M9301-YH

BOOTSTRAP TERMINATOR

MD-11-DZM9A-C

EP-DZM9A-C-DL-C

APR 1977

FICHE 1 OF 1

digital

MADE IN USA

```
000000                          .REPT 0
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZM9A-C-D
PRODUCT NAME:	BOOTSTRAP/TERMINATOR (M9301, M9400)
PROGRAM DATE:	FEBRUARY 15,1977
MAINTAINER:	DIAGNOSTIC ENGINEERING

```
                        ****************************************
                        *                                      *
                        * 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.


                * *  CAUTION  * *

BECAUSE THE CONTENTS READ FROM LOCATION 773024
OF THE M9301 OPTION IS CONFIGURATION DEPENDANT(SWITCH
REGISTER DEPENDANT), THIS LOCATION IS NOT INCLUDED IN
THE DATA CHECK.
THIS LOCATION CAN BE VERIFIED BY EXAMINING IT OR BY USING
THE ALTERNATE STARTING ADDRESS(SEE SECTION 2.1.3) TO
PRINT OUT THIS LOCATION.
```

```
                                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

WARNING... IN ORDER TO ALLOW TESTING OF M7942-YB BOARDS ON THE VT71, IF LOACTION 176 IS SET TO 6, THE SOFTWARE SWITCH REGISTER WILL BE USED REGARDLESS OF HARDWARE SWITCH REGISTER AVAILABILITY.

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
6	M7942-YB
7	M9301-YD
10	M9400-YH (OR YK)
11	M9311
12	M9301-YH

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 VAULE 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
AROM 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 APT 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.

