENV THEN**

**LET @BTCSR := @BTCSR SET.BY @MAINT**

**LET RO := @CLR**

**LET R1 := #0**

**LET BTBUF := B := #0**

**REPEAT**

**IF @RCVRACT SETIN @BCSR THEN**

**LET RO := @SET**

**ELSE**

**LET R1 := R1 + #1**

**ENDIF**
UNTIL RO EQ #SET OR R1 HI MAX

IF R1 HI MAX THEN

; IT NEVER SET
; RCVRACT DID NOT SET IN RCSR.
; CANNOT LEAVE WITH MAINT SET

LET @TCSR := @TCSR CLR.BY @MAINT

ERRPR 75, DNDMT
EXIT TEST

EXIT THIS TEST
ENDIF

CHECK FOR TIMING OF RCVRACT. CLEARING
VS RCVRDONE SETTING

WHILE @RCVRACT SETIN @RCSR DO

IF @RCVRDONE SETIN @RCSR THEN

; RCVRDONE AND RCVRACT
; BOTH SET
; CAN NOT LEAVE WITH MAINT SET

LET @TCSR := @TCSR CLR.BY @MAINT

ERRPR 76, DONEACT

EXIT TEST

NO USE CONTINUING
ENDIF

EXIT THIS TEST
ENDIF

ENDDO
2073  004614  500751:
2074
2075
2076  004614  032777  000200  174436
2077  004614
2078  004622  001010
2079
2080
2081  004624
2082  004624  042777  000004  174432
2083  004632
2084  004632  104077
2085  004634
2086  004634  012767  000001  174316
2087  004642  000414
2088  004644
2089  004644  501001:
2090
2091
2092
2093
2094
2095  004644
2096  004644  017700  174412
2097
2098  004650
2099  004650  032777  000200  174402
2100  004656  001404
2101
2102
2103
2104  004660
2105  004660  042777  000004  174376
2106  004666
2107  004666  104100
2108  004670
2109
2110  004670
2111  004670  000401
2112  004672  070000
2113
2114  004674
2115

;RCVRACT = 0 NOW.
; IF RCVRDONE NOTSET IN BCSR THEN
BIT @RCVRDONE,BCSR
BNE 501001

;RCVRDONE DID NOT SET IN RCSR
; CAN NOT LEAVE WITH MAINT SET
LET @BCSR := @BCSR CLR.BY @MAINT

ERROR 77,.DIDNOT
EXIT TEST

;EXIT THIS TEST
ENDIF

;TEST THAT READING THE RECEIVER
;BUFFER CLEARS RCVRDONE

;READ CHAR.
LET RO := @BRBUF

;RCVRDONE SET IN BCSR THEN
IF @RCVRDONE SET IN BCSR THEN
BIT @RCVRDONE,BCSR
BEQ 501011

;RCVRDONE DID NOT CLEAR IN RCSR
; CAN NOT LEAVE WITH MAINT SET
LET @BCSR := @BCSR CLR.BY @MAINT

ERROR 100,.DIDNOT
ENDIF

EXIT
BR TST15
;EXIT THIS TEST
ENDTST
**MAINDEC.A-DVVC.C**

**MACY11: 30A(0152) 12-SEP-84 15:41 PAGE 51**

**CVDCC.P11 12-SEP-84 08:55**

**T14 TEST THAT RCVRACT - RCSR 11 - SETS**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2116</td>
<td>004674</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2117</td>
<td>004676</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2118</td>
<td>004674</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2119</td>
<td>004704</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2120</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2121</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2122</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2123</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2124</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2125</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2126</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2127</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2128</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2129</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2130</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2131</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2132</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2133</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2134</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2135</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2136</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2137</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2138</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2139</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2140</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2141</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2142</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2143</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2144</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2145</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2146</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2147</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2148</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2149</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2150</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2151</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2152</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2153</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2154</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2155</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2156</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2157</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2158</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2159</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2160</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2161</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2162</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2163</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2164</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2165</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2166</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2167</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2168</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2169</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2170</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
<tr>
<td>2171</td>
<td>004677</td>
<td><strong>SCPE</strong></td>
</tr>
</tbody>
</table>

**CODE**

**TST15: SCPE**

**TEST 15 TEST THE OVERRUN BIT - ABUF 14**

**DO 10 ITERATIONS**

**SET TEST NUMBER IN APT MAIL BOX**

**IF @ERABITS NOTSETIN @USR OR CONSOLE EQ @TRUE**

**BIT @ERABITS, @USR**

**BEQ 50102#**

**CMPB CONSOLE, @TRUE**

**BNE 50103#**

**50102#: EXIT TEST**

**MOV @1, @TIMES**

**BR TST16**

**EXIT THIS TEST**

**50103#: EXIT TEST**

**IF @MAINTJUMP NOTSETIN @USR THEN**

**EXIT TEST**

**50104#: EXIT TEST**

**LET @1CSR := @TCSR SET BY @MAINT**

**BIS @MAINT, @TCSR**

**LET @1CSR := @TCSR SET BY @MAINT**

**BGSUB**

**MOV @644, @LPER**

**OUTPUT 2 CHARACTERS WITH**

**AMPLE DELAYS BETWEEN FOR RECEPTION.**

**THIS SHOULD AN CAUSE OVERRUN ERROR.**

**OUTPUT 1 CHARACTER**

**LET @TBUF :B= @0**

**GO AWAY FOR 200 M SEC**

**WAITMS 200.**

**OUTPUT 2ND CHARACTER**

**LET @TBUF :B= @0**

**LET OVERRUN HAPPEN**

**WAITMS 200.**
; READ BUFFER AND ERROR BITS
LET R4 := @RBUF

; IT DIDN'T SET
IF @ORERR NOTSETIN R4 THEN

; ORERR DID NOT SET IN RBUF
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY @MAINT
ERRNAD 101,,DIDNOT

; NO USE COMBundiNG ERRORS
EXIT TST

; NOW SEE IF ERROR BIT SET WITH OVERRUN ERROR:
BGNSUB

; ERROR DID NOT SET IN RBUF
; CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR.BY @MAINT
ERRNAD 102,,DIDNOT

; WHEN ORERR SET:
; GET OUT NOW.
EXIT TST

; CHECK REAL RBUF TO SEE IF ORERR IS STILL SET.

IF @ORERR NOTSETIN @RBUF THEN
MAINEC-11 DVDVC-C MACY11 50A(1052) 12-SEP-84 15:41 PAGE 53

CVDVCC P11 12-SEP-84 08:55 T15 TEST THE OVERRNN BIT - RBUF 14

2226 005140 001010
2227
2228 005142
2229
2230 005142 042777 000004 174114
2231 BIC @MAINT,BTCSR
2232
2233 005150
2234 005150 104103
2235 ERROR 103
2236
2237 ERRHBD 103,ITCLRD
2238
2239 MOV @1,@TIMES
2240 BR TST16
2241 005162
2242 EXIT THIS TEST
2243 ENDIF
2244 ENDSUB
2245
2246 MOV @674,ILPERR
2247 ;NOW SEE IF THEY CLEAR WHEN ANOTHER CHAR. IS RECEIVED
2248
2249 005170
2250 005170 105077 174074
2251 CLRB @TBUF
2252
2253 005174
2254 005174 010546
2255 MOV @R5,-(SP)
2256 MOV @R5,0-(R5)
2257 JSR PC,WAIT
2258 MOV (SP)+,R5
2259
2260 005202 004767 003370
2261 005206 012605
2262
2263 005210
2264 005210 032777 040000 174044
2265 BIT @ORERR,BRBUF
2266 BEQ 501104
2267
2268 005216 001410
2269
2270 005216
2271 005216 012767 000001 173722
2272 MOV @1,@TIMES
2273 BR TST16
2274 005240
2275 501104:

;READE NG RBUF CLEARED ORERR.
;C AN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR,BY @MAINT

;SKIP REST OF TEST
EXIT TST

;SEND A CHARACTER AROUND.
LET @TBUF := @O

;LET IT CIRCULATE
WAITMS 200.

;IF @ORERR SET IN BRBUF THEN

;ORERR DID NOT CLEAR IN RBUF
;C AN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR,BY @MAINT

;AFTER RECEIVING ANOTHER CHAR
;SKIP AROUND REST
EXIT TST

;EXIT THIS TEST
ENDIF
IF @ERROR.SETIN(ERRBUF THEN
ERROR.DID.NOT.CLEAR.IN.ERRBUF
; CANNOT LEAVE WITH MAINT SET
LET BICSR := BICSR CLR.BY @MAINT
ERROR 105, DID NOT
ENDIF

50111:
ENDSUB
ENDTST
C5

PAGE 55

005260 000004
005262 012767 000010 173670
005270 012767 000016 173702

MOV #010, #TIMES 11 DO 10 ITERATIONS
MOV #016, #TESTN 11 SET TEST NUMBER IN APT MAIL BOX

TST16:  

IF #OPBR NOTSETIN #USWR OR #MAINTJMP NOTSETIN #U

BIT #OPBR, #USWR
BEQ 501121
BIT #MAINTJMP, #USWR
BNE 501131

501121:
EXIT TEST

501131:

IFB #APTENV SETIN #ENV THEN

BIT #APTENV, #ENV
BEQ 501141

EXIT TST

501141:

