M9301/9400

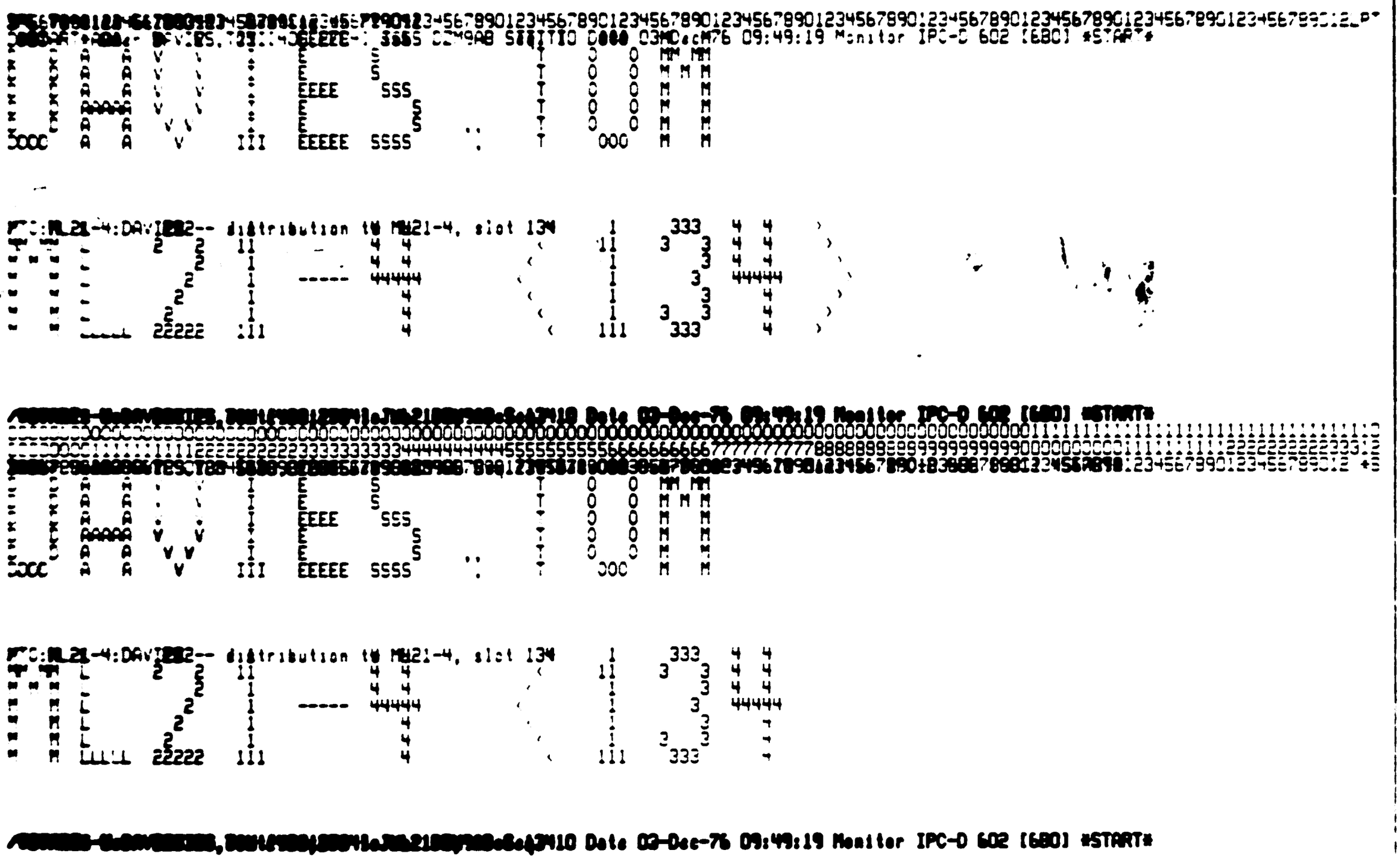
BOOTSTRAP
MD-11-DZM9A-B

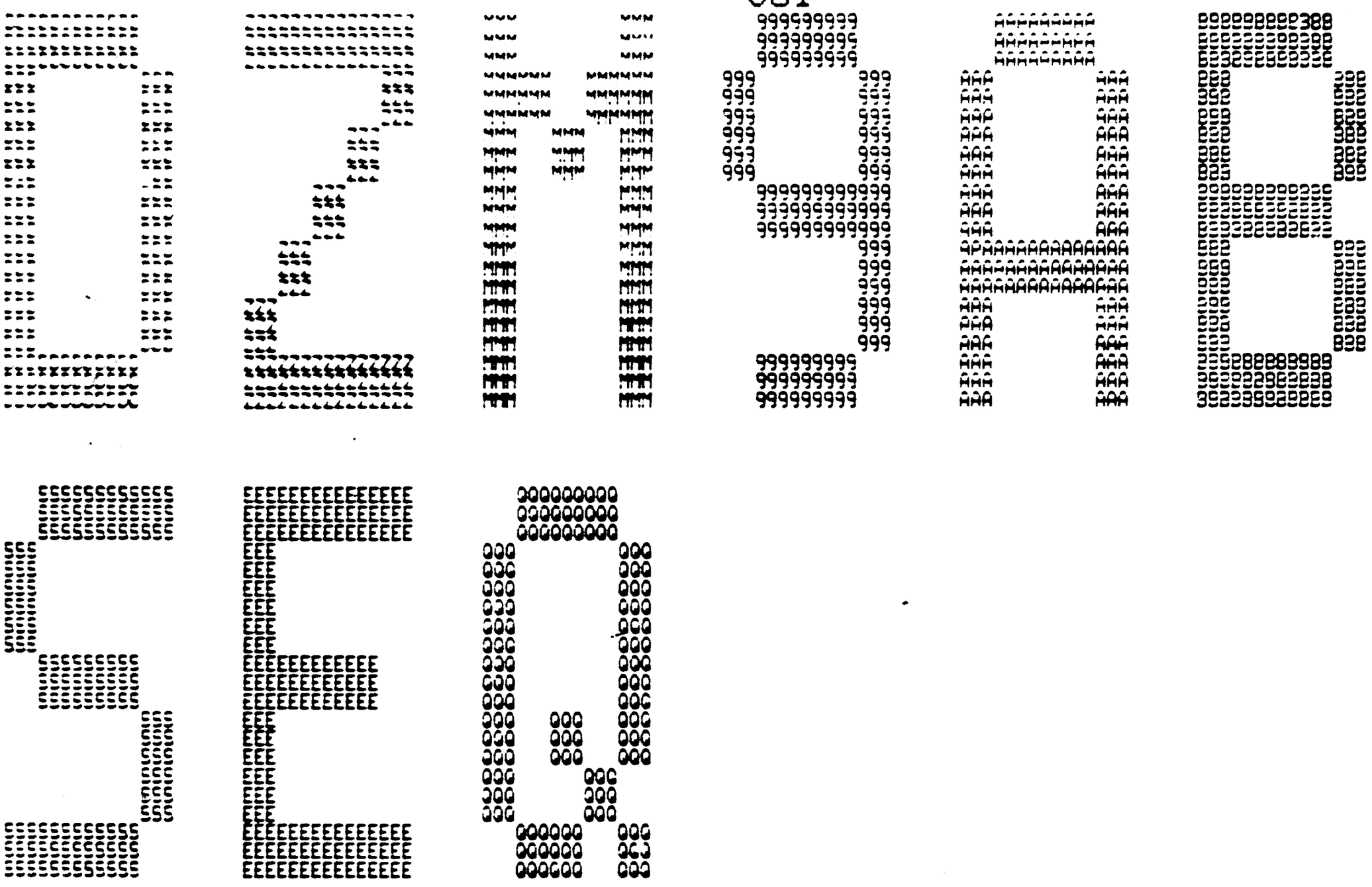
EP-DZM9A-B-DL-B

FICHE 1 OF 1

DEC 1976
digital
MADE IN USA

B01


Monitor IPC-D 602 [680] #START#
distribution to MH21-4, slot 134
Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] #START#



SYSP version 6(103344) Running on MTA041
START User DAVIES,TOM [400,2704] Job DZM9AB Seq. 10 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 (6BC) *START*
Request created: 03-Dec-76 09:26:29 /TO:ML21-4:DAVIES -- distribution to ML21-4, slot 134

[illegible] 09:49:28
File will be RENAMEd to SYS: protection

DOI

LPTSPL Version 6(100344) Running on MTA041
 START User DAVIES TOM [400,2704] Job DZM908 Seq. 10 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [683] *START*
Request created: 03-Dec-76 09:26:29   /TO:ML21-4:DAVIES -- distribution to ML21-4, slot 134

[illegible]09:49:28
File will be RENAMEd to DSK: protection

```
                    000000                          .REPT 0
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZM9A-B-D
PRODUCT NAME:	BOOTSTRAP/TERMINATOR (M9301, M9400)
PROGRAM DATE:	DECEMBER 1976
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHOR:	GLENN JOHNSON/VIN KALINOWSKI

```
                        ****************************************
                        *                                      *
                        * SUMMARY OF OPERATING INSTRUCTIONS    *
                        *                                      *
                        ****************************************


                        THE FOLLOWING PROCEDURE CAN BE USED TO RUN THIS DIAGNOSTIC
                        IN A DEVICE VERIFICATION MODE.  IF THE PROGRAM DOES NOT
                        RUN SUCCESSFULLY CONSULT THE FOLLOWING DOCUMENT FOR ASSISTANCE.

                        OPERATING PROCEDURE:

                                1.  LOAD THE PROGRAM USING NORMAL PROCEDURES.

                                2.  LOAD ADDRESS 200

                                3.  SET SWITCH REGISTER TO SELECT THE PROPER
                                    VERSION OF THE ROM UNDER TEST.
                                    (SEE INSTRUCTIONS IF A SOFTWARE SWITCH REGISTER
                                    IS TO BE USED.)

                                4.  PRESS START

                                5.  THE PROGRAM SHOULD TAKE ABOUT 1 SEC TO
                                    COMPLETE THE TEST AND PRINT: "END OF TEST".