*** * CAUTION * ***

BECAUSE THE CONTENTS READ FROM LOCATION 773024 OF THE M9301 OPTION IS CONFIGURATION DEPENDANT(SWITCH REGISTER DEPENDANT), THIS LOCATION IS NOT INCLUDED IN THE DATA CHECK.
THIS LOCATION CAN BE VERIFIED BY EXAMINING IT OR BY USING THE ALTERNATE STARTING ADDRESS(SEE SECTION 2.1.3) TO PRINT OUT THIS LOCATION.

```
                                6.0     LISTING


                                        .ENDR
```

```
                                .ENABLE ABS
                                .LIST   ME
                                .NLIST  MC,MD,CND
000000                          $SWR=0
                                .SBTTL  BASIC DEFINITIONS
(1)
(1)                             ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                          STACK=  1100
(1)                             .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
(1)                             .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
(1)
(1)                             ;*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
(1)                             .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
(1)
(1)                             ;*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
(1)
(1)                             ;*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
(1)
(1)                             ;*"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
```

```
(1)             000100                  SW06=   100
(1)             000040                  SW05=   40
(1)             000020                  SW04=   20
(1)             000010                  SW03=   10
(1)             000004                  SW02=   4
(1)             000002                  SW01=   2
(1)             000001                  SW00=   1
(1)                                     .EQUIV  SW09,SW9
(1)                                     .EQUIV  SW08,SW8
(1)                                     .EQUIV  SW07,SW7
(1)                                     .EQUIV  SW06,SW6
(1)                                     .EQUIV  SW05,SW5
(1)                                     .EQUIV  SW04,SW4
(1)                                     .EQUIV  SW03,SW3
(1)                                     .EQUIV  SW02,SW2
(1)                                     .EQUIV  SW01,SW1
(1)                                     .EQUIV  SW00,SW0
(1)
(1)                                     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)             100000                  BIT15=  100000
(1)             040000                  BIT14=  40000
(1)             020000                  BIT13=  20000
(1)             010000                  BIT12=  10000
(1)             004000                  BIT11=  4000
(1)             002000                  BIT10=  2000
(1)             001000                  BIT09=  1000
(1)             000400                  BIT08=  400
(1)             000200                  BIT07=  200
(1)             000100                  BIT06=  100
(1)             000040                  BIT05=  40
(1)             000020                  BIT04=  20
(1)             000010                  BIT03=  10
(1)             000004                  BIT02=  4
(1)             000002                  BIT01=  2
(1)             000001                  BIT00=  1
(1)                                     .EQUIV  BIT09,BIT9
(1)                                     .EQUIV  BIT08,BIT8
(1)                                     .EQUIV  BIT07,BIT7
(1)                                     .EQUIV  BIT06,BIT6
(1)                                     .EQUIV  BIT05,BIT5
(1)                                     .EQUIV  BIT04,BIT4
(1)                                     .EQUIV  BIT03,BIT3
(1)                                     .EQUIV  BIT02,BIT2
(1)                                     .EQUIV  BIT01,BIT1
(1)                                     .EQUIV  BIT00,BIT0
(1)
(1)                                     ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1)             000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)             000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)             000014                  TBITVEC=14              ;;"T" BIT
(1)             000014                  TRTVEC= 14              ;;TRACE TRAP
(1)             000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)             000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)             000024                  PWRVEC= 24              ;;POWER FAIL
(1)             000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)             000034                  TRAPVEC=34              ;;"TRAP" TRAP
```

```
(1)             000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)             000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
(1)             000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
384             000000                          .=0
385                                     .SBTTL  TRAP CATCHER
(1)
(1)             000000                          .=0
(1)                                     ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)                                     ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)                                     ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)             000174                          .=174
(1)  000174     000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
(1)  000176     000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
387             000200                  .=200
388  000200     005067  000624                  CLR     TYPOUT
389  000204     000167  000666                  JMP     START
390  000210     012767  000001  000612          MOV     #1,TYPOUT
391  000216     000167  000654                  JMP     START
392
393             177776                  PS=177776
394             000034                  TRAPVEC=34
395
396             001000                  .=1000
397  001000     177570                  SWR:    177570
398  001002     177570                  DISPLAY:        177570
399  001004     173000                  ROMSA1: 173000
400  001006     001000                  DATLN1: 512.
401  001010     165000                  ROMSA2: 165000
402  001012     001000                  DATLN2: 512.
403  001014     000000                  XORS:   0
404  001016     000000                  EXCRC:  0
405  001020     000000                  EXLPC:  0
406  001022     000000                  ACTCRC: 0
407  001024     000000                  ACTLPC: 0
408  001026     000000                  PARCNT: 0
409  001030     000000                  TYPOUT: 0
410
411                                     .SBTTL  ACT11 HOOKS
(1)
(2)                                     ;;*****************************************************************
(1)                                     ;HOOKS REQUIRED BY ACT11
(1)             001032                          $SVPC=.                 ;SAVE PC
(1)             000046                          .=46
(1)  000046     002060                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)             000052                          .=52
(1)  000052     000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
(1)             001032                          .=$SVPC                 ;; RESTORE PC
412                                     .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                     ;;*****************************************************************
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;;*****************************************************************
(1)             001032                          .$X=.   ;;SAVE CURRENT LOCATION
(1)             000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1)  000024     000200                          200     ;;FOR APT START UP
(1)             000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
```

```
    (1)  000044  001032                        $APTHDR ;;POINT TO APT HEADER BLOCK
    (1)          001032                        .=.$X   ;;RESET LOCATION COUNTER
    (2)                                ;;*****************************************************************
    (1)                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
    (1)                                ;INTERFACE SPEC.