LET ERRCHK := #0 1 CLEAR ERROR WORD

LET OLD := #0-1
LET OLD+2 := #1
LET BTCSR := #BTCSR SET, BY #MAINT

11 EACH BAUD RATE
INCR R3 FROM #0 TO #13. BY #1

LET RO := #BRBUF

11 CHANGE BAUD RATE
LET #BTCSR# := #RATES(R3)
FLAG
LET BIT := 00

; OUTPUT THE CHARACTER
LET BTBUF := 00

; INITIALIZE COUNTER
LET NEW := 00
LET NEW+2 := 00
WHILE BIT EQ 00 DO

IF ORCVRDONE SETIN BCRSR THEN

; DONE - ITS READY
LET BIT := 01
ELSE

; OTHERWISE - INCREMENT TIME
LET NEW := NEW + 01
LET NEW+2 := NEW+2 + CARRY
ENDIF

; SIGNALS DONE
ENDDO

IF NEW+2 LO OLD+2 THEN

; OK
ELSE

; NEW+2 = OLD+2
IF NEW+2 EQ OLD+2 AND NEW LO OLD THEN

; OK
ELSE

; NEW+2 > OLD+2 OR
; (NEW+2 . GT. OLD+2 AND
2403 005330 012767 000004 002434 MOV @BIT2,ERRCHK
2405 005330 S01276:
2407 005336 50125:
2409 005336 *01256:
2410 005336 ENDIF
2411 005336 ENDF
2412 005336 UPDATE OLD TIME
2413 005336 LET OLD := NEW
2414 005344 001004 MOV NEW,OLD
2415 005344 MOV NEW2,OLD2
2416 005344 ENDINC ;BAUD RATE
2417 005352 000712 BR S01161:
2418 005352 S01117:
2419 005354 173441 MOV R3,USR1.R3
2420 005354 MOV R3,-(SP)
2421 005354 BICH @17,(SP)
2422 005366 142603 BICH (SP),R3
2423 005366 LET R3 := R3 CLR.BY @177400
2424 005374 @177400.R3
2425 005374 LET @TCSR1 :R := RATES(R3) ; LIKE ME WANTED IT
2426 005374 ; CAN NOT LEAVE WITH MAINT SET
2427 005374 LET @TCSR := @TCSR1 CLR.BY @MAINT
2428 005374 IF @BIT2 SET IN ERRCHK THEN
2429 005374 REPORT DEFERRED ERROR ERRHAD 126
2430 005374 ENDF
2431 005374 ERROR 126
2432 005620 104126 BR TST17 ;EXIT ;SKIP TABLE
2433 005622 S01301:
2434 005622 004141 EXIT ;EXIT THIS TEST
2435 005624 RATES: ;A TABLE OF THE ACTUAL BYTES TO MOVE INTO THE
2436 005624 ;UPPER BYTE OF XCSR FOR EACH BAUD RATE
2437 005624 ;** NOTE:** THE VALUE INDICATED IN THE COLUMN 'OFFSET
2438 005624 ;INTO TABLE' CAN BE PLACED INTO BITS<11:0>
2439 005624 ;** OF LOCATION 'USR1' TO CAUSE THE CORRESPONDING
2440 005624 ;BAUD TO BE SELECTED IN THE DLV11-F UPON
2441 005624 ;** COMPLETION OF THIS TEST.
2442 005624 RO050: .BYTE 010
2443 005624 RO070: .BYTE 050
2444 005624 RO110: .BYTE 050
2445 005624 RO135: .BYTE 070
2446 005624 RO150: .BYTE 110
2447 005624 RO300: .BYTE 130
2448 005624 RO050: 50 0
2449 005624 RO070: 70 1
2450 005624 RO110: 110 2
2451 005624 RO135: 135 3
2452 005624 RO150: 150 4
2453 005624 RO300: 300 5
<table>
<thead>
<tr>
<th>ADDR</th>
<th>VALUE</th>
<th>BAUD RATE</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>2459</td>
<td>005632</td>
<td>150</td>
<td>R0600: .BYTE 150</td>
</tr>
<tr>
<td>2460</td>
<td>005633</td>
<td>170</td>
<td>R0200: .BYTE 170</td>
</tr>
<tr>
<td>2461</td>
<td>005634</td>
<td>210</td>
<td>R1800: .BYTE 210</td>
</tr>
<tr>
<td>2463</td>
<td>005636</td>
<td>250</td>
<td>R2400: .BYTE 250</td>
</tr>
<tr>
<td>2464</td>
<td>005637</td>
<td>270</td>
<td>R3600: .BYTE 270</td>
</tr>
<tr>
<td>2465</td>
<td>005640</td>
<td>310</td>
<td>R4800: .BYTE 310</td>
</tr>
<tr>
<td>2466</td>
<td>005641</td>
<td>330</td>
<td>R7200: .BYTE 330</td>
</tr>
<tr>
<td>2467</td>
<td>005642</td>
<td>350</td>
<td>R9600: .BYTE 350</td>
</tr>
<tr>
<td>2468</td>
<td>005643</td>
<td>370</td>
<td>R10000: .BYTE 370</td>
</tr>
</tbody>
</table>

NEW: 0.0
OLD: 0.0

ENDTST
```assembly
TST17:  SCOPE

MOV 010, #TIMES ; I DO 10 ITERATIONS
MOV 017, #TESTN ; I SET TEST NUMBER IN APT MAIL BOX

IF #APTEVN SETIN #ENV THEN

BIT #APTEVN,#ENV
BEQ 50131:

EXIT TEST

BR TST20 ; I EXIT THIS TEST

ENDIF

CLR INTFLAG

LET INTFLAG := #0

GET VECTOR ADDRESS

LET R3 := DLVEC

FOR THE TRANSMITTER

LET R3 := R3 + #4

SET VECTOR TO POINT TO TRANS.SAV AT PRI

SETVEC R3, #INTSRV, #PR7

BGNSUB

MAKE SURE THAT TRANSMITTER READY IS SET

CALL TIMER IN <#500, #XMITRY, TCSR, #SET>

CLEAR INTERRUPT ENABLE

LEI @TCSR := @TCSR CLR.BY @XMITIE
```
; SET IT TO 0
; NOW SET I.E. BIT
LET @TCSR := @TCSR SET BY @XMITIE
; LET INTERRUPT HAVE TIME TO OCCUR
WAITMS 200.

; DID EXACTLY 1 INTERRUPT OCCUR
IF INFLAG NE 01 THEN
; NO - WAS IT 0 OR MORE THAN ONCE
IF INFLAG EQ 00 THEN
; TRANSMITTER DID NOT INTERRUPT IN TIME
; ERR#RD 106., DID NOT
ELSE
; TWICE
; TRANSMITTER INTERRUPTED TWICE
ERR#RD 107., TWICE
ENDIF
ENDIF

; INTERRUPT WITHOUT INTERRUPT ENABLE SET
BGNSUB
; CLEAR 'INTERRUPT OCCURED' FLAG
LET INFLAG := 00
; CLEAR INTERRUPT ENABLE
LET @TCSR := @TCSR CLR, BY @XMITIE
; NO INTERRUPTS SHOULD OCCUR.
MAINDC-11-DVDVCC-C  MACY11 30A(1052) 12-SEP-84 15:41  PAGE 61

MAINDEC-11-DVDVCC-C  MACY11 30A(1052) 12-SEP-84 08:55  PAGE 61

T17  TRANSMITTER INTERRUPT LOGIC TEST

;DARE IT TO HAPPEN

WAITMS 2

IF INTFLAG NE #0 THEN

;INTERRUPT OCCURRED WITH I E CLEARED

ENDRHD 110,NOTENAB

ENDIF

50135:

RESET

ENDSUB

;RESTORE VECTOR AREA

LET R4 := @DLVEC

MDV @DLVEC,R4

CLRVEC R4

MDV R1,-(SP)

;PUSH R1 ON STACK

MDV R2,-(SP)

;PUSH R2 ON STACK

MDV @R1,R2

ADD @R2,R2

MDV R2,(R1)+

CLR (R1)

MDV (SP)+,R2

;POP STACK INTO R2

MDV (SP)+,R1

;POP STACK INTO R1

ENDTST
T17 TRANSMITTER INTERRUPT LOGIC TEST

<table>
<thead>
<tr>
<th>Line</th>
<th>Assembly Code</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>2625</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2626</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2627</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2628</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2629</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2631</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2632</td>
<td>006202 000004</td>
<td></td>
</tr>
<tr>
<td>2633</td>
<td>006204 012767 000010 172746</td>
<td></td>
</tr>
<tr>
<td>2634</td>
<td>006212 012767 000020 172760</td>
<td></td>
</tr>
<tr>
<td>2635</td>
<td>006220</td>
<td></td>
</tr>
<tr>
<td>2636</td>
<td>006222 032767 040000 172772</td>
<td></td>
</tr>
<tr>
<td>2637</td>
<td>006226 001404</td>
<td></td>
</tr>
<tr>
<td>2638</td>
<td>006230 126727 002671 000001</td>
<td></td>
</tr>
<tr>
<td>2639</td>
<td>006236 001002</td>
<td></td>
</tr>
<tr>
<td>2640</td>
<td>006240</td>
<td></td>
</tr>
<tr>
<td>2641</td>
<td>006244</td>
<td></td>
</tr>
<tr>
<td>2642</td>
<td>006244</td>
<td></td>
</tr>
<tr>
<td>2643</td>
<td>006244</td>
<td></td>
</tr>
<tr>
<td>2644</td>
<td>006244</td>
<td></td>
</tr>
<tr>
<td>2645</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2646</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2647</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2648</td>
<td>006244 010146</td>
<td></td>
</tr>
<tr>
<td>2649</td>
<td>006246 016701 173004</td>
<td></td>
</tr>
<tr>
<td>2650</td>
<td>006252 012721 010656</td>
<td></td>
</tr>
<tr>
<td>2651</td>
<td>006256 012711 000340</td>
<td></td>
</tr>
<tr>
<td>2652</td>
<td>006262 012601</td>
<td></td>
</tr>
<tr>
<td>2653</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2654</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2655</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2656</td>
<td>006264 012767 006272 172616</td>
<td></td>
</tr>
<tr>
<td>2657</td>
<td>006272 005067 002366</td>
<td></td>
</tr>
<tr>
<td>2658</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2659</td>
<td>006276</td>
<td></td>
</tr>
<tr>
<td>2660</td>
<td>006276 052777 000004 172760</td>
<td></td>
</tr>
<tr>
<td>2661</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2662</td>
<td>006304</td>
<td></td>
</tr>
<tr>
<td>2663</td>
<td>006304 042777 000100 172746</td>
<td></td>
</tr>
<tr>
<td>2664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2665</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2666</td>
<td>006312 012746 000000</td>
<td></td>
</tr>
<tr>
<td>2667</td>
<td>006316 012746 006324</td>
<td></td>
</tr>
<tr>
<td>2668</td>
<td>006322 000002</td>
<td></td>
</tr>
<tr>
<td>2669</td>
<td>006324</td>
<td></td>
</tr>
<tr>
<td>2670</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2671</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2672</td>
<td>006324</td>
<td></td>
</tr>
<tr>
<td>2673</td>
<td>006324 105077 172740</td>
<td></td>
</tr>
<tr>
<td>2674</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2675</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2676</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2677</td>
<td>006330</td>
<td></td>
</tr>
<tr>
<td>2678</td>
<td>006330 010546</td>
<td></td>
</tr>
<tr>
<td>2679</td>
<td>006332 012745 177777</td>
<td></td>
</tr>
<tr>
<td>2680</td>
<td>006336 016745 172716</td>
<td></td>
</tr>
</tbody>
</table>

Instructions: 1. Test 20 (Receiver Interrupt Logic Test)
2. This test covers all of the receiver side of the interrupt logic in character mode.
3. Test 20: Scope 10, 8 times, 10 iterations.
4. Do test number in apt mail box if maint jump not set in usur or console eq true.
5. Clear interrupt occurred flag.
6. Set up receiver inter vector set vec dlvec, @intsv, @pr7.
7. Branch sub.
8. Clear interrupt test -recvrie.
9. Evaluate INFLAG := 00.
10. Set maint. bit, let @tcsr := @tcsr set by maint.
11. Clear interrupts, let @rcr := @rcr clr by @rcr.
12. Change priority.
13. Send a character, let @tbu := @tbu.
14. Wait a maximum of 500 msec for rcvr done to set in rcsr.
15. Call timer in <sno5, @rcvrdone, rcsr, @set>.
RECEIVER INTERRUPT LOGIC TEST

SET INTERRUPT ENABLE
LET BCRSR := BCRSR SET.BY @RCVRIE

; LET IT COME IN.
WAITMS 1

LET R0 := BRBUF : CLEAR RCVRIE

; DID HE DO IT RIGHT?
IF INFLAG NE @1 THEN

; NONE OCCURED
; CAN NOT LEAVE WITH MAINT SET
LET @BCSR := @BCSR CLR.BY @MAINT

IF INFLAG EQ @0 THEN

; RECEIVER DID NOT INTERRUPT IN TIME
ERRHARD 111,, DIDNOT

; TWICE OR MORE ELSE

; RECEIVER INTERRUPTED TWICE
ERRHARD 112,, TWICE

ENDDIF

ENDSUB

; CLEAR THE WORLD
LET BCRSR := BCRSR CLR.BY @RCVRIE

; RESET MAINT. BIT.
LET @BCSR := @BCSR CLR.BY @MAINT

LET R4 := @DLVEC
CLRVEC R4
L5

MAINDEC-11 DQVCC-C
CVQVCC.P11

RECEIVER INTERRUPT LOGIC TEST

2737 006460 010146
2736 006462 010246
2739 006464 012701 000004
2740 006470 010102
2741 006472 062702 000002
2742 006476 010221
2743 006500 005011
2744 006502 012602
2745 006504 012601
2746 006506

MOV R1,(SP) ; PUSH R1 ON STACK
MOV R2,(SP) ; PUSH R2 ON STACK
ADD R2,R2
MOV R2,(R1)+
CLR (R1)
MOV (SP)+,R2 ; POP STACK INTO R2
MOV (SP)+,R1 ; POP STACK INTO R1
ENDTST

SEQ 63
M5

MAINCDC-11-OVDC-C MACY11 30A(1052) 12-SEP-84 15:41 PAGE 65

CVODCC P11 12-SEP-84 08:55 T20 RECEIVER INTERRUPT LOGIC TEST

; TEST 21 TEST ACTUAL DATA TRANSFERED
; NON-INTERRUPT MAINTENANCE BIT SET

; ******************************************************************************
; ******************************************************************************
; TST21: SCOPE

2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905

; DO 1 ITERATION
; SET TEST NUMBER IN APT MAIL BOX
; IF MAINT.JUMP NOTSET IN USRM OR CONSOLE EQ TRUE

MDV 01, ITIMES
MDV #21, ITESN
MDV 021, ITESN

IF #MAINT.JUMP NOTSET IN USRM OR CONSOLE EQ TRUE

501431:

CLR ERROR

LET ERROR := #0

LET MAINT.BIT := #0 MAINT.

LET BTCSR := BTCSR SET BY MAINT.