                                6.  IF THE PROGRAM DOES NOT RUN AS DESCRIBED
                                    ABOVE, CONSULT THE FULL OPERATING INSTRUCTIONS
                                    WHICH FOLLOW.
```

1.0 GENERAL PROGRAM INFORMATION

1.1 PROGRAM PURPOSE

THIS DIAGNOSTIC PROGRAM IS INTENDED TO VERIFY THE
ROM CONTENTS OF THE ROM BOOTSTRAP MODULES. THE PROGRAM
COMPUTES AND CHECKS A CYCLIC REDUNDANCY CHARACTER
AND A LONGITUDINAL PARITY CHARACTER FOR THE CONTENTS
OF THE ROM STORAGE AVAILABLE IN AN M9301 OR M9400 MODULE.

A SEPARATE ROUTINE INCLUDED ALLOWS THE USER TO TYPE
THE CONTENTS OF THE ROM STORAGE ON THE TELETYPE AS
AN AID TO DEBUGGING.

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE

PDP/11 PROCESSOR
TELETYPE OR EQUIVALENT
4K OF MEMORY
M9301 OR M9400 MODULE

1.2.2 SOFTWARE

THIS PROGRAM IS WRITTEN TO BE RUN AS A STAND-ALONE PROGRAM.
HOWEVER, THE PROGRAM IS DESIGNED TO RUN UNDER AUTOMATED
PRODUCT TEST SYSTEM (APT) IN ALL THREE MODES.

THE PROGRAM CAN ALSO BE RUN UNDER THE ACT 11 MONITOR.

1.3 RELATED DOCUMENTS AND STANDARDS

DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS PROGRAMMING PRACTICES
DOCUMENT NO. 175-003-009-00

APT INTERFACE SPECIFICATION, REV. 13

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE, HOWEVER THE CPU IS ASSUMED TO BE FUNCTIONING

1.5 FAILURE ASSUMPTIONS

THE PROCESSOR IS ASSUMED TO BE FUNCTIONING PROPERLY.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOADING

USE NORMAL PROCEDURES FOR LOADING DIAGNOSTIC PROGRAMS.

2.1.2 NORMAL START

1. LOAD SOFTWARE SWITCH REGISTER (IF USED) TO SELECT THE ROM VERSION UNDER TEST. (SEE 2.3)
2. LOAD ADDRESS 200
3. SET HARDWARE SWITCH REGISTER (IF AVAILABLE) TO SELECT THE ROM VERSION UNDER TEST (SEE 2.3).
4. START

2.1.3 OPTIONAL START

THE OPTIONAL STARTING ADDRESS IS USED TO TYPE OUT THE CONTENTS OF THE ROM FOR USE IN VISUAL VERIFICATION OR AS A DEBUGGING TOOL.

USE THE SAME PROCEDURE AS A NORMAL START EXCEPT USE ADDRESS 210 IN STEP 2.

2.2 SPECIAL ENVIRONMENTS

THIS PROGRAM IS WRITTEN TO COMPLY WITH ALL THE REQUIREMENTS OF THE APT INTERFACE SPECIFICATION. IT WILL RUN UNDER APT IN EITHER QUICK VERIFY, PROGRAM OR RUN-TIME MODES.

THIS PROGRAM IS WRITTEN TO COMPLY WITH THE ACT11/XXDP INTERFACE REQUIREMENTS.

WHEN RUNNING IN ACT11 QUICK VERIFY OR XXDP CHAIN MODE (LOC. 42 NOT 0), OR APT QUICK VERIFY AND PROGRAM MODE THE PROGRAM ATTEMPTS TO RUN WITHOUT OPERATOR INTERVENTION OR SWITCH REGISTER SELECTION. THE COMPUTED CRC IS COMPARED AGAINST THE CRC FOR ALL KNOWN VERSIONS OF THE ROM BOOTSTRAP. WHEN A MATCH IS FOUND THE VERSION OF THE MODULE IS TYPED FOR VISUAL VERIFICATION.

2.3 PROGRAM OPTIONS

THE PROGRAM AUTOMATICALLY CHECKS FOR THE PRESENCE OF A HARDWARE SWITCH REGISTER. IF NO RESPONSE IS FOUND WHEN ADDRESSING THE HARDWARE SWR (177570), THE ADDRESS OF THE SOFTWARE SWR (176) IS SUBSTITUTED.

FOR PROCESSORS WITH NO HARDWARE SWITCH REGISTER, THE OPERATOR SHOULD SET THE DESIRED SWITCH VALUE IN LOCATION 176 BEFORE STARTING THE PROGRAM.

2.3.1 SWITCH SELECTION

THE SWITCH REGISTER (HARDWARE OR SOFTWARE) IS USED TO SELECT THE VERSION OF THE ROM BOOTSTRAP TO BE TESTED ACCORDING TO THE FOLLOWING TABLE.

SWR	MODULE VERSION
1	M9301-YA
2	M9301-YB
3	M9301-YC
4	M9400-YA (OR YC)
5	M9301-YF

IF THE CRC AND LPC FOR NEW VERSIONS ARE KNOWN BUT NOT IN THE ABOVE TABLE, SET THE SWITCH REGISTER TO ZERO AND ANSWER THE TELETYPE DIALOG.

TO DETERMINE THE CRC AND LPC FOR A NEW VERSION, START THE DIAGNOSTIC AT 200 WITH SWR=0. ANSWER 0 TO THE REQUESTS FOR THE LPC AND CRC. THE RESULTING MESSAGES WILL INDICATE THE CORRECT FUTURE RESPONSES FOR CRC AND LPC PROVIDED THE TEST IS RUN ON A KNOWN-GOOD MODULE.

2.3.2 TELETYPE DIALOG

SEVERAL QUESTIONS ARE ASKED OF THE OPERATOR IN ORDER TO OBTAIN SUFFICIENT INFORMATION FOR TESTING A ROM MODULE NOT PREVIOUSLY SUPPORTED IN THE DIAGNOSTIC. THE DIALOG IS INITIATED IF THE PROGRAM IS STARTED WITH THE SWR = 0. ALL RESPONSES ARE IN OCTAL AND TERMINATED BY A CARRIAGE RETURN.

ALL RESPONSES ARE CHECKED FOR VALID OCTAL NUMBERS. IF AN ILLEGAL CHARACTER IS TYPED, THE PROGRAM WILL TYPE A "?", CARRIAGE RETURN-LINE FEED AND AWAIT THE PROPER INPUT.

IF A MISTAKE IS NOTICED BEFORE THE CARRIAGE RETURN IS USED TO TERMINATE THE INPUT, A RUBOUT CAN BE USED TO DELETE MISTYPED INPUT.

1. TYPE CRC VALUE:
THIS REQUESTS THE VALUE OF THE CYCLIC REDUNDANCY CHECK
PREVIOUSLY CALCULATED FOR THIS VERSION OF THE ROM MODULE.
IT IS THE VALUE AGAINST WHICH THE UNIT UNDER TEST'S CRC
WILL BE COMPARED.

2. TYPE LPC VALUE:
THIS REQUESTS THE VALUE OF THE LONGITUDINAL PARITY CHECK
PREVIOUSLY CALCULATED FOR THIS VERSION OF THE ROM MODULE.
IT IS THE VALUE AGAINST WHICH THE UNIT UNDER TEST'S LPC
WILL BE COMPARED.

3. TYPE STARTING ADDR. OF 1ST ROM ADDR. SPACE:
THIS QUESTION REFERS TO THE FACT THAT THE ROM SPACE IN
A ROM BOOTSTRAP MODULE IS DIVIDED INTO 2 DISTINCT ADDRESS SPACES. TYPE
THE STARTING ADDRESS OF THE 1ST RANGE OF ADDRESSES. THE
STANDARD M9301 & M9400 BEGIN AT 173000.

4. TYPE LENGTH (BYTES) OF 1ST ROM ADDR. SPACE:
THIS REQUESTS THE LENGTH OF THE 1ST GROUP OF ROM ADDRESSES
IN BYTES. THE STANDARD M9301 & M9400 HAVE AN INITIAL ADDRESS SPACE
OF 1000 BYTES. IF THIS SECTION OF ADDRESSES IS NOT
USED BY THIS VERSION, ANSWER 0 TO THIS QUESTION.

5. TYPE STARTING ADDR. OF 2ND ROM ADDR. SPACE:
THIS REFERS TO THE FIRST ADDRESS IN THE SECOND DISTINCT
GROUP OF ROM ADDRESSES. THE RESPONSE FOR A STANDARD
M9301 & M9400 WOULD BE 165000.
6. TYPE LENGTH (BYTES) OF 2ND ROM ADDR. SPACE:
THIS REQUESTS THE LENGTH OF THE 2ND GROUP OF ROM ADDRESSES
IN BYTES. THE STANDARD M9301 & M9400 HAVE A SECOND ADDRESS SPACE
OF 1000 BYTES. IF THIS SECTION OF ADDRESSES IS NOT USED
BY THIS VERSION, ANSWER 0 TO THIS QUESTION.

2.4 EXECUTION TIMES

THE DIAGNOSTIC COMPLETES 1 PASS IN LESS THAN 1 SEC.
ONCE THE INPUT DIALOG HAS BEEN COMPLETED.
THE PROGRAM WILL HALT UPON COMPLETION; HOWEVER, IF RUNNING
UNDER AFT THE PROGRAM WILL CYCLE CONTINUOUSLY.

```
                                3.0     ERROR INFORMATION
                                        ----- -----------

                                        WHEN THE DIAGNOSTIC DETECTS A DISCREPANCY BETWEEN THE
                                        EXPECTED AND COMPUTED CRC OR LPC BOTH ARE PRINTED
                                        ON THE TELETYPE.

                                        ANY DISCREPANCY IN THE LPC CAN ASSIST IN ISOLATING THE
                                        PROBLEM TO THE ROM IC.

                                        UNDER APT THE ERROR IS INDICATED BY DEPOSITING ERROR INFORMATION
                                        IN THE APT MAILBOX BEFORE HALTING.

                                4.0     PROGRESS REPORTS
                                        -------- -------

                                        AT THE END OF EACH PASS THE PROGRAM INCREMENTS TO THE
                                        LOCATION $PASS WHICH IS IN THE APT MAILBOX. THIS
                                        LOCATION WILL ALWAYS CONTAIN THE NUMBER OF PASSES
                                        COMPLETED.  $PASS IS RESET WITH EVERY RESTART.