    (1)
    (1)  001032                        $APTHD:
    (1)  001032  000000                $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
    (1)  001034  001046                $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
    (1)  001036  000002                $TSTM:  .WORD   2       ;;RUN TIM OF LONGEST TEST
    (1)  001040  000002                $PASTM: .WORD   2       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
    (1)  001042  000000                $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
    (1)  001044  000014                        .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
   413                                 .SBTTL  APT MAILBOX-ETABLE
    (1)
    (2)                                ;;*****************************************************************
    (1)                                .EVEN
    (1)  001046                        $MAIL:                  ;;APT MAILBOX
    (1)  001046  000000                $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
    (1)  001050  000000                $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
    (1)  001052  000000                $TESTN: .WORD   ATESTN  ;;TEST NUMBER
    (1)  001054  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
    (1)  001056  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
    (1)  001060  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
    (1)  001062  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
    (1)  001064  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
    (1)  001066                        $ETABLE:                ;;APT ENVIRONMENT TABLE
    (1)  001066     000                $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
    (1)  001067     000                $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
    (1)  001070  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
    (1)  001072  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
    (1)  001074  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
    (1)                                ;*                      BITS 15-11=CPU TYPE
    (1)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
    (1)                                ;*                              11/70=06,PDQ=07,Q=10
    (1)                                ;*                      BIT 10=REAL TIME CLOCK
    (1)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
    (1)                                ;*                      BIT  8=MEMORY MANAGEMENT
    (1)  001076                        $ETEND:
    (1)                                .MEXIT
   414
   415  001076  005067  177746         START:  CLR     $FATAL                  ;CLEAR ERROR NO.
   416  001102  005067  177740                 CLR     $MSGTYP                 ;CLEAR MESSAGE TYPE (APT)
   417  001106  012767  000001  177736         MOV     #1,$TESTN               ;SET TEST NO.
   418                                 .SBTTL  INITIALIZE THE COMMON TAGS
    (1)  001114  012706  000500                 MOV     #500,SP         ;;SETUP THE STACK POINTER
    (1)                                ;;INITIALIZE A FEW VECTORS
    (1)  001120  012737  004570  000034         MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
    (1)  001126  012737  000340  000036         MOV     #340,@#TRAPVEC+2;LEVEL 7
    (1)  001134  012737  004412  000024         MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
    (1)  001142  012737  000340  000026         MOV     #340,@#PWRVEC+2 ;;LEVEL 7
    (2)                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
    (2)                                ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
    (2)  001150  013746  000004                 MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
    (2)  001154  012737  001210  000004         MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
    (2)  001162  012767  177570  177610         MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
```

```
    (2)  001170  012767  177570  177604          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
    (2)  001176  022777  177777  177574          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
    (2)  001204  001012                           BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
    (2)                                                                   ;;AND  THE HARDWARE SWR IS NOT = -1
    (2)  001206  000403                           BR      65$             ;;BRANCH IF NO TIMEOUT
    (2)  001210  012716  001216          64$:     MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
    (2)  001214  000002                           RTI
    (2)  001216  012767  000176  177554  65$:     MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
    (2)  001224  012767  000174  177550          MOV     #DISPREG,DISPLAY
    (2)  001232  012637  000004          66$:     MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
    (1)
    (2)  001236  005067  177612                   CLR     $PASS           ;;CLEAR PASS COUNT
    (2)  001242  132767  000200  177617          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
    (2)  001250  001403                           BEQ     67$             ;;YES,USE NON-APT SWITCH
    (2)  001252  012767  001070  177520          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
    (2)  001260                          67$:
  419    001260  026727  176556  002060          CMP     42,#$ENDAD              ;ACT AUTO MODE?
  420    001266  001402                           BEQ     RESTRT                  ;YIF SO: BR
  421    001270  104401                           TYPE
  422    001272  005224                           TITL
  423
  424    001274  123727  000176  000006  RESTRT: CMPB    @#176,#6                ;IS SOFTWARE SWITCH REGISTER =6?
  425    001302  001003                           BNE     1$                      ;IF NOT, TEST NORMALY
  426    001304  012767  000176  177466          MOV     #SWREG,SWR              ;IF SO USE THE SOFTWARE SWITCH REG
  427    001312  017700  177462          1$:     MOV     @SWR,R0                 ;GET SWR
  428    001316  001424                           BEQ     GETIN                   ;IF ZERO: GET INPUT
  429    001320  006300                  ST2:     ASL     R0
  430    001322  016067  004706  177470          MOV     TXLPC(R0),EXLPC         ;FETCH EXPECT. LPC
  431    001330  016067  004644  177460          MOV     TXCRC(R0),EXCRC         ;FETCH EXPECTED CRC
  432    001336  016067  004752  177442          MOV     TDLN1(R0),DATLN1        ;FETCH 1ST LENGTH
  433    001344  016067  005014  177432          MOV     TRMSA1(R0),ROMSA1       ;FETCH 1ST STARTING ADDR.
  434    001352  016067  005056  177432          MOV     TDLN2(R0),DATLN2                ;FETCH 2ND LENGTH
  435    001360  016067  005120  177422          MOV     TRMSA2(R0),ROMSA2       ;FETCH 2ND STARTING ADDR
  436    001366  000450                           BR      CHECK                   ;GO COMPUTE
  437    001370  005767  176446          GETIN:  TST     42                      ;UNDER ACT AUT ACCEPT?
  438    001374  001045                           BNE     CHECK                   ;IF SO; BR USE DEFAULT PARAMETERS
  439    001376  122767  000001  177462          CMPB    #1,$ENV                 ;UNDER APT?
  440    001404  001441                           BEQ     CHECK                   ;IF SO: BR
  441    001406  005767  177416                   TST     TYPOUT                  ;ROM TYPE OPTION
  442    001412  001012                           BNE     GET2                    ;IF SO:  BR
  443    001414  104401                           TYPE
  444    001416  005237                           GETCRC
  445    001420  104407                           RDOCT
  446    001422  012667  177370                   MOV     (SP)+,EXCRC             ;STORE EXPECT. CRC
  447    001426  104401                           TYPE                            ;TYPE LPC INPUT REQUEST
  448    001430  005263                           GETLPC
  449    001432  104407                           RDOCT
  450    001434  012667  177360                   MOV     (SP)+,EXLPC             ;STORE EXPECTED LPC
  451    001440  104401                  GET2:    TYPE