; CHANGE PRIORITY
; TO 0

MDV #PRO.-(SP)
MDV #PRO.-(SP)
MDV #4641.(SP)
MDV #4641.(SP)
MDV (R5).*R1

RTI

64:

; GET DATA MASK.

CALL DATLING OUT <R1>

SUB #1a2,R5
JSR PC.DATLING
MDV (R5).*R1

MDV #DBUF,R0

LET RO := #DBUF ; START CLEAR

INC R2 FROM #0 TO #377 BY #1

MDV R5.*-(SP)
MDV #SET.*-(R5)

; TRANSMIT CHAR IN R2

CALC TIMER IN <#500,#XMITR0,#TCSR,#SET>

; ******************************************************************************
; ******************************************************************************
IF ERROR THEN
LET ERRCHK := ERRCHK SET BY @BIT3
ENDIF

;TRANSMIT IT
LET BTBUF := R2

CALL TIMER IN <#500,#RCVRDONE,#RCSR,#SET>

IF ERROR THEN
LET ERRCHK := ERRCHK SET BY @BIT4
ENDIF

; AND SAVE IT
LET R3 := BRBUF

;COMPARE TO SEE IF WE RECEIVED IT ALL
;CLEAN OFF NON-DATA BITS
;ON BOTH TRANSMITTED AND
LET R4 := R2 CLR BY R1
LET R3 := R3 CLR BY R1

; RECEIVED DATA
IF R4 NE R3 THEN
;DATA COMPARE ERROR
;CAN NOT LEAVE WITH MAINT SET
LET @TCSR := @TCSR CLR BY @MAINT
ERRRND 116,COMP,SBMAS
EXIT TEST ; ON ERROR
MOV 01, @TIMES
BR TST22

EXIT THIS TEST
ENDIF

ENDINC R2

;RESET MAINT. BIT.
LET OITCSR 1: OITCSR CLR.BY OMAINT
IF OBIT3 SET IN ERRCHK THEN

ERROR 130

END IF

ERROR 131

END IF
007030 000004
007032 012767 000001 172120
007040 012767 000022 172132
007046 000004
007048 032767 020000 172144
007050 010104
007056 032767 000100 172154
007064 001004
007066
007068
007074 000516
007076
007078
007080
007086
007090 012767 000001 172054
007096 000516
007098
00709e
0070a0 032777 000004 172156
007100 052777 000011 172146
007106 052777 000011 172146
007112 012746 000000
007114 012746 007124
007122 000002
007124
007126
007128
00712a
00712c
007134 012501
007136
007138
007140 017700 172120
007142
007144 000002
007146 000401
007148 005202
007150 020227 000377
007154 005063

007034 000004
007035 012767 000001 172120
007040 012767 000022 172132
007046 000004
007048 032767 020000 172144
007050 010104
007056 032767 000100 172154
007064 001004
007066
007068
007074 000516
007076
007078
007080
007086
007090 012767 000001 172054
007096 000516
007098
00709e
0070a0 032777 000004 172156
007100 052777 000011 172146
007106 052777 000011 172146
007112 012746 000000
007114 012746 007124
007122 000002
007124
007126
007128
00712a
007134 012501
007136
007138
007140 017700 172120
007142
007144 000002
007146 000401
007148 005202
007150 020227 000377
007154 005063

007112 012746 000000
007114 012746 007124
007122 000002
007124
007126
007128
00712a
007134 012501
007136
007138
007140 017700 172120
007142
007144 000002
007146 000401
007148 005202
007150 020227 000377
007154 005063

007112 012746 000000
007114 012746 007124
007122 000002
007124
007126
007128
00712a
007134 012501
007136
007138
007140 017700 172120
007142
007144 000002
007146 000401
007148 005202
007150 020227 000377
007154 005063
;TRANSMIT THE CHAR. IN R2.
;MAKE SURE IT'S READY
CALL TIMER IN "$500,"$XVRDY.$CRA.$SET

IF .ERROR THEN
;TRANSMITTER NEVER BECAME READY
EXIT TEST

;EXIT THIS TEST
ENDIF

;START IT ON ITS WAY
LET $TBUF := R2
;NOW WAIT FOR RECEIVER DONE
CALL TIMER IN "$500,"$XVRDY.$CRA.$SET

IF .ERROR THEN
ERROR 124
;RECEIVER NEVER BECAME READY
EXIT TEST

;EXIT THIS TEST
ENDIF

;RETRIEVE
LET R3 := $TBUF

;STRIP OFF JUNK ON BOTH
LET R4 := R2 CLR.BY R1
LET R5 := R3 CLR.BY R1

;WE HAVE TROUBLE
IF R4 NE R3 THEN

; DATA COMPARISON ERROR
ERRAWD 117, COMP, SBWAS

EXIT TEST ; ON ERROR

TST25

; EXIT THIS TEST
ENDIF

; NOW THAT THE TEST IS DONE
; WE WILL TOGGLE READER RUN
; TO TURN OFF THE SPECIAL MODULE.
LET RCSR := RCSR SET BY 011

ENDTST
;****************************************************************************
;** TEST 23  FULL DATA TRANSFER WITH INTERRUPTS
;** AND MAINTENANCE MODE.
;****************************************************************************

;TST25: SCPE
         MOV 07,0TINES
         MOV 025,0TESTN
         MOV 01,0ITERATION
         MOV 023,0TESTN
         MOV 01,0ITERATION
         MOV 025,0TESTN
         MOV 01,0ITERATION
         CALL 01,0ITERATION

         IF MAINT_JUMP 0SETIN 0USER ORB 0CONSOLE EQ 0TRUE

         BIT MAINT_JUMP 0USER
         BEQ 501651
         CMP CONSOLE 0TRUE
         BNE 501661

         501651:
         EXIT TEST
         ENDIF

         501661:
         ;GET DATA MASK
         CALL DATMG OUT <R3>

         ;THIS TEST WILL RUN BOTH TRANSMITTER AND
         ;RECEIVER AT FULL SPEED TESTING
         ;THE ABILITY OF THE MODULE
         ;TO HANDLE INTERRUPTS FROM BOTH SIDES
         ;AT ONCE. ALSO, THE DOUBLE BUFFERING
         ;OF THE UART WILL BE FULLY TESTED.
         ;THIS TEST WILL TRANSFER A MAXIMUM OF 400(8)
         ;CHARACTERS THROUGH THE MODULE, BUT IF AN ERROR
         ;IS DETECTED BY THE TEST A PREMATURE SHUTDOWN OCCURS.

         ;CHANGE PRIORITY
         ;.TO 0

         ;GET VECTOR ADDRESS
         LET R1 := DLVEC

         ;RCVR VECTOR
         LET (R1)+ := @REC
         LET (R1)+ := @PR7

         ;POINT TO TRANSMITTER VECTOR
         ;AND SET IT UP ALSO
         LET (R1)+ := @TRAN

         MOV @PRO,-(SP)
         PUT NEW PS ON STACK
         MOV @648,-(SP)
         PUT NEW PC ON STACK
         MOV (R5)+,R3

         SUB @1+2,R5
         JSR PC,DATMG
         MOV (R5)+,R3

         MOV 07412,012746 000000
         MOV 07416,012746 007424
         MOV 07422 000002
         RTI

         640:
         ;END

         MOV 07424,012746 007570
         MOV 07440,012721 007570
LET (R1) := @PR7

CLEAR ERROR COUNTER
LET ERRCNT := 0

INITIALIZE COUNTERS
LET R1 := 0.1

RECEIVER STORAGE
LET R2 := 0

OF RECEIVED CHAR. COUNT.
LET R4 := 0.1

RESET ;SET UP ALL REGISTERS

SET UP MAINTENANCE
LET @TCSR := @TCSR SET.BY @maint

;SET I.E. IN TRANSMITTER
LET @TCSR := @TCSR SET.BY @xmitie

;AND RECEIVER
LET @BCSR := @BCSR SET.BY @rcvrie

:NOW WE WAIT UNTIL R4 COUNT (RECEIVED) IS EQUAL
:REPEAT
UNTIL R4 EQ NUMBER OR ERRCNT GT 00

LET @TCSR := @TCSR CLR.BY @maint

CHECK FOR DATA COMPARE ERRORS.
IF ERRCNT NE 00 THEN

;DATA COMPARE ERROR
ERRHDL 120,COMP,FIRST
ENDIF

LET @TCSR := @TCSR CLR.BY @xmitie

LET @BCSR := @BCSR CLR.BY @rcvrie
EXIT

;SKIP OVER SUPPORT ROUTINES & STORAGE
BR TST24

;EXIT THIS TEST

;TRANSMIT INTERRUPT HANDLER
BGNsrv tran

;INCREMENT CHAR COUNT
LET R1 := R1 + 1

;SET UP FOR TRANSFER
LET HOLD := R1 CLR.BY R3

;AND SEND
LET @TBUF := HOLD

;ALL DONE
IF R1 EQ NUMBER THEN

;STOP INTERRUPT PROCESSING
LET @TCSR := @TCSR CLR.BY @XMITIE

ENDIF

;RECEIVER INTERRUPT HANDLER
BGNsrv rec

;COUNT THIS CHAR
LET R4 := R4 + 1

;GET CHAR IN + MASK IT
LET R2 := @RBUF CLR.BY R3

MOV @RBUF,.R2

INC R4
BIC R3, R2

MOV R4, RHL D

BIC R3, RHL D

CMP R2, RHL D

BEQ 50173:

TST ERRCNT

BNE 50174:

MOV RHL D, SB

MOV R2, WAS

50174:

INC ERRCNT

50173:

CMP R4, NUMBER

BNE 50175:

STOP RECEIVER INTERRUPTS

LET BRCSR := BRCSR CLR.BY #ACVRIE

BIC #ACVRIE, BRCSR

#ACVRIE, BRCSR

50175:

BR ZZZZ

EXIT SRV

RHL D: 0

ZZZZ:

RTI

ENDSRV

ENDTST
MAINDEC-11-DVDC-C  MACY11 30A(1052)  12-SEP-84  15:41  PAGE 75
CVDCC.P11  12-SEP-84  06:55  T23  FULL DATA TRANSFER WITH INTERRUPTS

;******************************************************************************
;** TEST 24  TEST BREAK GENERATION LOGIC
;******************************************************************************
;* TRANSMIT KNOWN CHAR WITH BREAK SET
;* AND COMPARE RECEIVED WITH 0.
;* FRAMING ERROR WILL ALSO BE CHECKED
;* IF ERROR BITS ARE ENABLED.
;******************************************************************************

TST24:  SCOPE
07726  000004
07730  012767  00010  171222
07736  012767  000024  171234
07744  032767  04000  171246
07752  001404
07754  032767  01000  171236
07762  001004

50176:  EXIT TEST

50177:  IFB CONSOLE EQ @TRUE

50200:  LET ERRCHK := #0  ; CLEAR ERROR WORD

50235  010042
50239  010044  012745  177777
50243  010050  016745  171204
50247  010054  012745  000200
50251  010056  012745  00250
50255  010064  004767  000230
50259  010070  012605
50263  010072
50267  010072  103001
50271  010074
50275  010074  104115

ERROR 115
3303 010076
3304 010076
3305 010076
3306 010076
3307 010076 105777 171160
3308 010102 001404
3309 010104
3310 010104
3311 010104 052767 000001 000060
3312 010112
3313 010112 000413
3314 010112
3315 010114
3316 010114 032767 100000 171076
3317 010122 001407
3318 010124
3319 010124 032777 020000 171130
3320 010132 001003
3321 010134
3322 010134 052767 000002 000030
3323 010142
3324 010142
3325 010142
3326 010142
3327 010142
3328 010142
3329 010142
3330 010142 000005
3331 010142
3332 010144
3333 010144 032767 000001 000020
3334 010152 001401
3335 010154
3336 010154 104121
3337 010154
3338 010156
3339 010156
3340 010156
3341 010156 032767 000002 000006
3342 010164 001401
3343 010166
3344 010166 104122
3345 010166
3346 010170
3347 010170
3348 010170 000401
3349 010172 000000
3350 010174