                                        ADDITIONALLY, THE MESSAGE "END OF TEST" IS PRINTED ON THE CONSOLE
                                        TELETYPE AFTER EACH PASS.  NORMALLY ONLY ONE PASS NEEDS
                                        TO RUN TO VERIFY THE MODULE.

                                5.0     TROUBLE SHOOTING
                                        ------- --------

                                        THE ALGORITHM FOR COMPUTING THE CRC IS THE SAME AS THAT
                                        USED ON 9-TRACK MAGNETIC TAPE WITH ODD PARITY.  THE
                                        CRC IS CALCULATED ON A BYTE-BY-BYTE BASIS.  WHILE THE
                                        ALGORITHM IS SUCCESSFUL IN DETECTING MULTIPLE ERRORS, ITS
                                        USE AS A DEBUGGING AID IS LIMITED.

                                        THE LPC IS CALCULATED BY ASSEMBLING THE XOR OF EVERY
                                        WORD IN THE ROM.  WHILE ONLY USEFUL IN CATCHING AN
                                        ODD NUMBER OF ERRORS IN EACH BIT POSITION, IT IS VERY
                                        USEFUL IN ISOLATING THE PROBLEM TO A CHIP.  BY LOCATING WHICH
                                        BIT POSITIONS ARE IN DISCREPANCY, THE CORRESPONDING ROM
                                        CHIPS CAN BE ISOLATED.

                                        IF NO OTHER CLUES CAN BE OBTAINED, START THE PROGRAM
                                        AT 210 AND COMPARE THE PRINTOUT OF THE CODE WITH THE
                                        LISTING FOR THE VERSION BEING TESTED.

                                6.0     LISTING
                                        -------



                                        .ENDR
```

```
                                .ENABLE ABS
                                .LIST   ME
                                .NLIST  MC,MD,CND
000000                          $SWR=0
                                .SBTTL  BASIC DEFINITIONS

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

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

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

                                ;*PRIORITY LEVEL DEFINITIONS
000000                          PR0=    0               ;;PRIORITY LEVEL 0
000040                          PR1=    40              ;;PRIORITY LEVEL 1
000100                          PR2=    100             ;;PRIORITY LEVEL 2
000140                          PR3=    140             ;;PRIORITY LEVEL 3
000200                          PR4=    200             ;;PRIORITY LEVEL 4
000240                          PR5=    240             ;;PRIORITY LEVEL 5
000300                          PR6=    300             ;;PRIORITY LEVEL 6
000340                          PR7=    340             ;;PRIORITY LEVEL 7

                                ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                          SW15=   100000
040000                          SW14=   40000
020000                          SW13=   20000
010000                          SW12=   10000
004000                          SW11=   4000
002000                          SW10=   2000
001000                          SW09=   1000
000400                          SW08=   400
000200                          SW07=   200
```

BASIC DEFINITIONS

```
                000100          SW06=   100
                000040          SW05=   40
                000020          SW04=   20
                000010          SW03=   10
                000004          SW02=   4
                000002          SW01=   2
                000001          SW00=   1
                                .EQUIV  SW09,SW9
                                .EQUIV  SW08,SW8
                                .EQUIV  SW07,SW7
                                .EQUIV  SW06,SW6
                                .EQUIV  SW05,SW5
                                .EQUIV  SW04,SW4
                                .EQUIV  SW03,SW3
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

                                ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
                100000          BIT15=  100000
                040000          BIT14=  40000
                020000          BIT13=  20000
                010000          BIT12=  10000
                004000          BIT11=  4000
                002000          BIT10=  2000
                001000          BIT09=  1000
                000400          BIT08=  400
                000200          BIT07=  200
                000100          BIT06=  100
                000040          BIT05=  40
                000020          BIT04=  20
                000010          BIT03=  10
                000004          BIT02=  4
                000002          BIT01=  2
                000001          BIT00=  1
                                .EQUIV  BIT09,BIT9
                                .EQUIV  BIT08,BIT8
                                .EQUIV  BIT07,BIT7
                                .EQUIV  BIT06,BIT6
                                .EQUIV  BIT05,BIT5
                                .EQUIV  BIT04,BIT4
                                .EQUIV  BIT03,BIT3
                                .EQUIV  BIT02,BIT2
                                .EQUIV  BIT01,BIT1
                                .EQUIV  BIT00,BIT0

                                ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                000014          TBITVEC=14              ;;"T" BIT
                000014          TRTVEC= 14              ;;TRACE TRAP
                000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                000024          PWRVEC= 24              ;;POWER FAIL
                000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                000034          TRAPVEC=34              ;;"TRAP" TRAP
```

```
.MAIN.  MACY11 27(1006)  27-SEP-76  14:46  PAGE 10
DZM9A8.P11    27-SEP-76 14:43            BASIC DEFINITIONS

    450           000060                 TKVEC=  60              ;;TTY KEYBOARD VECTOR
    451           000064                 TPVEC=  64              ;;TTY PRINTER VECTOR
    452           000240                 PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
    453           000000                         .=0
    454                                  .SBTTL  TRAP CATCHER
    455
    456           000000                         .=0
    457                                  ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
    458                                  ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
    459                                  ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
    460           000174                         .=174
    461  000174   000000                 DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
    462  000176   000000                 SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
    463           000200                 .=200
    464  000200   005067  000624                 CLR     TYPOUT
    465  000204   000167  000666                 JMP     START
    466  000210   012767  000001  000612         MOV     #1,TYPOUT
    467  000216   000167  000654                 JMP     START
    468
    469           177776                 PS=177776
    470           000034                 TRAPVEC=34
    471
    472           001000                 .=1000
    473  001000   177570                 SWR:    177570
    474  001002   177570                 DISPLAY:        177570
    475  001004   173000                 ROMSA1: 173000
    476  001006   001000                 DATLN1: 512.
    477  001010   165000                 ROMSA2: 165000
    478  001012   001000                 DATLN2: 512.
    479  001014   000000                 XORS:   0
    480  001016   000000                 EXCRC:  0
    481  001020   000000                 EXLPC:  0
    482  001022   000000                 ACTCRC: 0
    483  001024   000000                 ACTLPC: 0
    484  001026   000000                 PARCNT: 0
    485  001030   000000                 TYPOUT: 0
    486
    487
    488                                  .SBTTL  ACT11 HOOKS
    489
    490                                  ;;*****************************************************************
    491           001032                 ;HOOKS REQUIRED BY ACT11
    492           000046                         $SVPC=.                 ;SAVE PC
    493  000046   002042                         .=46
    494           000052                         $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
    495  000052   000000                         .=52
    496           001032                         .WORD   0               ;;2)SET LOC.52 TO ZERO
    497                                          .=$SVPC                 ;; RESTORE PC
    498                                  .SBTTL  APT PARAMETER BLOCK
    499
    500                                  ;;*****************************************************************
    501                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
    502                                  ;;*****************************************************************
    503           001032                         .$X=.   ;;SAVE CURRENT LOCATION
    504           000024                         .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
    505  000024   000200                         200     ;;FOR APT START UP
    506           000044                         .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
```

```
                                        $APTHDR ;;POINT TO APT HEADER BLOCK
001036  001036                          .=.$X   ;;RESET LOCATION COUNTER
                                ;;**********************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001036                          $APTHD:
001036  000000                 $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001040  001046                 $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001042  000002                 $TSTM:  .WORD   2       ;;RUN TIM OF LONGEST TEST
001044  000002                 $PASTM: .WORD   2       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001046  000000                 $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001050  000014                         .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;**********************************************************************
                                .EVEN
001046                          $MAIL:                  ;;APT MAILBOX
001046  000000                 $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
001050  000000                 $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
001052  000000                 $TESTN: .WORD   ATESTN  ;;TEST NUMBER
001054  000000                 $PASS:  .WORD   APASS   ;;PASS COUNT
001056  000000                 $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001060  000000                 $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001062  000000                 $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001064  000000                 $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001066                          $ETABLE:                ;;APT ENVIRONMENT TABLE
001066  000                    $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001067  000                    $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001070  000000                 $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001072  000000                 $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001074  000000                 $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                                       BITS 15-11=CPU TYPE
                                                               11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                                               11/70=06,PDQ=07,Q=10
                                                       BIT 10=REAL TIME CLOCK
                                                       BIT  9=FLOATING POINT PROCESSOR
                                                       BIT  8=MEMORY MANAGEMENT
001076                          $ETEND:
                                .MEXIT

001076  005067  177746          START:  CLR     $FATAL                  ;CLEAR ERROR NO.
001102  005067  177740                  CLR     $MSGTYP                 ;CLEAR MESSAGE TYPE (APT)
001106  012767  000001  177736         MOV     #1,$TESTN               ;SET TEST NO.
                                .SBTTL  INITIALIZE THE COMMON TAGS
001114  012706  000500                 MOV     #500,SP         ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
001120  012737  004552  000034         MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001126  012737  000340  000036         MOV     #340,@#TRAPVEC+2;LEVEL 7
001134  012737  004374  000024         MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001142  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;LEVEL 7
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001150  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
001154  012737  001210  000004         MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
001162  012767  177570  177610         MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
```

INITIALIZE THE COMMON TAGS

```
                                                MOV     @DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                                                CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
                                                BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
                                                BR      65$             ;;BRANCH IF NO TIMEOUT
                                        64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
                                                RTI
                                        65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
                                                MOV     #DISPREG,DISPLAY
                                        66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                                CLR     $PASS           ;;CLEAR PASS COUNT
                                                BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
                                                BEQ     67$             ;;YES,USE NON-APT SWITCH
                                                MOV     #SSWREG,SWR     ;;NO,USE APT SWITCH REGISTER
                                        67$:
                                                CMP     42,#SENDAD              ;ACT AUTO MODE?
                                                BEQ     RESTRT                  ;YIF SO: BR
                                                TYPE
                                                TITL