  452    001442  005443                           SA1                             ;REQUEST 1ST ADDRESS SPACE
  453    001444  104407                           RDOCT                           ;INPUT SA
  454    001446  012667  177332                   MOV     (SP)+,ROMSA1
  455    001452  104401                           TYPE                            ;REQUEST LENGTH OF 1ST ADDR. SPACE
  456    001454  005603                           SIZE1
  457    001456  104407                           RDOCT                           ;INPUT LENGTH
  458    001460  012667  177322                   MOV     (SP)+,DATLN1
```

```
                        TYPE                    ;REQUEST START ADDR. FOR 2ND SPACE
                        SA2
                        RDOCT                   ;INPUT SA
                        MOV     (SP)+,ROMSA2
                        TYPE                    ;REQUEST LENGTH OF 2ND SPACE
                        SIZE2
                        RDOCT                   ;INPUT LENGTH
                        MOV     (SP)+,DATLN2
CHECK:                  TST     TYPOUT          ;IS TYPOUT REQUESTED?
                        BEQ     1$              ;BRANCH IF NOT
                        JMP     TYPROM          ;GO TYPE OUT POM
1$:                     CLR     ACTCRC          ;CLEAR STORAGE FOR ACTUAL CRC
                        CLR     ACTLPC          ;CLEAR STORAGE FOR ACTULAL LPC
                        MOV     DATLN1,R0       ;SET LENGTH OF 1ST ROM SPACE
                        BEQ     CH0             ;IF NO VERSION SELECTED:  BR
                        MOV     ROMSA1,R1       ;POINT TO START OF 1ST ROM SPACE
                        JSR     PC,CRC          ;COMPUTE FIRST HALF OF CRC
                        MOV     ROMSA1,R1       ;POINT TO START OF 1ST ROM ADDRS.
                        MOV     DATLN1,R0       ;SET LENGTH OF 1ST ROM ADDRS.
                        ASR     R0              ;CONVERT TO WORDS
                        JSR     PC,LPC          ;COMPUTE FIRST HALF OF CRC
CH0:                    MOV     ROMSA2,R1       ;POINT TO 2ND ROM ADDRS.
                        MOV     DATLN2,R0       ;SET LENGTH OF 2ND ROM ADDRS.
                        BEQ     CH1             ;BR IF THIS SPACE NOT USED
                        JSR     PC,CRC          ;COMPUTE REMAINDER OF CRC
                        MOV     ROMSA2,R1       ;POINT TO START OF 2ND ROM ADDRS.
                        MOV     DATLN2,R0       ;SET LENGTH OF 2ND ROM ADDRS.
                        ASR     R0              ;CONVERT TO WORDS
                        JSR     PC,LPC          ;COMPUTE REMAINDER OF LPC
                        CMPB    #1,$ENV         ;UNDER APT?
                        BEQ     1$              ;IF SO:  BR
                        TST     42              ;UNDER ACT AUTO ACCEPT?
                        BEQ     CH1             ;IF NOT:  BR
                        JMP     AUTACT
CH1:                    CMP     EXCRC,ACTCRC    ;COMPUTED = EXPECTED ?
                        BEQ     CK1             ;IF SO:  BR

                        TYPE                    ;TYPE CRC ERROR MESG.
                        EXCRMG
                        MOV     EXCRC,-(SP)     ;PUT EXPECT CRC ON STACK
                        TYPOC                   ;TYPE EXPECTED CRC
                        TYPE                    ;TYPE ACTUAL CRC MESSG
                        ACCRMG
                        MOV     ACTCRC,-(SP)    ;PUT ACTUAL CRC ON STACK
                        TYPOC                   ;TYPE ACTUAL CRC
                        CMP     42,#$ENDAD      ;UNDER ACT AUTO MODE?
                        BEQ     1$              ;IF SO:  BR
                        CMPB    #1,$ENV         ;UNDER APT?
                        BNE     CK1             ;IF NOT:  BR
1$:                     MOV     #2,$FATAL       ;MOVE TO MAILBOX ERROR NO.  ****  2  ****
                        MOV     #1,$MSGTYP      ;SET MAILBOX FOR FATAL ERROR
                        HALT                    ;CRC ERROR

CK1:                    CMP     ACTLPC,EXLPC    ;COMPARE EXPT. LPC=ACTUAL LPC
                        BEQ     CK2             ;IF SO:  BR
                        TYPE                    ;TYPE LPC ERROR MESSG.
```

```
                                                EXLPMG
001750  005332                                  MOV     EXLPC,-(SP)             ;PUT EXPECTED LPC ON STACK
001752  016746  177042                          TYPOC                           ;TYPE EXPECTED LPC
001756  104402                                  TYPE                            ;TYPE ACTUAL LPC MESSG.
001760  104401                                  ACLPMG
001762  005401                                  MOV     ACTLPC,-(SP)            ;PUT ACTUAL LPC ON STACK
001764  016746  177034                          TYPOC                           ;TYPE ACTUAL LPC
001770  104402                                  CMP     42,#SENDAD              ;UNDER ACT AUTO MODE?
001772  026727  176044  002060                  BEQ     1$                      ;IF SO:  BR
002000  001404                                  CMPB    #1,$ENV                 ;UNDER APT?
002002  122767  000001  177056                  BNE     CK2                     ;IF NOT:  BR
002010  001007                                  MOV     #3,$FATAL       1$:     ;MOVE TO MAILBOX ERROR NO.  ****  3  ****
002012  012767  000003  177030                  MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
002020  012767  000001  177020                  HALT                            ;LPC ERROR
002026  000000
                                        CK2:    CMP     42,#SENDAD              ;ACT AUTO ACCEPT?
002030  026727  176006  002060                  BEQ     1$                      ;IF SO:  BR
002036  001402                                  TYPE                            ;TYPE END OF TEST
002040  104401                                  EOTST
002042  005424                                  INC     $PASS           1$:     ;BUMP PASS COUNT
002044  005267  177004                          MOV     @#42,R0                 ;CHECK APT
002050  013700  000042                          BEQ     GOAGIN                  ;KEEP GOING
002054  001405                                  RESET
002056  000005                          $SENDAD: JSR     PC,(R0)                 ;ACT HOOKS
002060  004710                                  NOP
002062  000240                                  NOP
002064  000240                                  NOP
002066  000240                          GOAGIN: JMP     RESTRT          ;DO AGAIN
002070  000167  177200
                                        CRC:    MOV     ACTCRC,XORS
002074  016767  176722  176712                  MOVB    (R1),R4         ;GET CHAR.
002102  111104                          CLO:    CMP     #173024,R1              ;LOCATION EFFECTED BY SWITCHES
002104  022701  173024                          BNE     CL3                     ;IF NOT:  BR
002110  001004                                  DEC     R0                      ;FIX COUNTERS
002112  005300                                  DEC     R0
002114  005300                                  TST     (R1)+                   ;FIX POINTER
002116  005721                                  BR      CLO                     ;CONTINUE
002120  000770                          CL3:    JSR     PC,PARITY               ;GO GET PARITY
002122  004767  000114                          JSR     PC,XOR                  ;XOR CHAR
002126  004767  000166                          CLC
002132  000241                                  ROR     R4                      ;ROTATE 1 POS. RIGHT
002134  006004                                  BCC     CL2                     ;IF NO CARRY:  BR
002136  103014                                  BIS     #400,R4                 ;SET BIT NINE
002140  052704  000400                          CLC
002144  000241                          CL1:    MOV     R4,R5                   ;SAVE CHAR
002146  010405                                  BIC     #177703,R5
002150  042705  177703                          COM     R5
002154  005105                                  BIC     #177703,R5
002156  042705  177703                          BIC     #74,R4
002162  042704  000074                          BIS     R5,R4
002166  050504                          CL2:    MOV     R4,XORS
002170  010467  176620                          DEC     R0