ENDTF

IFB BRBUF NE 00 THEN

; BREAK DID NOT EQUAL 0
LET ERRCHK := ERRCHK SET BY #BIT0
ELSE

IF #ERRBITS SETIN #USR THEN

IF #FRERR NOTSETIN BRBUF THEN

LET ERRCHK := ERRCHK SET BY #BIT1

ENDIF

ENDIF

ENDIF

BLESET 1CLEAN UP

IF #BIT0 SETIN ERRCHK THEN

ERRHNRD 121 ; BREAK ERROR

ENDIF

IF #BIT1 SETIN ERRCHK THEN

ERRHNRD 122 ; FRAMING ERROR

ENDIF

EXIT 1EXIT THIS TEST

ERRCHK := WORD 0

ENDTF
*TEST 23 NOT A TEST - SEND BACK TO LOOP

TST25: SCOPE

MOV #01,0 TIMES

TYPE ,654

BR 644

GET OVER THE ASCIZ

11650: .ASCIZ <CRLF>CS:

640:

MOV DLADD,-(SP)

SAVE DLADD FOR TYPEOUT

TYPDC

GO TYPE--OCTAL ASCII(ALL DIGITS)

TYPE ,671

TYPE ASCII STRING

PR 660

GET OVER THE ASCII

11670: .ASCIZ *,VECTOR:

660:

MOV DLVDC,-(SP)

SAVE DLVDC FOR TYPEOUT

TYPDC

GO TYPE--OCTAL ASCII(ALL DIGITS)

TYPE ,691

TYPE ASCII STRING

BR 681

GET OVER THE ASCII

11690: .ASCIZ *,ERRORS:

681:

MOV #ERTTL,-(SP)

SAVE #ERTTL FOR TYPEOUT

TYPDS

GO TYPE--DECIMAL ASCII WITH SIGN

CLR #ERTTL

RESET FOR NEXT DEVICE/PASS

TYPE ,#CRLF

JMP LOOP

BACK UP TO THE BEGINNING
```
; SUBROUTINE
; ROUTINE TIMER <MOLONG, WHICHBIT, REG, SETCLR>
; TIMER:
; ROUTINE: TIMER
; THIS ROUTINE IS USED TO TEST THE STATUS OF ANY BIT
; IN ANY REGISTER.
; INPUTS:
; MOLONG     THE MAXIMUM AMOUNT OF TIME TO SPEND IN
; THIS ROUTINE.
; WHICHBIT    A MASK WITH THE BIT(S) SET THAT ARE
; TO BE CHECKED.
; REG        A POINTER TO THE REGISTER TO BE CHECKED
; SETCLR     THE DESIRED RESULTS
; EITHER #SET OR #CLEAR
; OUTPUT:
; THE 'C' BIT IS SET TO INDICATE AN ERROR
; BUT IT IS TESTED BY THE IFERROR STATEMENT
;
; NOTE:  THE USE OF (R5) IS PART OF THE LINKAGE
; MECHANISM BETWEEN THE CALLER AND THE CALLED

; .ENABL LSB
; TRUE=   1
; FALSE=  0

; LET REGSAV := REG(R5) ; GET POINTER TO REGIST
; LET TIMSAV := MOLONG(R5) ; SAVE MOLONG FOR
; LET FLAG:B := FALSE ; INITIALIZE THE EXIT FLA

; START OF AN INFINITE LOOP

; LOOP
; TEST TO SEE IF WHICHBIT IS SET
; IF WHICHBIT(R5) NOTSET, IN REGSAV THEN

; LET MOLDS :B := #CLR

; ELSE

; LET MOLDS :B := #SET ; REMEMBER THIS

; ENDIF

; NOW SEE IF THAT WAS WHAT WE WANTED
; IFB MOLDS EQ SETCLR(R5) THEN
```
ROUTINE DATING + (MASK)

ROUTINE: DATING

INPUT: - NOTHING IS PASSED TO THIS ROUTINE

GLOBAL INFORMATION: ASSUMED TO EXIST:

USER -- THE WORD WITH SOFTWARE PARAMETERS

DATA -- A MASK FOR THE LOCATION OF THE OCTAL

NUMBER OF DATA BITS

OUTPUT:

MASK -- A MASK OF BINARY ONES RIGHT-justified

THE NUMBER OF WHICH IS DEFINED IN USER WORD.

END?4

LENBL LSB

LET MASK(R5) := #0

LET NUMBER := USER AND #DATA

INCR I FROM #1 TO NUMBER BY #1

LET MASK(R5) := MASK(R5) SHIFT 1

LET MASK(R5) := MASK(R5) SET BY #1

ENDINC

LET MASK(R5) := COMP MASK(R5)

RETURN

ENDRTN
ROUTINE WAIT <TIME>

WAIT:

; ROUTINE=WAIT
; THIS ROUTINE IS USED TO DELAY EXECUTION OF THE
; MAIN PROGRAM FOR A SPECIFIED AMOUNT OF TIME.
; THIS IS ACCOMPLISHED BY INCREMENTS A
; REGISTER UP TO A LIMIT. THE INNER LOOP IS SET
; TO APPROXIMATE 1 MILLI SEC.

MOV R1,-(SP) ; PUSH R1 ON STACK
MOV R2,-(SP) ; PUSH R2 ON STACK
MOV R3,-(SP) ; PUSH R3 ON STACK
LET R1 = TIME(R5)
MOV TIME(R5),R1
INCR R2 FROM 01 TO R1 BY 01
MOV 01,R2
BR 500024

500024:
ADD #01,R2

500025:
CMP R2,R1
BHI 500044

500044:
INC R3 FROM #0 TO #100 BY #1
CLR R3
BR 500051

500051:
INC R3

500052:
CMP R3,#100
BGT 500071

500071:
ENDINC
BR 500061

500061:
ENDINC
BR 500031

500031:
MOV (SP),R3 ; POP STACK INTO R3
MOV (SP),R2 ; POP STACK INTO R2
MOV (SP),R1 ; POP STACK INTO R1

ENDRTNM

500001:
RTS PC
.SB TTL INTSRV INTERRUPT SERVICE ROUTINE

**INTSRV:**

; SERVICE ROUTINE: INTSRV
; THIS GLOBAL ROUTINE DOES NOTHING BUT INCREMENT
; 'INTFLAG' EACH TIME IT IS CALLED. IT ASSUMES
; THAT THE MAIN CALLING ROUTINE WILL KNOW WHAT
; TO LOOK FOR.

;ADD 1 TO 'INTERRUPT OCCURRED' FLAG
LET INTFLAG := INTFLAG + 1
INC INTFLAG
ENDSRV
RTI

INTFLAG: 0
ROUTINE CYCLE

CYCLE:

ROUTINE: CYCLE

** This routine causes ADRS to point to the
** address of DLVII-F Under Test, ADRS +2 to
** point to the vector of the DLVII-F Under Test.
** It keeps track of the current device and bit
** masks. The console is treated special by this routine.
** It is only tested once if under apt, if not under apt
** all tests that require the main bit are not run.

```
; Enable LSB
LET APTCON := #FALSE ; Set default value
LET CONSOLE := #FALSE

REPEAT ; Until bitmask setin #DLVM

50002:\
TST BITMASK
BNE 50003\

IF INITFLAG EQ #00 THEN

50004:\
CLR INITFLAG
ELSE

CALL #EOP ; As a subroutine

SPECIALADDRESS: ; Because #EOP returns as a jump
LET RO := POP

ENDIF

50005:\
MOV (SP)+,RO

50006:\
BR 50003\

LET R4 := #10
LET BITMASK := BITMASK ROTATE 1
LET ADDRESS := ADDRESS + R4
LET VECTOR := VECTOR + R4
```
ADD R4,VECTOR
ENDIF

500061:
UNTIL BITMASK SET IN ADEVM

BIT BITMASK,#ADEVM
BEQ 500024
IF BITMASK EQ #BIT15 THEN

CMP BITMASK,#BIT15
BNE 500024
LET CONSOLE :B= #TRUE

MOV @TRUE,CONSOLE
LET ADDRESS := CONADR

MOV CONADR,ADDRESS
LET VECTOR := CONVEXT

MOV CONVEXT,VECTOR
IF #CONMAINT HOTSETIN #USUR THEN
LET NOCONMAINT :B= #TRUE
ENDIF
IF #APTENV SETIN #ENV THEN ; APT MODE

ENDIF

500024:

BIT @APTENV,#ENV
BEQ 500014
IF #PASS ME #0 THEN ; NOT FIRST PASS

TST #PASS
BEQ 500014
DEFINE DEVICE AS APT CONSOLE
LET APTCON :B= #TRUE

MOV @TRUE,APTCN
ENDIF ; FIRST PASS

ENDIF ; APT

ENDIF ; BITMASK

500014:

500014:

500014:

500014:

500074:

500074:

500074:

500074:

500074:

LET ADRS := #ADDRESS

MOV #ADDRESS,ADRS
LET #DEVCT := #DEVCT + #1

INC #DEVCT
RETURN

BR 500001
; CONSOLE FIRST

BITMASK: 100000
INITFLAG: 1
ADDRESS: 0
VECTOR: 0
OK: 0
CONADR: 177560
CONNECT: 60
; CONSOLE ADDRESS
; CONSOLE VECTOR
APTCON: .BYTE 0
CONSOLE: .BYTE 0
NOCOMMART: .BYTE 0

ENDRTN

500001:

500011:

RTS PC

.DSABL LSB
ROUTINE MYTYPE
MYTYPE:

TYPE .650
тип ASCII STRING
BR 641
GET OVER THE ASCII

MOV $TESTH,-(SP)  Irene save TESTH for TYPEOUT
MOV 0,TYPE--OCTAL ASCII(ALL DIGITS)

TYPE .670
тип ASCII STRING
BR 641
GET OVER THE ASCII

ERROR 0

ITEM0,FATAL 5
APT FATAL ERROR NUMBER

BYTE 6
TYPE ASCII STRING
BR 681
GET OVER THE ASCII

PC 0

ERRPC,-(SP)  Irene save ERRPC FOR TYPEOUT
MOV 0,TYPE--OCTAL ASCII(ALL DIGITS)

TYPE .710
тип ASCII STRING
BR 701
GET OVER THE ASCII

CSR 0

DLADD,-(SP)  Irene save DLADD FOR TYPEOUT
MOV 0,TYPE--OCTAL ASCII(ALL DIGITS)

TYPE .750
тип ASCII STRING
BR 720
GET OVER THE ASCII

VECTOR 0

DLVEC,-(SP)  Irene save DLVEC FOR TYPEOUT
MOV 0,TYPE--OCTAL ASCII(ALL DIGITS)

CRLF 0

ENDRTN
000004 5
000014
0011320
00207
RTS PC
SBTL END OF PASS ROUTINE

***---------------------------------------------------------------------***

; INCREMENT THE PASS NUMBER (APASS)
; INDICATE END-OF-PROGAM AFTER 1 PASSES THRU THE PROGRAM
; TYPE "END PASS XXXXX"  (WHERE XXXXX IS A DECIMAL NUMBER)
; IF THERE'S A MONITOR GO TO IT
; IF THERE ISN'T JUMP TO SPECIAL ADDRESS

$EUP:
SCAPE
CLR $TSTNM  ; ZERO THE TEST NUMBER
CLR $TREMS  ; ZERO THE NUMBER OF ITERATIONS
INC $PASS  ; INCREMENT THE PASS NUMBER
BIC $100000,$PASS  ; DON'T ALLOW A NEG. NUMBER
DEC (PC)+  ; LOOP?