                                        RESTRT: MOV     @SWR,R0                 ;GET SWR
                                                BEQ     GETIN                   ;IF ZERO: GET INPUT
                                        ST2:    ASL     R0
                                                MOV     TXLPC(R0),EXLPC         ;FETCH EXPECT. LPC
                                                MOV     TXCRC(R0),EXCRC         ;FETCH EXPECTED CRC
                                                MOV     TDLN1(R0),DATLN1        ;FETCH 1ST LENGTH
                                                MOV     TRMSA1(R0),ROMSA1       ;FETCH 1ST STARTING ADDR.
                                                MOV     TDLN2(R0),DATLN2                ;FETCH 2ND LENGTH
                                                MOV     TRMSA2(R0),ROMSA2       ;FETCH 2ND STARTING ADDR
                                                BR      CHECK                   ;GO COMPUTE
                                        GETIN:  TST     42                      ;UNDER ACT AUT ACCEPT?
                                                BNE     CHECK                   ;IF SO; BR USE DEFAULT PARAMETERS
                                                CMPB    #1,$ENV                 ;UNDER APT?
                                                BEQ     CHECK                   ;IF SO: BR
                                                TST     TYPOUT                  ;ROM TYPE OPTION
                                                BNE     GET2                    ;IF SO:  BR
                                                TYPE
                                                GETCRC
                                                RDOCT
                                                MOV     (SP)+,EXCRC             ;STORE EXPECT. CRC
                                                TYPE                            ;TYPE LPC INPUT REQUEST
                                                GETLPC
                                                RDOCT
                                                MOV     (SP)+,EXLPC             ;STORE EXPECTED LPC
                                        GET2:   TYPE
                                                SA1                             ;REQUEST 1ST ADDRESS SPACE
                                                RDOCT                           ;INPUT SA
                                                MOV     (SP)+,ROMSA1
                                                TYPE                            ;REQUEST LENGTH OF 1ST ADDR. SPACE
                                                SIZE1
                                                RDOCT                           ;INPUT LENGTH
                                                MOV     (SP)+,DATLN1
                                                TYPE                            ;REQUEST START ADDR. FOR 2ND SPACE
                                                SA2
                                                RDOCT                           ;INPUT SA
```

```
                                INITIALIZE THE COMMON TAGS
                                ;
001454 012667 177330            MOV     (SP)+,ROMSA2
001460 104401                   TYPE                            ;REQUEST LENGTH OF 2ND SPACE
001462 005465                   SIZE2
001464 104403                   RDOCT                           ;INPUT LENGTH
001466 012667 177320            MOV     (SP)+,DATLN2
001472 005767 177332    CHECK:  TST     TYPOUT                  ;IS TYPOUT REQUESTED?
001476 001402                   BEQ     1$                      ;BRANCH IF NOT
001500 000167 000660            JMP     TYPROM                  ;GO TYPE OUT ROM
001504 005067 177312    1$:     CLR     ACTCRC                  ;CLEAR STORAGE FOR ACTUAL CRC
001510 005067 177310            CLR     ACTLPC                  ;CLEAR STORAGE FOR ACTUAL LPC
001514 016700 177266            MOV     DATLN1,R0               ;SET LENGTH OF 1ST ROM SPACE
001520 001413                   BEQ     CH0                     ;IF NO VERSION SELECTED:  BR
001522 016701 177256            MOV     ROMSA1,R1               ;POINT TO START OF 1ST ROM SPACE
001526 004767 000324            JSR     PC,CRC                  ;COMPUTE FIRST HALF OF CRC
001532 016701 177246            MOV     ROMSA1,R1               ;POINT TO START OF 1ST ROM ADDRS.
001536 016700 177244            MOV     DATLN1,R0               ;SET LENGTH OF 1ST ROM ADDRS.
001542 006200                   ASR     R0                      ;CONVERT TO WORDS
001544 004767 000556            JSR     PC,LPC                  ;COMPUTE FIRST HALF OF CRC
001550 016701 177234    CH0:    MOV     ROMSA2,R1               ;POINT TO 2ND ROM ADDRS.
001554 016700 177232            MOV     DATLN2,R0               ;SET LENGTH OF 2ND ROM ADDRS.
001560 001422                   BEQ     CH1                     ;BR IF THIS SPACE NOT USED
001562 004767 000270            JSR     PC,CRC                  ;COMPUTE REMAINDER OF CRC
001566 016701 177216            MOV     ROMSA2,R1               ;POINT TO START OF 2ND ROM ADDRS.
001572 016700 177214            MOV     DATLN2,R0               ;SET LENGTH OF 2ND ROM ADDRS.
001576 006200                   ASR     R0                      ;CONVERT TO WORDS
001600 004767 000522            JSR     PC,LPC                  ;COMPUTE REMAINDER OF LPC
001604 122767 000001 177254     CMPB    #1,$ENV                 ;UNDER APT?
001612 001403                   BEQ     1$                      ;IF SO:  BR
001614 005767 176222            TST     42                      ;UNDER ACT AUTO ACCEPT?
001620 001402                   BEQ     CH1                     ;IF NOT:  BR
001622 000167 000656    1$:     JMP     AUTACT
001626 026767 177164 177166 CH1:CMP     EXCRC,ACTCRC            ;COMPUTED = EXPECTED ?
001634 001431                   BEQ     CK1                     ;IF SO:  BR

001636 104401                   TYPE                            ;TYPE CRC ERROR MESG.
001640 005111                   EXCRMG
001642 016746 177150            MOV     EXCRC,-(SP)             ;PUT EXPECT CRC ON STACK
001646 104402                   TYPOC                           ;TYPE EXPECTED CRC
001650 104401                   TYPE                            ;TYPE ACTUAL CRC MESSG
001652 005160                   ACCRMG
001654 016746 177142            MOV     ACTCRC,-(SP)            ;PUT ACTUAL CRC ON STACK
001660 104402                   TYPOC                           ;TYPE ACTUAL CRC
001662 026727 176154 002042     CMP     42,#SENDAD              ;UNDER ACT AUTO MODE?
001670 001404                   BEQ     1$                      ;IF SO:  BR
001672 122767 000001 177166     CMPB    #1,$ENV                 ;UNDER APT?
001700 001007                   BNE     CK1                     ;IF NOT:  BR
001702 012767 000002 177140 1$: MOV     #2,$FATAL               ;MOVE TO MAILBOX ERROR NO.  ****  2  ****
001710 012767 000001 177130     MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
001716 000000                   HALT                            ;CRC ERROR

001720 026767 177100 177072 CK1:CMP     ACTLPC,EXLPC            ;COMPARE EXPT. LPC=ACTUAL LPC
001726 001431                   BEQ     CK2                     ;IF SO:  BR
001730 104401                   TYPE                            ;TYPE LPC ERROR MESSG.
001732 005134                   EXLPMG
001734 016746 177060            MOV     EXLPC,-(SP)             ;PUT EXPECTED LPC ON STACK
001740 104402                   TYPOC                           ;TYPE EXPECTED LPC
```

INITIALIZE THE COMMON TAGS

```
   674  001742  104401                          TYPE                            ;TYPE ACTUAL LPC MESSG.
   675  001744  005203                          ACLPMG
   676  001746  016746  177052                  MOV     ACTLPC,-(SP)            ;PUT ACTUAL LPC ON STACK
   677  001752  104402                          TYPOC                           ;TYPE ACTUAL LPC
   678  001754  026727  176062  002042          CMP     42,#SENDAD              ;UNDER ACT AUTO MODE?
   679  001762  001404                          BEQ     1$                      ;IF SO:  BR
   680  001764  122767  000001  177074          CMPB    #1,$ENV                 ;UNDER APT?
   681  001772  001007                          BNE     CK2                     ;IF NOT:  BR
   682  001774  012767  000003  177046  1$:     MOV     #3,$FATAL               ;MOVE TO MAILBOX ERROR NO.  ****  3  ****
   683  002002  012767  000001  177036          MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
   684  002010  000000                          HALT                            ;LPC ERROR
   685
   686  002012  026727  176024  002042  CK2:    CMP     42,#SENDAD              ;ACT AUTO ACCEPT?
   687  002020  001402                          BEQ     1$                      ;IF SO:  BR
   688  002022  104401                          TYPE                            ;TYPE END OF TEST
   689  002024  005226                          EOTST
   690  002026  005267  177022          1$:     INC     $PASS                   ;BUMP PASS COUNT
   691  002032  013700  000042                  MOV     @#42,R0                 ;CHECK APT
   692  002036  001405                          BEQ     GOAGIN                  ;KEEP GOING
   693  002040  000005                          RESET
   694  002042  004710                  SENDAD: JSR     PC,(R0)                 ;ACT HOOKS
   695  002044  000240                          NOP
   696  002046  000240                          NOP
   697  002050  000240                          NOP
   698  002052  000167  177216          GOAGIN: JMP     RESTRT          ;DO AGAIN
   699
   700  002056  016767  176740  176730  CRC:    MOV     ACTCRC,XORS
   701  002064  111104          CL0:    MOVB    (R1),R4                 ;GET CHAR.
   702  002066  022701  173024                  CMP     #173024,R1              ;LOCATION EFFECTED BY SWITCHES
   703  002072  001004                          BNE     CL3                     ;IF NOT:  BR
   704  002074  005300                          DEC     R0                      ;FIX COUNTERS
   705  002076  005300                          DEC     R0
   706  002100  005721                          TST     (R1)+                   ;FIX POINTER
   707  002102  000770                          BR      CL0                     ;CONTINUE
   708  002104  004767  000114          CL3:    JSR     PC,PARITY               ;GO GET PARITY
   709  002110  004767  000166                  JSR     PC,XOR                  ;XOR CHAR
   710  002114  000241                          CLC
   711  002116  006004                          ROR     R4                      ;ROTATE 1 POS. RIGHT
   712  002120  103001                          BCC     CL2                     ;IF NO CARRY: BR
   713  002122  052704  000400                  BIS     #400,R4                 ;SET BIT NINE
   714  002126  000241                          CLC
   715  002130  010405          CL1:    MOV     R4,R5                   ;SAVE CHAR
   716  002132  042705  177703                  BIC     #177703,R5
   717  002136  005105                          COM     R5
   718  002140  042705  177703                  BIC     #177703,R5
   719  002144  042704  000074                  BIC     #74,R4
   720  002150  050504                          BIS     R5,R4
   721  002152  010467  176636          CL2:    MOV     R4,XORS
   722  002156  005300                          DEC     R0
   723  002160  001402                          BEQ     CLLAST                  ;IF LAST CAR.:  BR
   724  002162  000167  177676                  JMP     CL0                     ;GET NEXT CHAR.
   725  002166  016704  176622          CLLAST: MOV     XORS,R4
   726  002172  005167  176616                  COM     XORS
   727  002176  042767  177050  176610          BIC     #177050,XORS
   728  002204  042704  177727                  BIC     #177727,R4              ;COMPLEMENT ALL BUT BITS 3 & 5
   729  002210  050467  176600                  BIS     R4,XORS
```

INITIALIZE THE COMMON TAGS