002174  005300                                  BEQ     CLLAST                  ;IF LAST CAR.:  BR
002176  001402                                  JMP     CLO                     ;GET NEXT CHAR.
002200  000167  177676                  CLLAST: MOV     XORS,R4
002204  016704  176604                          COM     XORS
002210  005167  176600
```

```
  571  002214  042767  177050  176572          BIC     #177050,XORS
  572  002222  042704  177727                  BIC     #177727,R4          ;COMPLEMENT ALL BUT BITS 3 & 5
  573  002226  050467  176562                  BIS     R4,XORS
  574  002232  016767  176556  176562          MOV     XORS,ACTCRC
  575  002240  000207                          RTS     PC
  576  002242  005067  176560          PARITY: CLR     PARCNT              ;CLEAR BIT COUNTER
  577  002246  012703  000010                  MOV     #10,R3              ;SET NO. OF BITS
  578  002252  032704  000001          CLP0:   BIT     #1,R4               ;SEE IF ONE BIT
  579  002256  001402                          BEQ     CLP1                ;IF NOT:  BR
  580  002260  005267  176542                  INC     PARCNT              ;BUMP COUNTER
  581  002264  000241                  CLP1:   CLC
  582  002266  006004                          ROR     R4                  ;ROTATE TO NEXT BIT
  583  002270  005303                          DEC     R3
  584  002272  001367                          BNE     CLP0                ;CONTINUE FOR ALL BITS
  585  002274  112104                          MOVB    (R1)+,R4
  586  002276  042704  177400                  BIC     #177400,R4
  587  002302  032767  000001  176516          BIT     #1,PARCNT           ;SEE IF ODD # OF ONE BITS
  588  002310  001002                          BNE     CLP2                ;IF SO:  BR
  589  002312  052704  000400                  BIS     #400,R4             ;SET PARITY BIT
  590  002316  000207                  CLP2:   RTS     PC                  ;EXIT
  591
  592  002320  010446                  XOR:    MOV     R4,-(SP)            ;XOR SUBROUTINE: R4 WITH XORS
  593  002322  046716  176466                  BIC     XORS,(SP)
  594  002326  040467  176462                  BIC     R4,XORS
  595  002332  052667  176456                  BIS     (SP)+,XORS
  596  002336  016704  176452                  MOV     XORS,R4
  597  002342  000207                          RTS     PC
  598
  599  002344  016767  176454  176442  LPC:     MOV     ACTLPC,XORS
  600  002352  012104                  LPC1:   MOV     (R1)+,R4
  601  002354  022701  173026                  CMP     #173026,R1          ;LOCATION EFFECTED BY SWITCHES
  602  002360  001402                          BEQ     LPC2                ;IF SO:  SKIP LOC. BY BRANCHING
  603  002362  004767  177732                  JSR     PC,XOR
  604  002366  005300                  LPC2:   DEC     R0
  605  002370  001370                          BNE     LPC1
  606  002372  016767  176416  176424          MOV     XORS,ACTLPC
  607  002400  000207                          RTS     PC
  608
  609  002402  104401                  TYPROM: TYPE                        ;TYPE HEADER
  610  002404  005751                          TYPHDR
  611  002406  016700  176372                  MOV     ROMSA1,R0           ;POINT OT 1ST ROM SPACE
  612  002412  016701  176370                  MOV     DATLN1,R1           ;PUT LENGTH IN R1
  613  002416  006201                          ASR     R1                  ;CONTERT TO WORDS
  614  002420  001402                          BEQ     TYPR1               ;BRANCH IF 1ST ROM SPACE NOT USED
  615  002422  004767  000026                  JSR     PC,TYP              ;GO TYPE 1ST ADDR. SPACE
  616  002426  016700  176356          TYPR1:  MOV     ROMSA2,R0           ;POINT TO 2ND ADDR. SPACE
  617  002432  016701  176354                  MOV     DATLN2,R1           ;PUT LENGTH IN R1
  618  002436  006201                          ASR     R1                  ;CONVERT TO WORDS
  619  002440  001402                          BEQ     ENDOT               ;BR IF 2ND ADDR. SPACE NOT USED
  620  002442  004767  000006                  JSR     PC,TYP              ;GO TYPE 2ND ADDR. SPACE
  621  002446  104401                  ENDOT:  TYPE
  622  002450  005424                          EOTST
  623  002452  000000                          HALT
  624
  625  002454  104401                  TYP:    TYPE
  626  002456  005743                          CARLF
```

```
                                        BR      TYP3
                        000003  TYP0:   BIT     #3,R0           ;ADDRESS MULTIPLE OF 4?
                                        BNE     TYP2            ;IF NOT:  BR
                                TYP3:   TYPE
                                        CARLF
                                        MOV     R0,-(SP)                ;PUT ADDRESS ON STACK
                                        TYPOC                   ;TYPE ADDR.
                                        TYPE
                                        COLON
                                TYP2:   MOV     (R0)+,-(SP)     ;PUT DATA ON STACK
                                        TYPOC                   ;TYP DATA
                                        TYPE                    ;TYPE 2 SPACES
                                        SP2
                                        DEC     R1              ;FINISHED?
                                        BNE     TYP0            ;IF NOT:  BR
                                        RTS     PC              ;RETURN
                                AUTACT: CLR     R0
                        000002  AUT1:   ADD      #2,R0          ;BUMP TALBE INDEX
                005162  177777         CMP     TMSG(R0),#-1    ;CHECKED ALL KNOWN VERSIONS?
                                        BEQ     AUTERR          ;IF SO:  BR
                004644  176254         CMP     TXCRC(R0),ACTCRC ;DOES THIS CRC AGREE?
                                        BNE     AUT1            ;IF NOT:  KEEP LOOKING
                000042  002060         CMP     @#42,#$ENDAD    ;UNDER ACT AUTO ACCEPT?
                                        BEQ     AUT3            ;IF SO:  BR
                005162  000002         MOV     TMSG(R0),AUT2   ;SET UP VERSION MESSAGE
                                        TYPE
                000000          AUT2:   0
                004644  176216  AUT3:   MOV     TXCRC(R0),EXCRC         ;SET EXPECTED CRC
                004706  176212         MOV     TXLPC(R0),EXLPC         ;SET EXPECTED LPC
                177124                 JMP     CK1             ;CHECK LPC

                                AUTERR: TYPE
                                        AUTERM
                000001  176224         MOV     #1,$FATAL               ;MOVE TO MAILBOX ERROR NO.  ****  1  ****
                000001  176214         MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
                                        HALT                    ;AUTO ACCEPT FAILED

                                .SBTTL  TTY INPUT ROUTINE
(1)
(2)                                     ;;***************************************************************
(1)     177560          $TKS:   .WORD   177560          ;;TTY KBD STATUS
(1)     177562          $TKB:   .WORD   177562          ;;TTY KBD BUFFER
(1)                             .ENABL  LSB
(1)
(1)                             .DSABL  LSB
(1)
(1)
(2)                             ;;***************************************************************
(1)                             ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                             ;*CALL:
(1)                             ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                             ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                             ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                             ;
(1)
(1)     011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
```

```
(1)  002642  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  002650  105777  177760            1$:    TSTB    @$TKS           ;;WAIT FOR
(1)  002654  100375                           BPL     1$              ;;A CHARACTER