$EOPCT: .WORD 1
BGT $DOAGN  ; YES
MOV (PC)+,$KPC+  ; RESTORE COUNTER

$ENDCT: .WORD 1
$EOPCT

TYPE .ENDNG  ; TYPE "END PASS 0"
MOV $PASS,+(SP)  ; SAVE #PAS FOR TYPEOUT
TYPS  ; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $ENUL  ; TYPE A NULL CHARACTER
@GET42: MOV $E42,RO  ; GET MONITOR ADDRESS
BEG $DOAGN  ; BRANCH IF NO MONITOR
$RESET  ; CLEAR THE WORLD

$ENDAD: JSR PC,(RO)  ; GO TO MONITOR
NOP
NOP  ; SAVE Rom
NOP
NOP  ; FOR
NOP
JMP $KPC+  ; RETURN

$RTMAD: .WORD SPECIAL ADDRESS

$ENUL: .BYTE -1,-1,0  ; NULL CHARACTER STRING

$SNAG: .ASCIZ <15><12>/END PASS 0/
.SBTTL POWER DOWN AND UP ROUTINES

POWER DOWN ROUTINE

;POWER DOWN ROUTINE
3799 011444 012737 011610 000024
;PUWRDN: MOV @11LUP,#0PBWMVEC ;SET FOR FAST UP
3800 011432 012737 000340 000026
MOV @340,#0PBWMVEC+2 ;PRIOR:7
3801 011460 010046
MOV R0,-(SP) ;PUSH R0 ON STACK
3802 011462 010146
MOV R1,-(SP) ;PUSH R1 ON STACK
3803 011464 010246
MOV R2,-(SP) ;PUSH R2 ON STACK
3804 011466 010346
MOV R3,-(SP) ;PUSH R3 ON STACK
3805 011470 010446
MOV R4,-(SP) ;PUSH R4 ON STACK
3806 011472 010546
MOV R5,-(SP) ;PUSH R5 ON STACK
3807 011474 017746 167440
MOV BSR,-(SP) ;PUSH BSR ON STACK
3808 011500 010667 000110
MOV SP,#SAVR6 ;SAVE SP
3809 011504 012737 011516 000024
MOV @1UPUP,#0PBWMVEC ;SET UP VECTOR
3810 011512 000000
HALT
3811 011514 000776
BR -.2 ;HANG UP

POWER UP ROUTINE

3813 011516 012737 011610 000024
;PUUP: MOV @11LUP,#0PBWMVEC ;SET FOR FAST DOWN
3814 011524 016706 000064
MOV #SAVR6,SP ;GET SP
3815 011530 005067 000060
CLR #SAVR6 ;WAIT LOOP FOR THE TTY
3816 011534 005267 000054
10: INC #SAVR6 ;WAIT FOR THE INC
3817 011540 001375
OF WORK 10:
3818 011542 012677 167372
MOV (SP),+BSMR ;POP STACK INTO BSMR
3819 011544 012605
MOV (SP),R5 ;POP STACK INTO R5
3820 011550 012604
MOV (SP),R4 ;POP STACK INTO R4
3821 011552 012603
MOV (SP),R3 ;POP STACK INTO R3
3822 011554 012602
MOV (SP),R2 ;POP STACK INTO R2
3823 011556 012601
MOV (SP),R1 ;POP STACK INTO R1
3824 011558 012600
MOV (SP),R0 ;POP STACK INTO R0
3825 011560 012600
MOV @1UPUP,#0PBWMVEC ;SET UP THE POWER DOWN VECTOR
3826 011570 012737 000340 000026
MOV @340,#0PBWMVEC+2 ;PRIOR:7
3827 011576 104401
;PUWRMG: .WORD .POWER ;POWER FAIL MESSAGE POINTER
3828 011580 011606 000000
;PUWRMG: .WORD .POWER ;POWER FALL MESSAGE POINTER
3829 011602 012716
MOV (PC),+SP ;RESTART AT START
3830 011604 001336
;PUWRD: .WORD .START ;RESTART ADDRESS
3831 011606 000000
;PUWRD: .WORD .START ;RESTART ADDRESS
3832 011610 000000
;PUWRD: .WORD .START ;RESTART ADDRESS
3833 011612 000776
BR -.2 ;BEFORE THE POWER DOWN WAS COMPLETE
3834 011614 000600
;SAVR6: 0 ;PUT THE SP HERE
3835 011616 005015 047520 042527
;POWER: .ASCIZ <15><12>"POWER"
3836 011624 000122
.EVEN
.SB TTL TYPE ROUTINE

**#ROUTINE TO TYPE ASCII 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 'FTERN.**

**#CALL:**

**#1) USING A TRAP INSTRUCTION**

**#2) OR**

**#3) TYPE**

**#4) MESADR**

**#5) MESADR IS FIRST ADDRESS OF AN ASCII STRING**

```
3857 011626 105767 167325    TYPE: TSTB TPFLG
3858 011632 100002
3859 011634 000000
3860 011636 000430
3861 011640 010046
3862 011642 017600 000002
3863 011646 122767 000001 167340
3864 011654 001011
3865 011656 132767 000100 167331
3866 011664 001405
3867 011666 010007 000004
3868 011672 004762 001046
3869 011676 000000
3870 011700 132767 000040 167307
3871 011706 001003
3872 011710 112046
3873 011712 001005
3874 011714 005726
3875 011716 012600
3876 011720 062716 000002
3877 011724 000002
3878 011726 122716 000111
3879 011732 001430
3880 011734 122716 000200
3881 011740 010016
3882 011742 000526
3883 011744 104401
3884 011746 001117
3885 011750 105067 00202
3886 011754 003753
3887 011756 004767 000056
3888 011762 126716 167170
3889 011766 001350
3890 011770 161746 07160
3891 011774 105366 000001
3892 011776 004767 00032
3893 012000 002770
3894 012002 004767 00032
3895 012006 105367 000144
```

**IS THERE A TERMINAL?**

**BR IF YES**

**HALT HERE IF NO TERMINAL**

**LEAVE**

**SAVE RO**

**GET ADDRESS OF ASCII STRING**

**RUNNING IN APT MODE**

**STOP CHECK FOR APT CONSOLE**

**MESSAGE ADDRESS**

**AFT VER 0**

**AFT CONSOLE SUPPRESSED**

**YES, SKIP TYPE OUT**

**PAUSE CHARACTER TO BE TYPED ONTO STACK**

**BR IF IT ISN'T THE TERMINATOR**

**IF TERMINATOR POP IT OFF THE STACK**

**RESTORE RO**

**RETURN**

**BRANCH IF <HT**

**BRANCH IF NOT <CRLF**

**CLEAR CHARACTER COUNT**

**GET NEXT CHARACTER**

**IS IT TIME FOR FILLER CHARMS?**

**GET 0 OF FILLER CHARMS, NEEDED**

**AND THE NULL CHAR.**

**DOES A NULL NEED TO BE TYPED?**

**BR IF NO--GO POP THE NULL OFF OF STACK**

**GO TYPE A NULL**

**DO NOT COUNT AS A COUNT**
TTY INPUT ROUTINE

.SBTTL TTY INPUT ROUTINE

; START OF SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
; ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
; SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
; WHEN OPERATING IN TTY FLAG MODE.

; IS THE SOFT-SWR SELECTED?
.CKSMR: CMP @SWREG,SUR
  BNE 150 ; BRANCH IF NO
  TSTB @IKS
  BPL 150 ; CHAR THERE?
  MOVB @IKB, -(SP)
  BPL 150 ; IF NO, DON'T WAIT AROUND
  MOVB @MKB, -(SP)
  BPL 150 ; SAVE THE CHAR
  BIC @C177, -(SP)
  CMP 07, -(SP)
  BNE 150 ; IS IT A CONTROL G?
  MOVB @AUTOB, 01
  BIC 150 ; ARE WE RUNNING IN AUTO-MODE?
  BR 150 ; BRANCH IF YES

; ECHO THE CONTROL-G (+G)
.GTSMR: TYPE .ICNTLG
; TYPE CURRENT CONTENTS
  MOV @SWREG, -(SP)
  CMP @MKB
  BNE 150 ; SAVE SWREG FOR TYPEOUT
  CMP @C177, -(SP)
  BNE 150 ; GO TYPE - OCTAL ASCII(ALLOW DIGITS)
  TYPE .IPMNEW
  CMP 06, -(SP)
  BNE 150 ; PROMPT FOR NEW SUR
  CLR -(SP)
  CLR -(SP)
  JMP 150 ; CLEAR COUNTER
  CLR -(SP)
  JMP 150 ; THE NEW SUR
  JMP 150 ; IF NOT TRY AGAIN

; MAKE IT 7-BIT ASCII
.TXTSMR: MOVB @MKB, -(SP)
  BNE 150 ; PICK UP CHAR
  MOVB @C177, -(SP)
  JMP 150 ; MAKE IT 7-BIT ASCII

; IS IT A CONTROL-U?
.CNTLS: CMP (SP), 025
  BNE 150 ; BRANCH IF NOT
  JMP 150 ; YES, ECHO CONTROL-U (+U)

; IGNORE PREVIOUS INPUT
.ADDR: ADD 06, SP
  BR 190 ; LET'S TRY IT AGAIN

; IS IT A <CR>?
.CMSR: CMP (SP), 015
  BNE 160 ; BRANCH IF NO
  CMP (SP), 012
  BNE 160 ; YES, IS IT THE FIRST CHAR?
  CMP (SP), 014
  BNE 160 ; BRANCH IF YES
  CMP (SP), 010
  BNE 160 ; SAVE NEW SUR
  CMP (SP), 008
  BNE 160 ; CLEAR UP STACK
  CMP (SP), 006
  BNE 160 ; ECHO <CR> AND <LF>
  CMP (SP), 004
  BNE 160 ; RE-ENABLE TTY KBD INTERRUPTS?
  CMP (SP), 002
  BNE 160 ; BRANCH IF NOT
  JMP 150 ; RE-ENABLE - TTY KBD INTERRUPTS
  JMP 150 ; RETURN

; ECHO CHAR
.PC: JMP .TYPEPC
  JMP 150 ; ECHO CHAR

; TTY INPUT ROUTINE

; END OF SOFTWARE SWITCH REGISTER CHANGE ROUTINE.

; TTY INPUT ROUTINE
```assembly
; TTY INPUT ROUTINE
;
; MAINDEC-11 DVOC-C MACYI1 3OA(1052) 12-SPT-84 15:41 PAGE 92
; CVDOC.C-11 12-SPT-84 06:35

3992   012374 002420  BLT 181 ; BRANCH IF YES
3993   012376 002167  CMP (SP),#67 ; CHAR > 77
3994   012402 003015  BGT 181 ; BRANCH IF YES
3995   012404 004736  BIC #60,(SP) ; STRIP-OFF ASCII
3996   012410 005766 000002  TST 2(SP) ; IS THIS THE FIRST CHAR
3997   012414 001403  BEQ 171 ; BRANCH IF YES
3998   012416 006316  ASL (SP) ; CHAR REMAIN
3999   012420 006316  ASL (SP) ; CHAR OVER TO MAKE
4000   012422 006316  ASL (SP) ; ROOM FOR NEW ONE.
4001   012424 002566 000002  INC 2(SP) ; KEEP COUNT OF CHAR
4002   012430 006616 177776  BIS -2(SP),(SP) ; SET IN NEW CHAR
4003   012434 000707  BRA 71 ; GET THE NEXT ONE
4004   012436 104401 001170  181 ; TYPE .16UES
4005   012442 000720  BR 201 ; SIMULATE CONTROL-U
4006
4007
4008
4009
4010
4011
4012
4013
4014
4015
4016
4017   012444 011646  INCHR: MOV (SP),-(SP) ; PUSH DOWN THE PC
4018   012446 016666 000004 000002  MOV 4(SP),2(SP) ; SAVE THE PS
4019   012454 105777 166446 14:  TSTB $0TKS ; WAIT FOR
4020   012460 100375  BPL 10 ; A CHARACTER
4021   012462 117766 166446 000004 000002  MOVB $0TKS,#4(SP) ; READ THE TTY
4022   012470 042766 177600 00000C 000001  BIC $0C1777,#4(SP) ; GET RID OF JUNK IF ANY
4023   012476 006457 000000 000023  CMP 4(SP),#023 ; IS IT A CONTROL-S?
4024   012504 001013  BNE 31 ; BRANCH IF NO
4025   012506 105777 166432 21:  TSTB $0TKS ; WAIT FOR A CHARACTER
4026   012512 100375  BPL 20 ; LOOP UNTIL ITS THERE
4027   012514 117746 166246 000004 000002  MOVB $0TKS,#-(SP) ; GET CHARACTER
4028   012520 042766 177600 00000C 000001  BIC $0C1777,(SP) ; MAKE IT 7-BIT ASCII
4029   012526 022627 000021  CMP (SP),#21 ; IS IT A CONTROL-Q?
4030   012530 001386  BNE 20 ; IF NOT DISCARD IT
4031   012532 000750  BR 10 ; YES, RESUME
4032   012534 026447 000004 000140 31:  CMP 4(SP),#140 ; IS IT UPPERCASE?
4033   012542 002407  BLT 40 ; BRANCH IF YES
4034   012544 025627 000004 000175  CMP 4(SP),#175 ; IS IT A SPECIAL CHAR?
4035   012552 003003  BGT 48 ; BRANCH IF YES
4036   012554 042766 000040 000004 000000 40:  RTR ; GO BACK TO USER
4037   012562 000002
4038
4039
4040
4041
4042
4043
4044
4045   012564 010346  RDLIN: MOV R3,-(SP) ; SAVE R3
4046   012566 012703 012672 14:  MOV $#TYYIN,R3 ; GET ADDRESS
4047   012572 022703 012702 24:  CMP $#TYYIN+8,R3 ; BUFFER FULL?

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

; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
; CALL:
; RDLIN
; RETURN HERE
; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
; TERMINATOR WILL BE A BYTE OF ALL 0'S
```
B8