```
                                                MOV     XORS,ACTCRC
                                                RTS     PC
002222  005067  176576          PARITY:       CLR     PARCNT          ;CLEAR BIT COUNTER
002226  012703  000010                        MOV     #10,R3          ;SET NO. OF BITS
002232  032704  000001          CLP0:         BIT     #1,R4           ;SEE IF ONE BIT
002236  001402                                BEQ     CLP1            ;IF NOT:  BR
002240  005267  176560                        INC     PARCNT          ;BUMP COUNTER
002244  000241                  CLP1:         CLC
002246  006004                                ROR     R4              ;ROTATE TO NEXT BIT
002250  005303                                DEC     R3
002252  001361                                BNE     CLP0            ;CONTINUE FOR ALL BITS
002254  112104                                MOVB    (R1)+,R4
002256  042704  177400                        BIC     #177400,R4
002262  032767  000001  176534                BIT     #1,PARCNT       ;SEE IF ODD # OF ONE BITS
002270  001002                                BNE     CLP2            ;IF SO:  BR
002272  052704  000400                        BIS     #400,R4         ;SET PARITY BIT
002276  000207                  CLP2:         RTS     PC              ;EXIT

002300  010446                  XOR:          MOV     R4,-(SP)        ;XOR SUBROUTINE: R4 WITH XORS
002302  046716  176520                        BIC     XORS,(SP)
002306  042467  176514                        BIC     R4,XORS
002312  052667  176512                        BIS     (SP)+,XORS
002316  016704  176470                        MOV     XORS,R4
002322  000207                                RTS     PC

002324  016767  176472  176460  LPC:          MOV     ACTLPC,XORS
002332  012104                  LPC1:         MOV     (R1)+,R4
002334  022701  173026                        CMP     #173026,R1      ;LOCATION EFFECTED BY SWITCHES
002340  001402                                BEQ     LPC2            ;IF SO:  SKIP LOC. BY BRANCHING
002342  004767  177732                        JSR     PC,XOR
002346  005300                  LPC2:         DEC     R0
002350  001370                                BNE     LPC1
002352  016767  176434  176442                MOV     XORS,ACTLPC
002360  000207                                RTS     PC

002362  104401                  TYPROM:       TYPE                    ;TYPE HEADER
002364  005553                                TYPHDR
002366  016700  176410                        MOV     ROMSA1,R0       ;POINT AT 1ST ROM SPACE
002372  016701  176406                        MOV     DATLN1,R1       ;PUT LENGTH IN R1
002376  006201                                ASR     R1              ;CONVERT TO WORDS
002400  001402                                BEQ     TYPR1           ;BRANCH IF 1ST ROM SPACE NOT USED
002402  004767  000026                        JSR     PC,TYP          ;GO TYPE 1ST ADDR. SPACE
002406  016700  176374          TYPR1:        MOV     ROMSA2,R0       ;POINT TO 2ND ADDR. SPACE
002412  016701  176372                        MOV     DATLN2,R1       ;PUT LENGTH IN R1
002416  006201                                ASR     R1              ;CONVERT TO WORDS
002420  001402                                BEQ     ENDCT           ;BR IF 2ND ADDR. SPACE NOT USED
002422  004767  000006                        JSR     PC,TYP          ;GO TYPE 2ND ADDR. SPACE
002426  104403                  ENDCT:        TYPE
002430  005226                                EOTST
002432  000000                                HALT

002434  104401                  TYP:          TYPE
002436  005545                                CARLF
002440  000403                                BR      TYP3
002442  032700  000003          TYP0:         BIT     #3,R0           ;ADDRESS MULTIPLE OF 4?
002446  001006                                BNE     TYP2            ;IF NOT:  BR
```

```
002452  104401                  TYP3:   TYPE
002454  005545                          CARLF
002456  016046                          MOV     R0,-(SP)                ;PUT ADDRESS ON STACK
002460  104402                          TYPOC                           ;TYPE ADDR.
002462  104401                          TYPE
002464  005610                          COLON
002466  012046                  TYP2:   MOV     (R0)+,-(SP)             ;PUT DATA ON STACK
002470  104402                          TYPOC                           ;TYP DATA
002472  104401                          TYPE                            ;TYPE 2 SPACES
002474  005502                          SP2
002476  005301                          DEC     R1                      ;FINISHED?
002500  001361                          BNE     TYP0                    ;IF NOT:  BR
002502  000207                          RTS     PC                      ;RETURN
002504  005000                  ALTACT: CLR     R0
002506  062700  000002          ALT1:   ADD     #2,R0                   ;BUMP TALBE INDEX
002512  026027  005004  177777          CMP     TMSG(R0),#-1            ;CHECKED ALL KNOWN VERSIONS?
002520  001425                          BEQ     AUTERR                  ;IF SO:  BR
002522  026067  004626  176272          CMP     TXCRC(R0),ACTCRC        ;DOES THIS CRC AGREE?
002530  001366                          BNE     AUT1                    ;IF NOT:  KEEP LOOKING
002532  023727  000042  002042          CMP     @#42,#SENDAD            ;UNDER ACT AUTO ACCEPT?
002540  001405                          BEQ     AUT3                    ;IF SO:  BR
002542  016067  005004  000002          MOV     TMSG(R0),AUT2           ;SET UP VERSION MESSAGE
002550  104401                          TYPE
002552  000000                  AUT2:   0
002554  016067  004626  176234  ALT3:   MOV     TXCRC(R0),EXCRC         ;SET EXPECTED CRC
002562  016067  004650  176230          MOV     TXLPC(R0),EXLPC         ;SET EXPECTED LPC
002570  000167  177124                  JMP     CK1                     ;CHECK LPC

002574  104401                  AUTERR: TYPE
002576  005614                          AUTERM
002600  012767  000001  176242          MOV     #1,$FATAL               ;MOVE TO MAILBOX ERROR NO.  ****  :  ****
002606  012767  000001  176232          MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
002614  000000                          HALT                            ;AUTO ACCEPT FAILED

                                .SBTTL  TTY INPUT ROUTINE

                                ;;**********************************************************************
002616  177560                  $TKS:   .WORD   177560          ;;TTY KBD STATUS
002620  177562                  $TKB:   .WORD   177562          ;;TTY KBD BUFFER
                                .ENABL  LSB

                                .DSABL  LSB


                                ;;**********************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
002622  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
002624  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
002632  105777  177760          1$:     TSTB    @$TKS           ;;WAIT FOR
002636  100375                          BPL     1$              ;;A CHARACTER
```

```
                                        TTY INPUT ROUTINE

002640  117766  177754  000004            MOVB    @$TKB,4(SP)     ;;READ THE TTY
002646  042766  177600  000004            BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
002654  026627  000004  000023            CMP     4(SP),#23       ;;IS IT A CONTROL-S?
002662  001013                            BNE     3$              ;;BRANCH IF NO
002664  105777  177726            2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
002670  100375                            BPL     2$              ;;LOOP UNTIL ITS THERE
002672  117746  177722                    MOVB    @$TKB,-(SP)     ;;GET CHARACTER
002676  042716  177600                    BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
002702  022627  000021                    CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
002706  001366                            BNE     2$              ;;IF NOT DISCARD IT
002710  000750                            BR      1$              ;;YES, RESUME
002712  026627  000004  000140  3$:       CMP     4(SP),#140      ;;IS IT UPPER CASE?
002720  002407                            BLT     4$              ;;BRANCH IF YES
002722  026627  000004  000175            CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
002730  003003                            BGT     4$              ;;BRANCH IF YES
002732  042766  000040  000004            BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
002740  000002                    4$:     RTI                     ;;GO BACK TO USER
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                        ;*CALL:
                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

002742  010346                    $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
002744  005046                            CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
002746  012703  003176            1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
002752  022703  003206            2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
002756  101456                            BLOS    4$              ;;BR IF YES
002760  104405                            RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
002762  112613                            MOVB    (SP)+,(R3)      ;;GET CHARACTER
002764  122713  000177            10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
002770  001022                            BNE     5$              ;;BR IF NO
002772  005716                            TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
002774  001007                            BNE     6$              ;;BR IF NO
002776  112767  000134  000170            MOVB    #'\,9$          ;;TYPE A BACK SLASH
003004  104401  003174                    TYPE    ,9$
003010  012716  177777                    MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
003014  005303                    6$:     DEC     R3              ;;BACKUP BY ONE
003016  020327  003176                    CMP     R3,#$TTYIN      ;;STACK EMPTY?
003022  103434                            BLO     4$              ;;BR IF YES
003024  111367  000144                    MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
003030  104401  003174                    TYPE    ,9$             ;;GO TYPE
003034  000746                            BR      2$              ;;GO READ ANOTHER CHAR.
003036  005716                    5$:     TST     (SP)            ;;RUBOUT KEY SET?
003040  001406                            BEQ     7$              ;;BR IF NO
003042  112767  000134  000124            MOVB    #'\,9$          ;;TYPE A BACK SLASH
003050  104401  003174                    TYPE    ,9$
003054  005016                            CLR     (SP)            ;;CLEAR THE RUBOUT KEY
003056  122713  000025            7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
003062  001003                            BNE     8$              ;;BR IF NO
003064  104401  003212                    TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
003070  000726                            BR      1$              ;;GO START OVER
003072  122713  000022            8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "^R"?
003076  001011                            BNE     3$              ;;BRANCH IF NO
003100  105013                            CLRB    (R3)            ;;CLEAR THE CHARACTER
```