(1)  002656  117766  177754  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  002664  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  002672  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  002700  001013                           BNE     3$              ;;BRANCH IF NO
(1)  002702  105777  177726            2$:    TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  002706  100375                           BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  002710  117746  177722                   MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  002714  042716  177600                   BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  002720  022627  000021                   CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  002724  001366                           BNE     2$              ;;IF NOT DISCARD IT
(1)  002726  000750                           BR      1$              ;;YES, RESUME
(1)  002730  026627  000004  000140    3$:    CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  002736  002407                           BLT     4$              ;;BRANCH IF YES
(1)  002740  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
(1)  002746  003003                           BGT     4$              ;;BRANCH IF YES
(1)  002750  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
(1)  002756  000002                    4$:    RTI                     ;;GO BACK TO USER
(2)                                    ;;*****************************************************************
(1)                                    ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                    ;*CALL:
(1)                                    ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
(1)                                    ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                    ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  002760  010346                    $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
(1)  002762  005046                            CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
(1)  002764  012703  003214            1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
(1)  002770  022703  003224            2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
(1)  002774  101456                           BLOS    4$              ;;BR IF YES
(1)  002776  104405                           RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
(1)  003000  112613                           MOVB    (SP)+,(R3)      ;;GET CHARACTER
(1)  003002  122713  000177            10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
(1)  003006  001022                           BNE     5$              ;;BR IF NO
(1)  003010  005716                           TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
(1)  003012  001007                           BNE     6$              ;;BR IF NO
(1)  003014  112767  000134  000170          MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  003022  104401  003212                   TYPE    ,9$
(1)  003026  012716  177777                   MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
(1)  003032  005303                    6$:    DEC     R3              ;;BACKUP BY ONE
(1)  003034  020327  003214                   CMP     R3,#$TTYIN      ;;STACK EMPTY?
(1)  003040  103434                           BLO     4$              ;;BR IF YES
(1)  003042  111367  000144                   MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)  003046  104401  003212                   TYPE    ,9$             ;;GO TYPE
(1)  003052  000746                           BR      2$              ;;GO READ ANOTHER CHAR.
(1)  003054  005716                    5$:    TST     (SP)            ;;RUBOUT KEY SET?
(1)  003056  001406                           BEQ     7$              ;;BR IF NO
(1)  003060  112767  000134  000124          MOVB    #'\,9$          ;;TYPE A BACK SLASH
(1)  003066  104401  003212                   TYPE    ,9$
(1)  003072  005016                           CLR     (SP)            ;;CLEAR THE RUBOUT KEY
(1)  003074  122713  000025            7$:    CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
(1)  003100  001003                           BNE     8$              ;;BR IF NO
(1)  003102  104401  003230                   TYPE    ,$CNTLU         ;;TYPE A CONTROL "U"
(1)  003106  000726                           BR      1$              ;;GO START OVER
```

```
(1)  003110  122713  000022          8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
(1)  003114  001011                          BNE     3$              ;;BRANCH IF NO
(1)  003116  105013                          CLRB    (R3)            ;;CLEAR THE CHARACTER
(1)  003120  104401  003225                  TYPE    ,$CRLF          ;;TYPE A "CR" & "LF"
(1)  003124  104401  003214                  TYPE    ,$TTYIN         ;;TYPE THE INPUT STRING
(1)  003130  000717                          BR      2$              ;;GO PICKUP ANOTHER CHACTER
(1)  003132  104401  003224          4$:     TYPE    ,$QUES          ;;TYPE A '?'
(1)  003136  000712                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  003140  111367  000046          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(1)  003144  104401  003212                  TYPE    ,9$
(1)  003150  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  003154  001305                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  003156  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(1)  003162  104401  003226                  TYPE    ,$LF            ;;TYPE A LINE FEED
(1)  003166  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  003170  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  003172  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  003174  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  003202  012766  003214  000004          MOV     #$TTYIN,4(SP)
(1)  003210  000002                          RTI                     ;;RETURN
(1)  003212     000          9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  003213     000                  .BYTE   0                       ;;TERMINATOR
(1)  003214  000010          $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  003224     077          $QUES:  .ASCII  "?"     ;;QUESTION MARK
(1)  003225     015          $CRLF:  .ASCII  <15>    ;CARRIAGE RETURN
(1)  003226  000012          $LF:    .ASCIZ  <12>    ;;LINE FEED
(1)  003230  052536  005015     000  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
(1)  003235     136  006507  000012  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
(1)  003242  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  003250  020075     000
(1)  003253     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
(1)  003260  036440  000040
665                                  .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)                                  ;;**************************************************************
(1)                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                  ;*CHANGE IT TO BINARY.
(1)                                  ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
(1)                                  ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
(1)                                  ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
(1)                                  ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
(1)                                  ;*CALL:
(1)                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
(1)                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1)  003264  011646          $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  003266  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
(3)  003274  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  003276  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  003300  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  003302  104406          1$:     RDLIN                           ;;READ AN ASCIZ LINE
(1)  003304  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
(1)  003306  010067  000100                  MOV     R0,5$           ;;AND SAVE IT
(1)  003312  005001                          CLR     R1              ;;CLEAR DATA WORD
(1)  003314  005002                          CLR     R2
```