MAINDEC-11 DVOCV-C
MACR 11 304A (1002) 12-SEP-84 15:41 PAGE 93

CVOCV.C 11 12-SEP-84 08:55 TTY INPUT ROUTINE

4048 012576 101405 BLO$ 48 'BR IF YES
4049 012560 104410 RCHR 'GO READ ONE CHARACTER FROM THE
4050 012562 112613 MOVE (SP)+,(R3) 'GET CHARACTER
4051 012564 122713 CMPB 0177,(R3) 'IS IT A RUBOUT
4052 012566 001000 BNE 34 'SKIP IF NOT
4053 012568 104401 001170 48: 'TYPE .QUES
4054 01256A 000763 BR 14 'CLEAR THE BUFFER AND LOOP
4055 01256C 111367 000044 34: 'MOVE (R3),94 'END THE CHARACTER
4056 01256E 104401 012670 TYPE .94 'CARE<br>THE
4057 01256F 122723 000013 CMPB 015,(R3)+ 'CHECK FOR RETURN
4058 012634 001356 BNE 21 'LOOK IF NOT RETURN
4059 012636 105063 177777 CLR #1(R3) 'CLEAR RETURN (THE 15)
4060 012638 104401 001172 TYPE .ALF 'TYPE A LINE FEED
4061 012642 0126D3 RESTORE R3 'RESTORE R3
4062 012644 0126D5 011646 MOV (SP)-,(SP) 'ADJUST THE STACK AND PUT ADDRESS OF THE
4063 012646 016646 MOV 4(SP),0(SP) 'FIRST ASCII CHARACTER ON IT
4064 012648 012676 012672 000004 MOV @TTYIN,4(SP) 'RETURN
4065 01264A 000002 RTI 'RETURN
4066 012670 000 .BYTE 0 'STORAGE FOR ASCII CHAR. TO TYPE
4067 012671 000 .BYTE 0 'TERMINATOR
4068 012672 000010 @TTYIN: .BLK 8 'RESERVE 8 BYTES FOR TTY INPUT
4069 012702 052336 052501 CONTL: .ASCIZ <U/15<12> 'CONTROL "U"
4070 012707 136 08507 000012 CONTLG: .ASCIZ <G/15<12> 'CONTROL "G"
4071 012714 005013 053523 MSR: .ASCIZ <15<12>/MSR = /
4072 012718 020075 000
4073 012725 040 047040 053505 @NEW: .ASCIZ / NEW = /
4074 012732 036440 000040
SB TNL APT COMMUNICATIONS ROUTINE

- TO REPORT FATAL ERROR
- TO TYPE A MESSAGE
- TO ONLY REPORT FATAL ERROR

MOV RO, -(SP)
MOV R1, -(SP)
MOV HFLAG
SHOULD TYPE A MESSAGE?

BEQ 50
IF NOT: BR

CNPB CAPPENV, LENV
OPERATING UNDER APT?

BEQ 30
IF NOT: BR

BITB CAPPSPool, LENV
SHOULD SPOOL MESSAGES?

BEQ 30
IF NOT: BR

MOV B0(SP), RO
GET MESSAGE ADDR.

ADD #2, (SP)
BUMP RETURN ADDR.

TST MSGTYPE
SEE IF DONE W/ LAST MISSION?

BEQ 10
IF NOT: HLT

MOV RO, MSGAD
PUT ADDR IN MAILBOX

BEQ 20
IF NOT: WAIT

MOV RO, MSGAD, RO
SUB START OF MESSAGE

ASR RO
GET MESSAGE LENGTH IN WORDS

MOV RO, MSGLG
PUT LENGTH IN MAILBOX

MOV 04, MSGTYPE
TELL APT TO TAKE MSG.

BR 91

MOV 80(SP), 41
PUT MSG ADDR IN JSR LINKAGE

ADD #2, (SP)
BUMP RETURN ADDRESS

MOV 177776, -(SP)
PUSH 177776 ON STACK

JSR PC, TYPE
CALL TYPE MACRO

-END-
<table>
<thead>
<tr>
<th>Address</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4121</td>
<td>013172 012601</td>
<td>MOV (SP)+, R1</td>
</tr>
<tr>
<td>4122</td>
<td>013174 012600</td>
<td>MOV (SP)+, R0</td>
</tr>
<tr>
<td>4123</td>
<td>013176 000207</td>
<td>RTS</td>
</tr>
<tr>
<td>4124</td>
<td>013200 000</td>
<td>IFMLG: .BYTE 0</td>
</tr>
<tr>
<td>4125</td>
<td>013201 000</td>
<td>IFLFLG: .BYTE 0</td>
</tr>
<tr>
<td>4126</td>
<td>013202 000</td>
<td>IFFLFLG: .BYTE 0</td>
</tr>
<tr>
<td>4127</td>
<td>013204 .EVEN</td>
<td>EVEN</td>
</tr>
<tr>
<td>4128</td>
<td>000200</td>
<td>APTSIZE=200</td>
</tr>
<tr>
<td>4129</td>
<td>000001</td>
<td>APTENV=001</td>
</tr>
<tr>
<td>4130</td>
<td>000100</td>
<td>APTSPool=100</td>
</tr>
<tr>
<td>4131</td>
<td>000040</td>
<td>APTCSUP=040</td>
</tr>
</tbody>
</table>

**Note:** The code appears to be used for setting up or managing flags and parameters in a program, possibly for error handling or debugging purposes.
.SBTL  ERROR HANDLER RUTINE

; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
; AND GO TO NTTY 'PE ON ERROR

; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; SW1=1  HALT ON ERROR
; SW1=1  INHIBIT ERROR TYPEOUTS
; SW1=1  BELL ON ERROR
; SW1=1  LOOP ON ERROR

; CALL ERROR N  ;ERROR=ENT AND N=ERROR ITEM NUMBER

ERROR:

013204  013204  104407
013206  105267  165671
70:   OKSMR  ;TEST FOR CHANGE IN SOFT-SWR
013208  105267  165671
013212  001775
013214  016777  165662  165720
013218  032777  002000  165710
013222  032777  002000  165710
013230  001402
013232  104401  001164
10:   INCB  ;SET THE ERROR FLAG
013236  005367  165850
01323A  016667  165850
013242  016667  165850
013246  162767  000002  165642
013254  177767  165636  165632
013262  032777  020000  165650
013270  001004
013272  004767  175634
013276  104401  001171
201:  JSA  ;GO TO USER ERROR ROUTINE
013302  122767  000001  165704
013306  001007
013312  116767  165376  000004
013320  004767  177430
214:  JSR  ;REPORT FATAL ERROR TO APT
013324  000000
013328  000000
224:  BRT  ;BR IF NONE
01332C  000777
013330  000777  165604
24:   TSB  ;BSMR
013334  100002
013338  000000
013340  104407
013344  105277  000100  165370
30:   OKSMR  ;TEST FOR CHANGE IN SOFT-SWR
013348  001402
013352  016716  165532
40:   BRT  ;BR IF NO
013356  005767  165600
44:   TSB  ;ISCAPE
013362  001402
013364  161716  165572
54:   MOV  ;FUDGE RETURN ADDRESS FOR ESCAPE
013370  022737  011410  000042
013370  022737  011410  000042
013370  022737  011410  000042
013376  001001
013400  000000
60:   RTI  ;RETURN
SBRTR SCME HANLER ROUTINE

.**. This routine controls the looping of subtests. It will increment
**and load the test number (#STNM) 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:
**#SWI4=1 LOOP ON TEST
**#SWI1=1 INHIBIT ITERATIONS
**#SWI9=1 LOOP ON ERROR
**#SWN8=1 LOOP ON TEST IN SWI<7:0>
**#CALL
**#SCME
**#SCME=IOT

$SCOE:
#CKB:
#TEST FOR CHANGE IN SOFT-SWR
#BIT: #BIT14, #BSWR
#LOOP ON PRESENT TEST?
#BNE #OVER
#YES IF SWI4=1

#STSTR: #BR #60
#IF RUNNING ON THE "XOR" TESTER CHANGE
#THIS INSTRUCTION TO A "NOP" (NOP=240)

#MOV: #6ERREVC, -(SP)
#SAVE THE CONTENTS OF THE ERROR VECTOR
#MOV: #56, #6ERREVC
#SET FOR TIMEOUT
#MOV: #ST, #6ERREVC
#TIME OUT ON XOR?
#MOV: (SP)+, #6ERREVC
#RESTORE THE ERROR VECTOR
#BR #SLVAD
#GO TO THE NEXT TEST

#CMP: (SP)+, (SP)+
#CLEAR THE STACK AFTER A TIME OUT
#MOV: (SP)+, #6ERREVC
#RESTORE THE ERROR VECTOR
#LOOP ON THE PRESENT TEST

#MOV: #6ERREVC, -(SP)
#SAVE THE CONTENTS OF THE ERROR VECTOR
#BIT: #BIT00, #BSWR
#LOOP ON SPEC. TEST?
#BEQ #21
#BR IF NO0

#CMP: #BSWR, #STSTNM
#ON THE RIGHT TEST? #SWI<7:0>
#BR IF YES

#TSTB: #ERFLG
#HAS AN ERROR OCCURRED?
#BNE #30
#BR IF NO

#CMP: #ERMAX, #ERFLG
#MAX. ERRORS FOR THIS TEST OCCURRED?
#BL #30
#BR IF NO

#BIT: #BIT09, #BSWR
#LOOP ON ERROR?
#BEQ #48
#BR IF #1804

#MOV: #LPERR, #LPAORD
#SET LOOP ADDRESS TO LAST SCOPE
#BR #OVER

#CLR: #ERFLG
#ZERO THE ERROR FLAG
#CLR: #TIMES
#CLEAR THE NUMBER OF ITERATIONS TO MAKE
#BR: #10
#ESCAPE TO THE NEXT TEST

#BIT: #BIT11, #BSWR
#INHIBIT ITERATIONS?
#BNE #10
#BR IF YES

#TST: #PASS
#IF FIRST PASS OF PROGRAM
#BEQ #10
#INHIBIT ITERATIONS
#INC #ICTNT
#INCREMENT ITERATION COUNT
#CMP: #TIMES, #ICTNT
#CHECK THE NUMBER OF ITERATIONS MADE
#BGE #OVER
#BR IF MORE ITERATION REQUIRED

#MOV: #ICNT, #TIMES
#REINITIALIZE THE ITERATION COUNTER
#MOV: #TIMES, #STSTNM
#SET NUMBER OF ITERATIONS TO DO
#MOV: #STSTNM, #TESTN
#SET TEST NUMBER IN APT MAILBOX
4242 013624 011667 165256  MOV (SP),#LPADR  ;SAVE SCOPE LOOP ADDRESS
4243 013630 011667 165254  MOV (SP),#LPERR  ;SAVE ERROR LOOP ADDRESS
4244 013634 005067 165322  CLR #ESCAPE  ;CLEAR THE ESCAPE FROM ERROR ADDRESS
4245 013644 112767 000001 165247  MOV  #1,#ERMAX  ;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4246 013646 016777 165230 165266 ics: MOV  #1STNM,#DISPLAY  ;DISPLAY TEST NUMBER
4247 013654 016716 165226  MOV #LPADR,(SP)  ;FUDGE RETURN ADDRESS
4248 013660 000002  RTI  ;FIXES PS
4249 013662 003720  #MXCNT: 2000.  ;MAX. NUMBER OF ITERATIONS
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

**This routine is used to change a 16-bit binary number to a 5-digit signed decimal (ASCII) number and type it, depending on whether the number is positive or negative a space or a minus sign will be typed before the first digit of the number. Leading zeros will always be replaced with spaces.**

**Call:**
1. MOV NUM, -(SP)  // Put the binary number on the stack
2. TYPSD  // Go to the routine

**TYPSD:**

MOV R0, -(SP)  // Push R0 on stack
MOV R1, -(SP)  // Push R1 on stack
MOV R2, -(SP)  // Push R2 on stack
MOV R3, -(SP)  // Push R3 on stack
MOV R5, -(SP)  // Push R5 on stack

MOV 420200, -(SP)  // Set blank switch and sign

MOV 20(SP), R5  // Get the input number

BPL 10  // BR if input is pos.