```
                                        TTY INPUT ROUTINE

   898  003102  104401  003207                TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
   899  003106  104401  003176                TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
   900  003112  000717                        BR      2$              ;;GO PICKUP ANOTHER CHACTER
   901  003114  104401  003206          4$:   TYPE    ,$QUES          ;;TYPE A '?'
   902  003120  000712                        BR      1$              ;;CLEAR THE BUFFER AND LOOP
   903  003122  111367  000046          3$:   MOVB    (R3),9$         ;;ECHO THE CHARACTER
   904  003126  104401  003174                TYPE    ,9$
   905  003132  122723  000015                CMPB    #15,(R3)+       ;;CHECK FOR RETURN
   906  003136  001305                        BNE     2$              ;;LOOP IF NOT RETURN
   907  003140  105063  177777                CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
   908  003144  104401  003210                TYPE    ,$LF            ;;TYPE A LINE FEED
   909  003150  005726                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
   910  003152  012603                        MOV     (SP)+,R3        ;;RESTORE R3
   911  003154  011646                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
   912  003156  016666  000004  000002        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
   913  003164  012766  003176  000004        MOV     #$TTYIN,4(SP)
   914  003172  000002                        RTI                     ;;RETURN
   915  003174     000                 9$:    .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
   916  003175     000                        .BYTE   0               ;;TERMINATOR
   917  003176  000010                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
   918  003206     077                $QUES:  .ASCII  "?"     ;;QUESTION MARK
   919  003207     015                $CRLF:  .ASCII  <15>    ;CARRIAGE RETURN
   920  003210  000012                $LF:    .ASCIZ  <12>    ;;LINE FEED
   921  003212  052536  005015  000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
   922  003217     136  006507  000012 $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
   923  003224  005015  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
   924  003232  020075     000
   925  003235     040  047040  053505 $MNEW:  .ASCIZ  /  NEW = /
   926  003242  036440  000040
   927                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
   928
   929                                  ;;**************************************************************
   930                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
   931                                  ;*CHANGE IT TO BINARY.
   932                                  ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
   933                                  ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
   934                                  ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
   935                                  ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
   936                                  ;*CALL:
   937                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
   938                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
   939                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
   940
   941  003246  011646                $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
   942  003250  016666  000004  000002        MOV     4(SP),2(SP)     ;;INPUT NUMBER
   943  003256  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
   944  003260  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
   945  003262  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
   946  003264  104406                1$:     RDLIN                   ;;READ AN ASCIZ LINE
   947  003266  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
   948  003270  010067  000100                MOV     R0,5$           ;;AND SAVE IT
   949  003274  005001                        CLR     R1              ;;CLEAR DATA WORD
   950  003276  005002                        CLR     R2
   951  003300  112046                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
   952  003302  001420                        BEQ     3$              ;;IF ZERO GET OUT
   953  003304  122716  000060                CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
```

READ AN OCTAL NUMBER FROM THE TTY

```
003310  003026                  BGT     4$              ;;IS AN OCTAL DIGIT
003312  122716  000067          CMPB    #'7,(SP)
003316  002423                  BLT     4$
003320  006301                  ASL     R1              ;;*2
003322  006102                  ROL     R2
003324  006301                  ASL     R1              ;;*4
003326  006102                  ROL     R2
003330  006301                  ASL     R1              ;;*8
003332  006102                  ROL     R2
003334  042716  177770          BIC     #177770,(SP     ;;STRIP THE ASCII JUNK
003340  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
003342  000756                  BR      2$              ;;LOOP
003344  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
003346  010166  000012          MOV     R1,12(SP)       ;;SAVE THE RESULT
003352  010267  000026          MOV     R2,$HIOCT
003356  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
003360  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
003362  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
003364  000002                  RT*     
003366  005726          4$:             (SP)+           ;;CLEAN PARTIAL FROM STACK
003370  105010                  CLRB    (R0)            ;;SET A TERMINATOR
003372  104401                  TYPE                    ;;TYPE UP THRU THE BAD CHAR.
003374  000000          5$:     .WORD   0
003376  104401  003206          TYPE    ,$QUES          ;;"?" "CR" & "LF"
003402  000730                  BR      1$              ;;TRY AGAIN
003404  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                        .SBTTL  TYPE ROUTINE

                        ;;**************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*

003406  105767  000265  $TYPE:   TSTB    $TPFLG          ;;IS THERE A TERMINAL?
003412  100002                  BPL     1$              ;;BR IF YES
003414  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
003416  000430                  BR      3$              ;;LEAVE
003420  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
003422  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
003426  122767  000001  175432  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
003434  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
003436  132767  000100  175423  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
003444  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
003446  010067  000004          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
003452  004767  000230          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
003456  000000          61$:    .WORD   0               ;;MESSAGE ADDRESS
```

```
003460  132767  000040  175401  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
003466  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
003470  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
003472  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
003474  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
003476  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
003500  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
003504  000002                          RTI                     ;;RETURN
003506  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
003512  001430                          BEQ     8$
003514  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
003520  001006                          BNE     5$
003522  005726                          TST     (SP)+           ;;POP <CR><LF> EQUIV
003524  104401                          TYPE                    ;;TYPE A CR AND LF
003526  003207                          $CRLF
003530  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
003534  000755                          BR      2$              ;;GET NEXT CHARACTER
003536  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
003542  126726  000130          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
003546  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
003550  016746  000120                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
003554  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
003560  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
003562  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
003566  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
003572  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

003574  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
003600  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
003604  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
003612  001372                          BNE     9$              ;;TAB STOP
003614  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
003616  000724                          BR      2$              ;;GET NEXT CHARACTER
003620  105777  000044          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
003624  100375                          BPL     $TYPEC
003626  116677  000002  000036          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
003634  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
003642  001003                          BNE     1$              ;;BRANCH IF NO
003644  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
003650  000406                          BR      $TYPEX          ;;EXIT
003652  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
003660  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
003662  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
003664  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
003666  000207                  $TYPEX: RTS     PC

003670  177564                  $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
003672  177566                  $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
003674     000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
003675     002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
003676     012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
003677     000                  $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
                                .SBTTL  APT COMMUNICATIONS ROUTINE
```

L02


```
1066
1067                                    ;;*****************************************************************
1068  003700  112767  000001  000236  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
1069  003706  112767  000001  000226  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
1070  003714  000403                  BR      $ATYC
1071  003716  112767  000001  000220  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
1072  003724                          $ATYC:
1073  003724  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1074  003726  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1075  003730  105767  000206          TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
1076  003734  001450                  BEQ     5$              ;;IF NOT:  BR
1077  003736  122767  000001  175122  CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
1078  003744  001031                  BNE     3$              ;;IF NOT:  BR
1079  003746  132767  000100  175113  BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
1080  003754  001425                  BEQ     3$              ;;IF NOT:  BR
1081  003756  017600  000004          MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
1082  003762  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
1083  003770  005767  175052  1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
1084  003774  001375                  BNE     1$              ;;IF NOT:  WAIT
1085  003776  010067  175060          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
1086  004002  105720          2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
1087  004004  001376                  BNE     2$
1088  004006  166700  175050          SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
1089  004012  006200                  ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
1090  004014  010067  175044          MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
1091  004020  012767  000004  175020  MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
1092  004026  000413                  BR      5$
1093  004030  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
1094  004036  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
1095  004044  016746  173726          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
1096  004050  004767  177332          JSR     PC,$TYPE        ;;CALL TYPE MACRO
1097  004054  000000                  4$:     .WORD   0
1098  004056                          5$:
1099  004056  105767  000062          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
1100  004062  001416                  BEQ     12$             ;;IF NOT:  BR
1101  004064  005767  174776          TST     $ENV            ;;RUNNING UNDER APT?
1102  004070  001413                  BEQ     12$             ;;IF NOT:  BR
1103  004072  005767  174750          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
1104  004076  001375                  BNE     11$             ;;IF NOT:  WAIT
1105  004100  017667  000004  174742  MOV     @4(SP),$FATAL   ;;GET ERROR #
1106  004106  062766  000002  000004  ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
1107  004114  005267  174726          INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
1108  004120  105067  000020          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
1109  004124  105067  000013          CLRB    $LFLG           ;;CLEAR LOG FLAG
1110  004130  105067  000006          CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
1111  004134  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
1112  004136  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
1113  004140  000207                  RTS     PC              ;;RETURN
1114  004142  000             $MFLG:  .BYTE   0               ;;MESSG. FLAG
1115  004143  000             $LFLG:  .BYTE   0               ;;LOG FLAG
1116  004144  000             $FFLG:  .BYTE   0               ;;FATAL FLAG
1117  004146                          .EVEN
1118  000200                  APTSIZE=200
1119  000001                  APTENV=001
1120  000100                  APTSPOOL=100
1121  000040                  APTCSLP=040
```

```
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

004146  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
004152  116667  000001  000211          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
004160  112667  000207                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
004164  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
004170  000406                          BR      $TYPON
004172  112767  000001  000171  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
004200  112767  000006  000165          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
004206  112767  000005  000154  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
004214  010346                          MOV     R3,-(SP)        ;;SAVE R3
004216  010446                          MOV     R4,-(SP)        ;;SAVE R4
004220  010546                          MOV     R5,-(SP)        ;;SAVE R5
004222  116704  000145                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
004226  005404                          NEG     R4
004230  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
004234  110467  000132                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
004240  116704  000125                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
004244  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
004250  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
004252  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
004254  000404                          BR      3$              ;;GO DO MSB
004256  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
004260  006105                          ROL     R5
004262  006105                          ROL     R5
004264  010503                          MOV     R5,R3
004266  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
004270  105367  000076                  DECB    $OMODE          ;;TYPE THIS DIGIT?
004274  100016                          BPL     7$              ;;BR IF NO
004276  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
004302  001002                          BNE     4$              ;;TEST FOR 0
004304  005704                          TST     R4              ;;SUPPRESS THIS 0?
004306  001403                          BEQ     5$              ;;BR IF YES
```

BINARY TO OCTAL (ASCII) AND TYPE