READ AN OCTAL NUMBER FROM THE TTY SEQ 0020

```
(1)  003316  112046                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
(1)  003320  001420                        BEQ     3$              ;;IF ZERO GET OUT
(1)  003322  122716  000060                CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
(1)  003326  003026                        BGT     4$              ;;IS AN OCTAL DIGIT
(1)  003330  122716  000067                CMPB    #'7,(SP)
(1)  003334  002423                        BLT     4$
(1)  003336  006301                        ASL     R1              ;;*2
(1)  003340  006102                        ROL     R2
(1)  003342  006301                        ASL     R1              ;;*4
(1)  003344  006102                        ROL     R2
(1)  003346  006301                        ASL     R1              ;;*8
(1)  003350  006102                        ROL     R2
(1)  003352  042716  177770                BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
(1)  003356  062601                        ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
(1)  003360  000756                        BR      2$              ;;LOOP
(1)  003362  005726                3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
(1)  003364  010166  000012                MOV     R1,12(SP)       ;;SAVE THE RESULT
(1)  003370  010267  000026                MOV     R2,$HIOCT
(3)  003374  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  003376  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  003400  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  003402  000002                        RTI                     ;;RETURN
(1)  003404  005726                4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
(1)  003406  105010                        CLRB    (R0)            ;;SET A TERMINATOR
(1)  003410  104401                        TYPE                    ;;TYPE UP THRU THE BAD CHAR.
(1)  003412  000000                5$:     .WORD   0
(1)  003414  104401  003224                TYPE    $QUES           ;;"?" "CR" & "LF"
(1)  003420  000730                        BR      1$              ;;TRY AGAIN
(1)  003422  000000                $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
666
(1)                                .SBTTL  TYPE ROUTINE
(2)
(1)                                ;;*****************************************************************
(1)                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)                                ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)                                ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
(1)                                ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1)                                ;*
(1)                                ;*CALL:
(1)                                ;*1) USING A TRAP INSTRUCTION
(1)                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1)                                ;*OR
(1)                                ;*      TYPE
(1)                                ;*      MESADR
(1)                                ;*
(1)
(1)  003424  105767  000265        $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
(1)  003430  100002                        BPL     1$              ;;BR IF YES
(1)  003432  000000                        HALT                    ;;HALT HERE IF NO TERMINAL
(1)  003434  000430                        BR      3$              ;;LEAVE
(1)  003436  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
(1)  003440  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
(1)  003444  122767  000001  175414        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  003452  001011                        BNE     6$              ;;NO,GO CHECK FOR APT CONSOLE
(1)  003454  132767  000100  175405        BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1)  003462  001405                        BEQ     6$              ;;NO,GO CHECK FOR CONSOLE
```

```
(1)  003464  010067  000004                MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
(1)  003470  004767  000230                JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
(1)  003474  000000                61$:    .WORD   0               ;;MESSAGE ADDRESS
(1)  003476  132767  000040  175363  62$:  BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
(1)  003504  001003                        BNE     60$             ;;YES,SKIP TYPE OUT
(1)  003506  112046                2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)  003510  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
(1)  003512  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
(1)  003514  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  003516  062716  000002        3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
(1)  003522  000002                        RTI                     ;;RETURN
(1)  003524  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
(1)  003530  001430                        BEQ     8$
(1)  003532  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
(1)  003536  001006                        BNE     5$
(1)  003540  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
(1)  003542  104401                        TYPE                    ;;TYPE A CR AND LF
(1)  003544  003225                        $CRLF
(1)  003546  105067  000130                CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
(1)  003552  000755                        BR      2$              ;;GET NEXT CHARACTER
(1)  003554  004767  00005[                JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
(1)  003560  126726  00013[        6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
(1)  003564  001350                        BNE     2$              ;;IF NO GO GET NEXT CHAR.
(1)  003566  016746  0001[                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
(1)                                                                 ;;AND THE NULL CHAR.
(1)  003572  105366  0000[1        7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
(1)  003576  002770                        BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1)  003600  004767  000[2                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  003604  105367  000[2                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  003610  000770                        BR      7$              ;;LOOP
(1)
(1)                                ;HORIZONTAL TAB PROCESSOR
(1)
(1)  003612  112716  000040        8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  003616  004767  000014        9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  003622  132767  000007  000052        BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  003630  001372                        BNE     9$              ;;TAB STOP
(1)  003632  005726                        TST     (SP)+           ;;POP SPACE OFF STACK
(1)  003634  000724                        BR      2$              ;;GET NEXT CHARACTER
(1)  003636  105777  000044        $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  003642  100375                        BPL     $TYPEC
(1)  003644  116677  000002  000036        MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  003652  122766  000015  000002        CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
(1)  003660  001003                        BNE     1$              ;;BRANCH IF NO
(1)  003662  105067  000014                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  003666  000406                        BR      $TYPEX          ;;EXIT
(1)  003670  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  003676  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
(1)  003700  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  003702  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
(1)  003704  000207                $TYPEX: RTS     PC
(1)
(1)  003706  177564                $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
(1)  003710  177566                $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
(1)  003712     000                $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
(1)  003713     002                $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
```

```
(1)  003714     012                    $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
(1)  003715     000                    $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
667                                    .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                    ;;**************************************************************************
(1)  003716  112767  000001  000236   $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  003724  112767  000001  000226   $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  003732  000403                            BR      $ATYC
(1)  003734  112767  000001  000220   $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  003742                            $ATYC:
(3)  003742  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  003744  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  003746  105767  000206                    TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  003752  001450                            BEQ     5$              ;;IF NOT:  BR
(1)  003754  122767  000001  175104           CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  003762  001031                            BNE     3$              ;;IF NOT:  BR
(1)  003764  132767  000100  175075           BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  003772  001425                            BEQ     3$              ;;IF NOT:  BR
(1)  003774  017600  000004                    MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  004000  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  004006  005767  175034            1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  004012  001375                            BNE     1$              ;;IF NOT:  WAIT
(1)  004014  010067  175042                    MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  004020  105720                    2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  004022  001376                            BNE     2$
(1)  004024  166700  175032                    SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  004030  006200                            ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  004032  010067  175026                    MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  004036  012767  000004  175002           MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  004044  000413                            BR      5$
(1)  004046  017667  000004  000016    3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  004054  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  004062  016746  173710                    MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  004066  004767  177332                    JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  004072  000000                    4$:     .WORD   0
(1)  004074                            5$:
(1)  004074  105767  000062            10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  004100  001416                            BEQ     12$             ;;IF NOT:  BR
(1)  004102  005767  174760                    TST     $ENV            ;;RUNNING UNDER APT?
(1)  004106  001413                            BEQ     12$             ;;IF NOT:  BR