MOV B, 0, -(SP)  // Make the binary number pos.

MOV R0, 1, -(SP)  // Make the ASCII number neg.

MOV R1, 0, -(SP)  // Zero the constants index

MOVB 0, (SP)  // Setup the output pointer

MOV R0, 0, -(SP)  // Set the first character to blank pointer

MOV R0, R5  // Clear the BCD number

ADD R1, R5  // Add back the constant

MOV R0, R5  // Check if BCD digit = 0

JNZ 0  // Fall through if 0

INC R2  // Increase the BCD digit by 1

MOV R1, R5  // Form this BCD digit

MOV R1, R5  // Make the BCD digit ASCII

JNZ 90  // Print the BCD digit ASCII

MOV B, 0, -(SP)  // Yes-set the sign

MOV 1(SP), -(R3)  // Put the BCD digit ASCII

MOV R1, R2  // Make it a space if not already a digit

MOV R2, R3  // Put this character in the output buffer

MOV R1, R3  // Just incrementing

MOV R1, R3  // Check the table index

MOV R2, R3  // Go do the next digit

MOV R2, R3  // Go to exit

MOV R2, R3  // Get the LSD

MOV R2, R3  // Go change to ASCII

MOV R2, R3  // Was the LSD the first non-zero?

MOV R2, R3  // BR if no

MOV R2, R3  // Yes-set the sign for typing

MOV R2, R3  // Set the terminator

MOV R2, R3  // Pop stack into R5

MOV R2, R3  // Pop stack into R5

MOV R2, R3  // Pop stack into R5

MOV R2, R3  // Pop stack into R5
MDV (SP),R1 ;POP STACK INTO R1
MDV (SP),R0 ;POP STACK INTO R0
TYPE .IDBLK ;NOW TYPE THE NUMBER
MDV 2(SP),4(SP) ;ADJUST THE STACK
MDV (SP),(SP) ;RETURN TO USER

;DTBL: 10000.
;1000.
;100.
;10.
;IDBLK: .BLKW 4
.*SBTLT BINARY TO OCTAL (ASCII) AND TYPE

; THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT OCTAL (ASCII) NUMBER AND TYPE IT.
; TYPOS--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
; CALL:

; MOV: NURU-(SP)  ; NUMBER TO BE TYPED
; TYPOS: CALL FOR TYPEOUT
; .BYTE N  ; N+1 TO 6 FOR NUMBER OF DIGITS TO TYPE
; .BYTE M  ; M+1 OR 0
; 1; TYPE LEADING ZEROS
; 0; SUPPRESS LEADING ZEROS

; TYPOS: ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST TYPOS OR $TPOC
; CALL:

; MOV: NUM-(SP)  ; NUMBER TO BE TYPED
; TYPOS: CALL FOR TYPEOUT

; TYPOS: ENTER HERE FOR TYPEOUT OF A 16-BIT NUMBER
; CALL:

; MOV: NUM-(SP)  ; NUMBER TO BE TYPED
; TYPOS: CALL FOR TYPEOUT

; TYPOS: MOVB R(SP),(SP)  ; PICKUP THE MODE
; MOVB 1(SP), #OFILL  ; LOAD ZERO FILL SWITCH
; MOVB (SP), #MODE  ; NUMBER OF DIGITS TO TYPE
; ADD R0,(SP)  ; ADJUST RETURN ADDRESS
; BR #TPDN

; TYPOS: MOVB 01, #OFILL  ; SET THE ZERO FILL SWITCH
; MOVB #OFILL, #MODE  ; SET FOR SIX(6) DIGITS
; MOVB #MODE+1, #SP  ; SET THE ITERATION COUNT
; MOVB R3, -(SP)  ; SAVE R3
; MOVB R4, -(SP)  ; SAVE R4
; MOVB R5, -(SP)  ; SAVE R5

; MOV: MOVB #MODE+1, R4  ; GET THE NUMBER OF DIGITS TO TYPE
; MOV: MOVB R4  ; WEG R4
; ADD R6, R4  ; SUBTRACT IT FOR MAX. ALLOWED
; MOV: MOVB R4, #MODE  ; SAVE IT FOR USE
; MOVB #OFILL, R4  ; GET THE ZERO FILL SWITCH
; MOVB R4  ; PICKUP THE INPUT NUMBER
; MOV: MOVB R(SP), R5  ; CLEAR THE OUTPUT WORD
; ROL R5  ; ROTATE MSB INTO "C"
; MOV: MOVB R5, #R3  ; GO DO MSB
; MOV: MOVB R5, R3  ; GET LSB OF THIS DIGIT
; MOV: DECBO #MODE  ; TYPE THIS DIGIT?
; BPL #U  ; BR IF YES
; BIC 0177770, R3  ; GET RID OF JUNK
; BNE 40  ; TEST FOR 0
; ROL R4  ; SUPPRESS THIS 0?
; BNE 40  ; TEST FOR 0
; ROL R4  ; SUPPRESS THIS 0?
4373 014252 005204  40:  INC  R4  
4374 014254 052703 000060  DON'T SUPPRESS ANYMORE 0'S
4375 014260 052703 000040  MAKE THIS DIGIT ASCII
4376 014264 110367 000040  MAKE ASCII IF NOT ALREADY
4377 014264 110367 000040  SAVE FOR TYPING
4378 014270 104401 014330  TYPE ,8;
4379 014274 105367 000032  GO TYPE THIS DIGIT
4380 014300 003347  70:  DECB  $OCNT
4381 014302 002402  COUNT BY 1
4382 014304 005204  BF  20
4383 014306 000744  BR  IF MORE TO DO
4384 014310 012605  BR  IF DONE
4385 014312 012604  INSURE LAST DIGIT ISN'T A BLANK
4386 014314 012603  GO DO THE LAST DIGIT
4387 014316 016666 000002 000004  LAST DIGIT ISN'T A BLANK
4388 014324 012616  MOV 2(SP),A(SP)  SET THE STACK FOR RETURNING
4389 014324 012616  RTI  RETURN
4390 014330 0000  80:  .BYTE  0
4391 014331 0000  STORAGE FOR ASCII DIGIT
4392 014332 0000  .BYTE  0
4393 014333 0000  TERMINATOR FOR TYPE ROUTINE
4394 014334 0000  OCTAL DIGIT COUNTER
4395 014334 0000  ZERO FILL SWITCH
4396 014334 0000  NUMBER OF DIGITS TO TYPE
```
MAINDEC-11-OVDCV-C       MACY11 30A(1052)    12-SEP-84  15:41    PAGE 103
CVDCC.P11    12-SEP 84 08:95

TRAP DECORER

.SBTTL TRAP DECORER

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

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

; THIS IS USE TO HANDLE THE "GETPRI" MACRO

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

.SBTTL TRAP TABLE

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

<table>
<thead>
<tr>
<th>ROUTINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>--------</td>
</tr>
</tbody>
</table>

$TRAPAD:  .WORD  $TRAP2
        .TYPE   ; CALL = TYPE
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD
        .TYPE   ; CALL = TYPD

$TRAP0:  .WORD  TRAP+1(104401)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+2(104402)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+3(104403)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+4(104404)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+5(104405)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+6(104406)  GET SOFT-SWR SETTING
        .WORD  TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
        .WORD  TRAP+8(104408)  TTY TYPEOUT ROUTINE
        .WORD  TRAP+9(104409)  TTY TYPEOUT ROUTINE

.END
```
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASE</td>
<td>175610</td>
<td>956</td>
<td>997</td>
</tr>
<tr>
<td>ADD1</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDR4</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDP</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDRES</td>
<td>011112</td>
<td>3643*</td>
<td>3654*</td>
</tr>
<tr>
<td>ADDM</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDN</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDO</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDP</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDQ</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDS</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDT</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDU</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDR</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>ADDEVT</td>
<td>000000</td>
<td>956</td>
<td>962</td>
</tr>
<tr>
<td>ADDEVR</td>
<td>100000</td>
<td>1</td>
<td>956</td>
</tr>
<tr>
<td>EEN</td>
<td>000000</td>
<td>956</td>
<td>967</td>
</tr>
<tr>
<td>AFATAK</td>
<td>000000</td>
<td>956</td>
<td>959</td>
</tr>
<tr>
<td>APPD1</td>
<td>000000</td>
<td>956</td>
<td>984</td>
</tr>
<tr>
<td>APPD2</td>
<td>000000</td>
<td>956</td>
<td>958</td>
</tr>
<tr>
<td>APPD3</td>
<td>000000</td>
<td>956</td>
<td>991</td>
</tr>
<tr>
<td>APPD4</td>
<td>000000</td>
<td>956</td>
<td>994</td>
</tr>
<tr>
<td>APPD5</td>
<td>000000</td>
<td>956</td>
<td>978</td>
</tr>
<tr>
<td>APPE2</td>
<td>000000</td>
<td>956</td>
<td>986</td>
</tr>
<tr>
<td>APPE3</td>
<td>000000</td>
<td>956</td>
<td>989</td>
</tr>
<tr>
<td>APPE4</td>
<td>000000</td>
<td>956</td>
<td>989</td>
</tr>
<tr>
<td>APPE5</td>
<td>000000</td>
<td>956</td>
<td>964</td>
</tr>
<tr>
<td>APPE6</td>
<td>000000</td>
<td>956</td>
<td>965</td>
</tr>
<tr>
<td>APPE7</td>
<td>000000</td>
<td>956</td>
<td>958</td>
</tr>
<tr>
<td>APPTYP1</td>
<td>000000</td>
<td>956</td>
<td>979</td>
</tr>
<tr>
<td>APPTYP2</td>
<td>000000</td>
<td>956</td>
<td>987</td>
</tr>
<tr>
<td>APPTYP3</td>
<td>000000</td>
<td>956</td>
<td>950</td>
</tr>
<tr>
<td>APPTYP4</td>
<td>000000</td>
<td>956</td>
<td>993</td>
</tr>
<tr>
<td>APPTYP5</td>
<td>000000</td>
<td>956</td>
<td>961</td>
</tr>
<tr>
<td>APRTYP6</td>
<td>000000</td>
<td>956</td>
<td></td>
</tr>
<tr>
<td>APTC</td>
<td>000000</td>
<td>3643*</td>
<td>3668*</td>
</tr>
<tr>
<td>APTC1</td>
<td>000001</td>
<td>3643*</td>
<td>3668*</td>
</tr>
<tr>
<td>APTC2</td>
<td>000020</td>
<td>1067</td>
<td>4128*</td>
</tr>
<tr>
<td>APTC3</td>
<td>000010</td>
<td>3663</td>
<td>4087</td>
</tr>
<tr>
<td>APTC4</td>
<td>000010</td>
<td>3863</td>
<td>4089</td>
</tr>
<tr>
<td>ASP</td>
<td>000010</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ASSN</td>
<td>000000</td>
<td>956</td>
<td>969</td>
</tr>
<tr>
<td>ATEST</td>
<td>000000</td>
<td>956</td>
<td>960</td>
</tr>
<tr>
<td>AUTF</td>
<td>000000</td>
<td>956</td>
<td>963</td>
</tr>
<tr>
<td>AUSER</td>
<td>011110</td>
<td>1</td>
<td>956</td>
</tr>
<tr>
<td>AVECT1</td>
<td>000030</td>
<td>1</td>
<td>956</td>
</tr>
<tr>
<td>AVECT2</td>
<td>000000</td>
<td>956</td>
<td>996</td>
</tr>
<tr>
<td>BALD</td>
<td>007400</td>
<td>8590</td>
<td></td>
</tr>
<tr>
<td>SYMBOL</td>
<td>VALUE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>004026</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERNVEC</td>
<td>000004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EVENDO</td>
<td>000040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXITFL</td>
<td>004030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td>000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLG</td>
<td>010470</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FRENAB</td>
<td>020000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GNS</td>
<td>****** U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GSUR</td>
<td>014406</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOLD</td>
<td>007626</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOLDSC</td>
<td>010471</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HOMON</td>
<td>000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HT</td>
<td>000011</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>001272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILLMEN</td>
<td>000004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INITFL</td>
<td>011110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTFLA</td>
<td>010664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTSPV</td>
<td>010656</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOTVEC</td>
<td>000020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LF</td>
<td>000012</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOOP</td>
<td>002136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAINT</td>
<td>000004</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAINTJ</td>
<td>000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MASK</td>
<td>000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAX</td>
<td>006472</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MTYPE</td>
<td>011132</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEW</td>
<td>005664</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NDCOMP</td>
<td>011126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>007564</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>010572</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OK</td>
<td>011116</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OLD</td>
<td>005650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OREAR</td>
<td>004000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARITY</td>
<td>000020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PASS</td>
<td>004024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBDIALS</td>
<td>004000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBDIAL</td>
<td>010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBDIAL2</td>
<td>002000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBDIAL3</td>
<td>004000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBDIALS</td>
<td>100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PBR</td>
<td>000220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERA</td>
<td>010000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTER</td>
<td>177772</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDIRO</td>
<td>7480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PDNEVE</td>
<td>000240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRO</td>
<td>000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRL</td>
<td>000040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRL1</td>
<td>001000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRL1</td>
<td>001400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRL4</td>
<td>000220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRLS</td>
<td>000240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRLS</td>
<td>000300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PR7</td>
<td>000340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PS</td>
<td>177776</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSX</td>
<td>177776</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
</tr>
<tr>
<td>----</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>A1</td>
<td>109350</td>
<td>11010</td>
<td>11590</td>
</tr>
<tr>
<td>A2</td>
<td>12800</td>
<td>12900</td>
<td>12800</td>
</tr>
<tr>
<td>A3</td>
<td>14170</td>
<td>14250</td>
<td>14170</td>
</tr>
<tr>
<td>A4</td>
<td>15350</td>
<td>15400</td>
<td>15350</td>
</tr>
<tr>
<td>A5</td>
<td>18000</td>
<td>18100</td>
<td>18000</td>
</tr>
<tr>
<td>A6</td>
<td>19950</td>
<td>20050</td>
<td>19950</td>
</tr>
<tr>
<td>A7</td>
<td>20990</td>
<td>21080</td>
<td>20990</td>
</tr>
<tr>
<td>A8</td>
<td>22740</td>
<td>22770</td>
<td>22740</td>
</tr>
<tr>
<td>A9</td>
<td>23350</td>
<td>23500</td>
<td>23350</td>
</tr>
<tr>
<td>A10</td>
<td>29760</td>
<td>29870</td>
<td>29760</td>
</tr>
<tr>
<td>A11</td>
<td>31670</td>
<td>31730</td>
<td>31670</td>
</tr>
<tr>
<td>A12</td>
<td>33130</td>
<td>33260</td>
<td>33130</td>
</tr>
<tr>
<td>A13</td>
<td>37090</td>
<td>37170</td>
<td>37090</td>
</tr>
</tbody>
</table>