```
1178  004310  005204                  4$:     INC     R4                      ;;DON'T SUPPRESS ANYMORE 0'S
1179  004312  052703  000060                  BIS     #'0,R3                  ;;MAKE THIS DIGIT ASCII
1180  004316  052703  000040            5$:    BIS     #' ,R3                  ;;MAKE ASCII IF NOT ALREADY
1181  004322  110367  000040                  MOVB    R3,8$                   ;;SAVE FOR TYPING
1182  004326  104401  004366                  TYPE    ,8$                     ;;GO TYPE THIS DIGIT
1183  004332  105367  000032            7$:    DECB    $OCNT                   ;;COUNT BY 1
1184  004336  003347                          BGT     2$                      ;;BR IF MORE TO DO
1185  004340  002402                          BLT     6$                      ;;BR IF DONE
1186  004342  005204                          INC     R4                      ;;INSURE LAST DIGIT ISN'T A BLANK
1187  004344  000744                          BR      2$                      ;;GO DO THE LAST DIGIT
1188  004346  012605                  6$:     MOV     (SP)+,R5                ;;RESTORE R5
1189  004350  012604                          MOV     (SP)+,R4                ;;RESTORE R4
1190  004352  012603                          MOV     (SP)+,R3                ;;RESTORE R3
1191  004354  016666  000002  000004         MOV     2(SP),4(SP)             ;;SET THE STACK FOR RETURNING
1192  004362  012616                          MOV     (SP)+,(SP)
1193  004364  000002                          RTI                             ;;RETURN
1194  004366     000                  8$:     .BYTE   0                       ;;STORAGE FOR ASCII DIGIT
1195  004367     000                          .BYTE   0                       ;;TERMINATOR FOR TYPE ROUTINE
1196  004370     000                  $OCNT:  .BYTE   0                       ;;OCTAL DIGIT COUNTER
1197  004371     000                  $OFILL: .BYTE   0                       ;;ZERO FILL SWITCH
1198  004372  000000                  $OMODE: .WORD   0                       ;;NUMBER OF DIGITS TO TYPE
1199                                  .SBTTL  POWER DOWN AND UP ROUTINES
1200
1201                                  ;;**************************************************************
1202                                  ;POWER DOWN ROUTINE
1203  004374  012737  004534  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC        ;;SET FOR FAST UP
1204  004402  012737  000340  000026          MOV     #340,@#PWRVEC+2         ;;PRIO:7
1205  004410  010046                          MOV     R0,-(SP)                ;;PUSH R0 ON STACK
1206  004412  010146                          MOV     R1,-(SP)                ;;PUSH R1 ON STACK
1207  004414  010246                          MOV     R2,-(SP)                ;;PUSH R2 ON STACK
1208  004416  010346                          MOV     R3,-(SP)                ;;PUSH R3 ON STACK
1209  004420  010446                          MOV     R4,-(SP)                ;;PUSH R4 ON STACK
1210  004422  010546                          MOV     R5,-(SP)                ;;PUSH R5 ON STACK
1211  004424  017746  174350                  MOV     @SWR,-(SP)              ;;PUSH @SWR ON STACK
1212  004430  010667  000104                  MOV     SP,$SAVR6               ;;SAVE SP
1213  004434  012737  004446  000024          MOV     #$PWRUP,@#PWRVEC        ;;SET UP VECTOR
1214  004442  000000                          HALT
1215  004444  000776                          BR      .-2                     ;;HANG UP
1216
1217                                  ;;**************************************************************
1218                                  ;POWER UP ROUTINE
1219  004446  012737  004534  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC        ;;SET FOR FAST DOWN
1220  004454  016706  000060                  MOV     $SAVR6,SP               ;;GET SP
1221  004460  005067  000054                  CLR     $SAVR6                  ;;WAIT LOOP FOR THE TTY
1222  004464  005267  000050            1$:   INC     $SAVR6                  ;;WAIT FOR THE INC
1223  004470  001375                          BNE     1$                      ;;OF  WORD
1224  004472  012677  174302                  MOV     (SP)+,@SWR              ;;POP STACK INTO @SWR
1225  004476  012605                          MOV     (SP)+,R5                ;;POP STACK INTO R5
1226  004500  012604                          MOV     (SP)+,R4                ;;POP STACK INTO R4
1227  004502  012603                          MOV     (SP)+,R3                ;;POP STACK INTO R3
1228  004504  012602                          MOV     (SP)+,R2                ;;POP STACK INTO R2
1229  004506  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
1230  004510  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
1231  004512  012737  004374  000024          MOV     #$PWRDN,@#PWRVEC        ;;SET UP THE POWER DOWN VECTOR
1232  004520  012737  000340  000026          MOV     #340,@#PWRVEC+2         ;;PRIO:7
1233  004526  10  01                         TYPE                            ;REPORT THE POWER FAILURE
```

POWER DOWN AND UP ROUTINES

```
                                        $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
                                                RTI
                                        $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
                                                BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
                                        $SAVR6: 0                       ;;PUT THE SP HERE
004542  035015  047520  042523          $POWER: .ASCIZ  <15><12>"POWER"
004550  000122
                                                .EVEN
                                        .SBTTL  TRAP DECODER

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

004552  010046                          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
004554  016600  000002                          MOV     2(SP),R0        ;;GET TRAP ADDRESS
004560  005740                                  TST     -(R0)           ;;BACKUP BY 2
004562  111000                                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
004564  006300                                  ASL     R0              ;;POSITION FOR INDEXING
004566  016000  004636                          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
004572  000200                                  RTS     R0              ;;GO TO ROUTINE


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

004574  011646                          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
004576  016666  000004  000002                  MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
004604  000002                                  RTI                     ;;RESTORE THE PSW


                                        .SBTTL  TRAP TABLE

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

                                        ;       ROUTINE
                                        ;       -------
004606  004574                          $TRPAD: .WORD   $TRAP2
004610  003406                                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
004612  004172                                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
004614  004146                                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
004616  004206                                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)


004620  002622                                  $RDCHR  ;;CALL=RDCHR    TRAP+5(104405)  TTY TYPEIN CHARACTER ROUTINE
004622  002742                                  $RDLIN  ;;CALL=RDLIN    TRAP+6(104406)  TTY TYPEIN STRING ROUTINE
004624  003246                                  $RDOCT  ;;CALL=RDOCT    TRAP+7(104407)  READ AN OCTAL NUMBER FROM TTY


004626  177777                          TYCRC:  -1
004630  000571                                  571             ;M9301 - YA VERSION
004632  000457                                  457             ;M9301 - YB VERSION
004634  000243                                  243             ;M9301 - YC VERSION
004636  000635                                  635             ;M9400 - YA(OR YC) VERSION
004640  000420                                  420             ;M9301 - YF VERSION
004642  177777                                  -1
```

```
:290  004644  177777                    -1
1291  004646  177777                    -1
1292
1293  004650  177777          TXLPO:    -1
1294  004652  133725                    133725          ;M9301 - YA VERSION
1295  004654  017563                    17563           ;M9301 - YB VERSION
1296  004656  141744                    141744          ;M9301 - YC VERSION
1297  004660  047613                    47613           ;M9400 - YA(ORYC) VERSION
1298  004662  075747                    75747           ;M9301 - YF VERSION
1299  004664  177777                    -1
1300  004666  177777                    -1
1301  004670  177777                    -1
1302  004672  177777                    -1
1303
1304  004674  177777          TDLN1:    -1
1305  004676  001000                    1000            ;M9301 - YA VERSION
1306  004700  001000                    1000            ;M9301 - YB VERISION
1307  004702  001000                    1000            ;M9301 - YC VERSION
1308  004704  001000                    1000            ;M9400 - YA(OR YC) VERSION
1309  004706  001000                    1000            ;M9301 - YF VERSION
1310  004710  177777                    -1
1311  004712  177777                    -1
1312  004714  177777                    -1
1313
1314  004716  177777          TRMSA1:   -1
1315  004720  173000                    173000          ;M9301 - YA VERSION
1316  004722  173000                    173000          ;M9301 - YB VERSION
1317  004724  173000                    173000          ;M9301 - YC VERSION
1318  004726  173000                    173000          ;M9400 - YA(OR YC) VERSION
1319  004730  173000                    173000          ;M9301 - YF VERSION
1320  004732  177777                    -1
1321  004734  177777                    -1
1322  004736  177777                    -1
1323
1324  004740  177777          TDLN2:    -1
1325  004742  001000                    1000            ;M9301 - YA VERSION
1326  004744  001000                    1000            ;M9301 - YB VERSION
1327  004746  001000                    1000            ;M9301 - YC VERSION
1328  004750  001000                    1000            ;M9400 -YA(OR YC) VERSION
1329  004752  001000                    1000            ;M9301 -YF VERSION
1330  004754  177777                    -1
1331  004756  177777                    -1
1332  004760  177777                    -1
1333
1334  004762  177777          TRMSA2:   -1
1335  004764  165000                    165000          ;M9301 - YA VERSION
1336  004766  165000                    165000          ;M9301 - YB VERSION
1337  004770  165000                    165000          ;M9301 - YC VERSION
1338  004772  165000                    165000          ;M9400 - YA(OR YC) VERSION
1339  004774  165000                    165000          ;M9301 - YF VERSION
1340  004776  177777                    -1
1341  005000  177777                    -1
1342  005002  177777                    -1
1343
1344  005004  177777          TMSG:     -1
1345  005006  005636                    MSG1
```

```
                                                        MSG2
                                                        MSG3
                                                        MSG4
                                                        MSG5
                                                        -1
                                                        -1
                                                        -1

 005026  005015  030122  020115  TITL:   .ASCIZ  <15><12>/ROM TEST/
 042524  052123      000
     015  052012  050131  GETCRC: .ASCIZ  <15><12>/TYPE CRC VALUE:  /
 020105  051103  020103
 040526  052514  035105
     040  052012  050131  GETLPC: .ASCIZ  <15><12>/TYPE LPC VALUE:  /
 020105  050114  020103
 040526  052514  035105
     040  042412  050130  EXCRMG: .ASCIZ  <15><12>/EXPECTED CRC =  /
 041505  042524  020104
 051103  020103  020075
 000040
 005015  042412  050130  EXLPMG: .ASCIZ  <15><12><12>/EXPECTED LPC =  /
 041505  042524  020104
 050114  020103  020075
 000040
 005015  047503  050115  ACCRMG: .ASCIZ  <15><12>/COMPUTED CRC =  /
 052125  042105  041440
 041522  036440  020040
     000
     015  041412  046517  ACLPMG: .ASCIZ  <15><12>/COMPUTED LPC =  /
 052520  042524  020104
 050114  020103  020075
 000040
 005015  042412  042116  EOTST:  .ASCIZ  <15><12><12>/END OF TEST/
 047440  020106  042524
 052123      000
     015  052012  050131  SA1:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 1ST ROM ADDR. SPACE:  /
 020105  052123  051101
 040524  043516  040440
 042104  027122  047440
 020106  051461  020124
 047522  020115  042101
 051104  020056  050123
 041501  035105  020040
     000
     015  052012  050131  SA2:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 2ND ROM ADDR. SPACE:  /
 020105  052123  051101
 040524  043516  040440
 042104  027122  047440
 020106  047062  020104
 047522  020115  042101
 051104  020056  050123
 041501  035105  020040
     000
```

E03


```
1402  005405  000015  052012  050131  SIZE1:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 1ST ROM ADDR. SPACE:  /
1403  005412  020105  042514  043516
1404  005420  044124  024040  054502
1405  005426  042524  024523  047440
1406  005434  020106  051461  020124
1407  005442  047522  020115  042101
1410  005450  051104  020056  050123
1411  005456  041501  035105  020040
1412  005464  000000
1413  005465  000015  052012  050131  SIZE2:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 2ND ROM ADDR. SPACE:  /
1414  005472  020105  042514  043516
1415  005500  044124  024040  054502
1416  005506  042524  024523  047440
1417  005514  020106  047062  020104
1420  005522  047522  020115  042101
1421  005530  051104  020056  050123
1422  005536  041501  035105  020040
1423  005544  000000
1424  005545  000015  000012          CARLF:  .ASCIZ  <15><12>
1425  005550  020040  000000          SP2:    .ASCIZ  /  /
1426  005553  000015  040412  042104  TYPHDR: .ASCIZ  <15><12>/ADDRESS              DATA/
1427  005560  042522  051523  020040
1430  005566  020040  020040  020040
1431  005574  020040  020040  020040
1432  005602  042040  052101  000101
1433  005610  020072  000040          COLON:  .ASCIZ  /: /
1434  005614  005015  047125  047113  AUTERM: .ASCIZ  <15><12>/UNKNOWN MODULE /
1435  005622  053517  020116  047515
1436  005630  052504  042514  000040
1437
1440  005636  005015  034515  030063  MSG1:   .ASCIZ  <15><12>/M9301 - YA/
1441  005644  020061  020055  040531
1442  005652  000000
1443  005653  000015  046412  031471  MSG2:   .ASCIZ  <15><12>/M9301 - YB/
1444  005660  030460  026440  054440
1445  005666  000102
1446  005670  005015  034515  030063  MSG3:   .ASCIZ  <15><12>/M9301 - YC/
1447  005676  020061  020055  041531
1450  005704  000000
1451  005705  000015  046412  032071  MSG4:   .ASCIZ  <15><12>/M9400 - YA,YC/
1452  005712  030060  026440  054440
1453  005720  026101  041531  000000
1454  005726  000015  046412  031471  MSG5:   .ASCIZ  <15><12>/M9301 - YF/
1455  005732  030460  026440  054440
1456  005740  000106
1457          000001          .END
```

F03


```
ABASE = 000000    ALSWR = 000000    CRLF  = 000200    R6    =%000006    TYPE  = 104401
ACCRMG  005160    ARGACT  002504    DATLN1  001006    R7    =%000007    TYPHDR  005553
ACDW1 = 000000    ARGTRM  005614    DATLN2  001012    SA1     005246    TYPOC = 104402
ACDW2 = 000000    ARGERR  002574    DDISP = 177570    SA2     005325    TYPON = 104404
ACLPMG  005203    ARG1    002506    DISPLA  001002    SIZE1   005405    TYPOS = 104403
ACPUOP= 000000    ARG2    002552    DISPRE  000174    SIZE2   005465    TYPOUT  001030
ACTCRC  001022    ARG3    002554    DSWR  = 177570    SP2     005550    TYPROM  002364
ACTLPC  001024    ARGCT1= 000000    EMTVEC= 000030    STACK = 001100    TYPR1   002410
ADDW0 = 000000    ARGCT2= 000000    ENDOT   002430    START   001076    TYP0    002444
ADDW1 = 000000    BIT0    = 000001    EOTST   005226    STKLMT= 177774    TYP2    002466
ADDW10= 000000    BIT00   = 000001    ERRVEC= 000004    ST2     001302    TYP3    002452
ADDW11= 000000    BIT01   = 000002    EXCRC   001016    SWR     001000    XCR     002302
ADDW12= 000000    BIT02   = 000004    EXCRMG  005111    SWREG   000176    XCRS    001014
ADDW13= 000000    BIT03   = 000010    EXLPC   001020    SW0   = 000001    $APTHD  001032
ADDW14= 000000    BIT04   = 000020    EXLPMG  005134    SW00  = 000001    $ATYC   003724
ADDW15= 000000    BIT05   = 000040    GETCRC  005041    SW01  = 000002    $ATY1   003700
ADDW2 = 000000    BIT06   = 000100    GETIN   001352    SW02  = 000004    $ATY3   003736
ADDW3 = 000000    BIT07   = 000200    GETLPC  005065    SW03  = 000010    $ATY4   003716
ADDW4 = 000000    BIT08   = 000400    GET2    001422    SW04  = 000020    $CHPRC  003664
ADDW5 = 000000    BIT09   = 001000    GOAGIN  002052    SW05  = 000040    $CNT_G  003217
ADDW6 = 000000    BIT1    = 000002    HT    = 000011    SW06  = 000100    $CNTLJ  003212
ADDW7 = 000000    BIT10   = 002000    IOTVEC= 000020    SW07  = 000200    $CPUOP  001074
ADDW8 = 000000    BIT11   = 004000    LF    = 000012    SW08  = 000400    $CRLF   003207
ADDW9 = 000000    BIT12   = 010000    LPC     002326    SW09  = 001000    $DEVCT  001056
ADEVCT= 000000    BIT13   = 020000    LPC1    002334    SW1   = 000002    $ENDAD  002042
ADEVM = 000000    BIT14   = 040000    LPC2    002350    SW10  = 002000    $ENV    001066
AENV  = 000000    BIT15   = 100000    MSG1    005636    SW11  = 004000    $ENVM   001067
AENVM = 000000    BIT2    = 000004    MSG2    005653    SW12  = 010000    $ETABL  001066
AFATAL= 000000    BIT3    = 000010    MSG3    005670    SW13  = 020000    $ETEND  001076
AMAOR1= 000000    BIT4    = 000020    MSG4    005705    SW14  = 040000    $FATAL  001050
AMAOR2= 000000    BIT5    = 000040    MSG5    005725    SW15  = 100000    $FFLG   004144
AMAOR3= 000000    BIT6    = 000100    PARCNT  001026    SW2   = 000004    $FILLC  003676
AMAOR4= 000000    BIT7    = 000200    PARITY  002234    SW3   = 000010    $FILLS  003675
AMAMS1= 000000    BIT8    = 000400    PIRQ  = 177772    SW4   = 000020    $HISTS  001032
AMAMS2= 000000    BIT9    = 001000    PIRQVE= 000240    SW5   = 000040    $HIOCT  003404
AMAMS3= 000000    BPTVEC= 000014    PR0   = 000000    SW6   = 000100    $ILLJP  004534
AMAMS4= 000000    CARLF   005545    PR1   = 000040    SW7   = 000200    $LF     003210
AMSG20= 000000    CHECK   001472    PR2   = 000100    SW8   = 000400    $LFLG   004143
AMSGLG= 000000    CHO     001550    PR3   = 000140    SW9   = 001000    $MAIL   001046
AMSGTY= 000000    CH1     001626    PR4   = 000200    TBITVE= 000014    $MBADR  001034
AMTYP1= 000000    CH1     001720    PR5   = 000240    TDLN1   004674    $MFLG   004142
AMTYP2= 000000    CH2     002010    PR6   = 000300    TDLN2   004740    $MNEW   003235
AMTYP3= 000000    CL_AST  002166    PR7   = 000340    TITL    005026    $MSGAD  001062
AMTYP4= 000000    CLP0    002234    PS    = 177776    TKVEC = 000060    $MSGLG  001064
APASS = 000000    CLP1    002246    PSW   = 177776    TMSG    005004    $MSGTY  001046
APRIOR= 000000    CLP2    002300    PWRVEC= 000024    TPVEC = 000064    $MSWR   003234
APTCSU= 000040    CLO     002064    RDCHR = 104405    TRAPVE= 000034    $NULL   003674
APTENV= 000001    CL1     002130    RDLIN = 104406    TRMSA1  004716    $OCNT   004370
APTSIZ= 000200    CL2     002150    RDOCT = 104407    TRMSA2  004762    $OMODE  004372
APTSPO= 000100    CL3     002104    RESTRT  001274    TRTVEC= 000014    $PASS   001054
ASWREG= 000000    COLON   005610    RESVEC= 000010    TXCRC   004626    $PAST_M 001040
ATESTN= 000000    CR    = 000015    ROMSA1  001004    TXLPC   004650    $POWER  004542
AUTO  = 000000    CRC     002058    ROMSA2  001010    TYP     002436    $PWRDN  004374
```

```
.MAIN.  MACY11 27.1006  27-SEP-76  14:46  PAGE 30
DZM9A8.P11    27-SEP-76 14:43          SYMBOL TABLE

$PWRMG  004530          $SETUP= 000014          $TPB    003672          $TTYIN  003176          $UNITM  001042
$PWRUP  004446          $STUP = 177777          $TPFLG  003677          $TYPE   003406          $USWR   001072
$QUES   003206          $SVPC = 001032          $TPS    003670          $TYPEC  003620          $CFILL  004371
$RDCHR  002622          $SWR  = 000000          $TRAP   004552          $TYPEX  003666          .       = 005742
$RDLIN  002742          $SWREG  001070          $TRAP2  004574          $TYPOC  004172          .$Y     = 001032
$RDOCT  003246          $TESTN  001052          $TRP    = 000010        $TYPON  004206
$RDSZ = 000010          $TKB    002620          $TRPAD  004606          $TYPOS  004146
$SAVR6  004540          $TKS    002616          $TSTM   001036          $UNIT   001060

. ABS.  005742      000

  ERRORS DETECTED:  0
  DEFAULT GLOBALS GENERATED:  0

  DZM9A8,DZM9A8/SOL=DZM9A8
  RUN-TIME:  37.13 1 SECONDS
  RUN-TIME RATIO: 117/52=2.2
  CORE USED:  20K  (39 PAGES)
```

Spooler runtime 5 Seconds, 27 KCS, 128 disk reads, 4 disk writes, 29 pages