**INSC0** = 000300

**INSK0**

**INSK1** = 000110

**INSK2** = 000110

**INSK3** = 000110

**NULL** = 001154

**INMET** = 000001
<p>| ESCAPE | 10  | 7490 |
| EXIF   | 1803 | 3444 |
| EXIT   | 10  | 1220 | 1308 | 1611 | 1698 | 1809 | 1831 | 1907 | 1986 | 2058 | 2083 | 2110 | 2131 |
| GETPRI | 10  | 7490 | 10780 |
| GETSMR | 10  | 7490 |
| GPIMSD | 10  | 1168 |
| GPMPD  | 10  | 1145 |
| GPRL   | 10  | 1168 |
| HEADER | 10  | 1476 |
| IF     | 10  | 1592 |
| IFB    | 10  | 1560 |
| IF.ERR | 10  | 1754 |
| INCR   | 10  | 2335 |
| INCRU  | 10  | 3554 |
| LASTAD | 10  | 1103 |
| LET    | 10  | 1103 |
| LOOP   | 10  | 1706 |
| MSG    | 10  | 11304 |
| MULT   | 10  | 7490 |
| NEWST  | 10  | 7490 |
| POINTI | 10  | 7490 |
| POP    | 10  | 7490 |
| PRINTB | 10  | 7490 |
| PUSH   | 10  | 7490 |
| REPEAT | 10  | 1158 | 2004 | 3115 | 3617 |</p>
<table>
<thead>
<tr>
<th>AIND</th>
<th>10947</th>
<th>11695</th>
<th>12158</th>
<th>12178</th>
<th>12316</th>
<th>12478</th>
<th>12547</th>
<th>12633</th>
<th>12809</th>
<th>13035</th>
<th>13190</th>
<th>13345</th>
<th>13500</th>
<th>13655</th>
<th>13810</th>
<th>13965</th>
</tr>
</thead>
<tbody>
<tr>
<td>IFRANC</td>
<td>10950</td>
<td>11170</td>
<td>11170</td>
<td>11244</td>
<td>11258</td>
<td>11284</td>
<td>11284</td>
<td>11284</td>
<td>11284</td>
<td>11333</td>
<td>11400</td>
<td>11467</td>
<td>11535</td>
<td>11603</td>
<td>11671</td>
<td>11739</td>
</tr>
<tr>
<td>IFRANCE</td>
<td>13364</td>
<td>13435</td>
<td>13729</td>
<td>14215</td>
<td>14747</td>
<td>15279</td>
<td>15811</td>
<td>16343</td>
<td>16875</td>
<td>17407</td>
<td>17939</td>
<td>18472</td>
<td>19004</td>
<td>19536</td>
<td>19658</td>
<td>19779</td>
</tr>
<tr>
<td>ICALL</td>
<td>11090</td>
<td>11726</td>
<td>11767</td>
<td>11800</td>
<td>11833</td>
<td>11866</td>
<td>11899</td>
<td>11932</td>
<td>11965</td>
<td>11998</td>
<td>12031</td>
<td>12064</td>
<td>12097</td>
<td>12130</td>
<td>12163</td>
<td>12196</td>
</tr>
<tr>
<td>IEC</td>
<td>10950</td>
<td>11746</td>
<td>11787</td>
<td>12182</td>
<td>12418</td>
<td>12830</td>
<td>13242</td>
<td>13654</td>
<td>14066</td>
<td>14478</td>
<td>14890</td>
<td>15302</td>
<td>15714</td>
<td>16126</td>
<td>16538</td>
<td>16950</td>
</tr>
<tr>
<td>IECHEX</td>
<td>21260</td>
<td>21376</td>
<td>21692</td>
<td>22026</td>
<td>22349</td>
<td>22672</td>
<td>22995</td>
<td>23318</td>
<td>23641</td>
<td>23964</td>
<td>24287</td>
<td>24610</td>
<td>24933</td>
<td>25256</td>
<td>25579</td>
<td>25902</td>
</tr>
<tr>
<td>IECHEX</td>
<td>24922</td>
<td>25276</td>
<td>25630</td>
<td>25984</td>
<td>26338</td>
<td>26692</td>
<td>27046</td>
<td>27390</td>
<td>27744</td>
<td>28088</td>
<td>28432</td>
<td>28776</td>
<td>29120</td>
<td>29464</td>
<td>29808</td>
<td>30152</td>
</tr>
<tr>
<td>IOK01</td>
<td>31674</td>
<td>32094</td>
<td>32524</td>
<td>32954</td>
<td>33384</td>
<td>33814</td>
<td>34244</td>
<td>34674</td>
<td>35104</td>
<td>35534</td>
<td>35964</td>
<td>36394</td>
<td>36824</td>
<td>37254</td>
<td>37684</td>
<td>38114</td>
</tr>
<tr>
<td>IOK01</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
<td>56695</td>
</tr>
<tr>
<td>IOK02</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
<td>77350</td>
</tr>
<tr>
<td>IOK02</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
<td>99030</td>
</tr>
<tr>
<td>IOK03</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
<td>13550</td>
</tr>
<tr>
<td>IOK03</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
<td>217070</td>
</tr>
<tr>
<td>IOK04</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
<td>11710</td>
</tr>
<tr>
<td>IOK04</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
<td>14101</td>
</tr>
<tr>
<td>IOK04</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
<td>16955</td>
</tr>
<tr>
<td>MAINDEC-11</td>
<td>DVPDC-C</td>
<td>MACY11</td>
<td>30A(1052)</td>
<td>12-SEP-84 15:41 PAGE 132</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>--------</td>
<td>-----------</td>
<td>--------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVDVCC.P11</td>
<td>12-SEP-84 08:55</td>
<td>CROSS REFERENCE TABLE -- MACRO NAMES</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>36636</td>
<td>36596</td>
</tr>
<tr>
<td>36596</td>
<td>36593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11096</td>
<td>17234</td>
</tr>
<tr>
<td>17234</td>
<td>7679</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22829</td>
<td>25492</td>
</tr>
<tr>
<td>25492</td>
<td>28209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15956</td>
<td>16029</td>
</tr>
<tr>
<td>16029</td>
<td>19593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HENREY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAINDEC-11</th>
<th>DVPDC-C</th>
<th>MACY11</th>
<th>30A(1052)</th>
<th>12-SEP-84 15:41 PAGE 132</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDVCC.P11</td>
<td>12-SEP-84 08:55</td>
<td>CROSS REFERENCE TABLE -- MACRO NAMES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>36636</td>
<td>36596</td>
</tr>
<tr>
<td>36596</td>
<td>36593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11096</td>
<td>17234</td>
</tr>
<tr>
<td>17234</td>
<td>7679</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22829</td>
<td>25492</td>
</tr>
<tr>
<td>25492</td>
<td>28209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15956</td>
<td>16029</td>
</tr>
<tr>
<td>16029</td>
<td>19593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HENREY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MAINDEC-11</th>
<th>DVPDC-C</th>
<th>MACY11</th>
<th>30A(1052)</th>
<th>12-SEP-84 15:41 PAGE 132</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVDVCC.P11</td>
<td>12-SEP-84 08:55</td>
<td>CROSS REFERENCE TABLE -- MACRO NAMES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COMPA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>36636</td>
<td>36596</td>
</tr>
<tr>
<td>36596</td>
<td>36593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COUNT</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11096</td>
<td>17234</td>
</tr>
<tr>
<td>17234</td>
<td>7679</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DO</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22829</td>
<td>25492</td>
</tr>
<tr>
<td>25492</td>
<td>28209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELSE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15956</td>
<td>16029</td>
</tr>
<tr>
<td>16029</td>
<td>19593</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HENREY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
<tr>
<td>10546</td>
<td>10546</td>
</tr>
<tr>
<td>MAINMTL</td>
<td>DVDCC C</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>DVDCC P11</td>
<td>12-SEP-84</td>
</tr>
<tr>
<td>GETS</td>
<td>GETT</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>11101</td>
<td>11091</td>
</tr>
<tr>
<td>14230</td>
<td>14130</td>
</tr>
<tr>
<td>16170</td>
<td>16070</td>
</tr>
<tr>
<td>18080</td>
<td>18080</td>
</tr>
<tr>
<td>20700</td>
<td>20600</td>
</tr>
<tr>
<td>23950</td>
<td>23850</td>
</tr>
<tr>
<td>27920</td>
<td>27820</td>
</tr>
<tr>
<td>29660</td>
<td>29560</td>
</tr>
<tr>
<td>35150</td>
<td>35050</td>
</tr>
<tr>
<td>37090</td>
<td>37090</td>
</tr>
</tbody>
</table>
#TYPEO  10  4317
140CA  10
1170   10

ABS.  014416  000

ERRORS DETECTED:  0

CVDVCC.CVDVCC.SQL/CRF=CVDVCC.MLB/ML.SYSMAC.SML.CVDVCC.P11
RUN TIME: 77.77 6 SECONDS
RUN TIME RATIO: 246/162=1.5
CORE USED: 43K (45 PAGES)