(1)  004110  005767  174732            11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  004114  001375                            BNE     11$             ;;IF NOT:  WAIT
(1)  004116  017667  000004  174724           MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  004124  062766  000002  000004           ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  004132  005267  174710                    INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  004136  105067  000020            12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  004142  105067  000013                    CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  004146  105067  000006                    CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  004152  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  004154  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  004156  000207                            RTS     PC              ;;RETURN
(1)  004160     000            $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  004161     000            $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  004162     000            $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          004164                            .EVEN
(1)          000200            APTSIZE=200
```

```
(1)          000001                 APTENV=001
(1)          000100                 APTSPOOL=100
(1)          000040                 APTCSUP=040
668                                 .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                                 ;;*****************************************************************
(1)                                 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                                 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                                 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                                 ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                                 ;*      .BYTE   M               ;;M=1 OR 0
(1)                                 ;*                                      ;;1=TYPE LEADING ZEROS
(1)                                 ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                                 ;*
(1)                                 ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                                 ;*$TYPOS OR $TYPOC
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                                 ;*
(1)                                 ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                                 ;*CALL:
(1)                                 ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                                 ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  004164  017646  000000         $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  004170  116667  000001  000211         MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  004176  112667  000207                 MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  004202  062716  000002                 ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  004206  000406                         BR      $TYPON
(1)  004210  112767  000001  000171 $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
(1)  004216  112767  000006  000165         MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
(1)  004224  112767  000005  000154 $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
(1)  004232  010346                         MOV     R3,-(SP)        ;;SAVE R3
(1)  004234  010446                         MOV     R4,-(SP)        ;;SAVE R4
(1)  004236  010546                         MOV     R5,-(SP)        ;;SAVE R5
(1)  004240  116704  000145                 MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  004244  005404                         NEG     R4
(1)  004246  062704  000006                 ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  004252  110467  000132                 MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  004256  116704  000125                 MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  004262  016605  000012                 MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  004266  005003                         CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  004270  006105                 1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  004272  000404                         BR      3$              ;;GO DO MSB
(1)  004274  006105                 2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  004276  006105                         ROL     R5
(1)  004300  006105                         ROL     R5
(1)  004302  010503                         MOV     R5,R3
(1)  004304  006103                 3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  004306  105367  000076                 DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  004312  100016                         BPL     7$              ;;BR IF NO
(1)  004314  042703  177770                 BIC     #177770,R3      ;;GET RID OF JUNK
```

```
(1)  004320  001002                          BNE     4$              ;;TEST FOR 0
(1)  004322  005704                          TST     R4              ;;SUPPRESS THIS 0?
(1)  004324  001403                          BEQ     5$              ;;BR IF YES
(1)  004326  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
(1)  004330  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
(1)  004334  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
(1)  004340  110367  000040                  MOVB    R3,8$           ;;SAVE FOR TYPING
(1)  004344  104401  004404                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
(1)  004350  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
(1)  004354  003347                          BGT     2$              ;;BR IF MORE TO DO
(1)  004356  002402                          BLT     6$              ;;BR IF DONE
(1)  004360  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
(1)  004362  000744                          BR      2$              ;;GO DO THE LAST DIGIT
(1)  004364  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
(1)  004366  012604                          MOV     (SP)+,R4        ;;RESTORE R4
(1)  004370  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  004372  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
(1)  004400  012616                          MOV     (SP)+,(SP)
(1)  004402  000002                          RTI                     ;;RETURN
(1)  004404     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
(1)  004405     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
(1)  004406     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
(1)  004407     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
(1)  004410  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 669                                 .SBTTL  POWER DOWN AND UP ROUTINES
(1)
(2)                                  ;;**************************************************************************
(1)                                  ;POWER DOWN ROUTINE
(1)  004412  012737  004552  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
(1)  004420  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
(3)  004426  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  004430  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  004432  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  004434  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  004436  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
(3)  004440  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(3)  004442  017746  174332                  MOV     @$WR,-(SP)      ;;PUSH @$WR ON STACK
(1)  004446  010667  000104                  MOV     SP,$SAVR6       ;;SAVE SP
(1)  004452  012737  004464  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1)  004460  000000                          HALT
(1)  004462  000776                          BR      .-2             ;;HANG UP
(1)
(2)                                  ;;**************************************************************************
(1)                                  ;POWER UP ROUTINE
(1)  004464  012737  004552  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1)  004472  016706  000060                  MOV     $SAVR6,SP       ;;GET SP
(1)  004476  005067  000054                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
(1)  004502  005267  000050          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
(1)  004506  001375                          BNE     1$              ;;OF  WORD
(3)  004510  012677  174264                  MOV     (SP)+,@$WR      ;;POP STACK INTO @$WR
(3)  004514  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  004516  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
(3)  004520  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  004522  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  004524  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  004526  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
```

```
(1)  004530  012737  004412  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1)  004536  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
(1)  004544  104401                          TYPE                     ;REPORT THE POWER FAILURE
(1)  004546  004560                $PWRMG:   .WORD   $POWER           ;;POWER FAIL MESSAGE POINTER
(1)  004550  000002                          RTI
(1)  004552  000000                $ILLUP:   HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
(1)  004554  000776                          BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
(1)  004556  000000                $SAVR6:   0                        ;;PUT THE SP HERE
(1)  004560  005015  047520  042527 $POWER:  .ASCIZ  <15><12>"POWER"
(1)  004566  000122
(1)                                          .EVEN
670                                .SBTTL  TRAP DECODER
(1)
(2)                                ;;**********************************************************************
(1)                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
(1)                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
(1)                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
(1)                                ;*GO TO THAT ROUTINE.
(1)
(1)  004570  010046                $TRAP:    MOV     R0,-(SP)         ;;SAVE R0
(1)  004572  016600  000002                  MOV     2(SP),R0         ;;GET TRAP ADDRESS
(1)  004576  005740                          TST     -(R0)            ;;BACKUP BY 2
(1)  004600  111000                          MOVB    (R0),R0          ;;GET RIGHT BYTE OF TRAP
(1)  004602  006300                          ASL     R0               ;;POSITION FOR INDEXING
(1)  004604  016000  004624                  MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
(1)  004610  000200                          RTS     R0               ;;GO TO ROUTINE
(1)
(1)
(1)                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
(1)
(1)  004612  011646                $TRAP2:   MOV     (SP),-(SP)       ;;MOVE THE PC DOWN
(1)  004614  016666  000004  000002          MOV     4(SP),2(SP)      ;;MOVE THE PSW DOWN
(1)  004622  000002                          RTI                      ;;RESTORE THE PSW
(1)
(3)                                .SBTTL  TRAP TABLE
(3)
(3)                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
(3)                                ;*BY THE "TRAP" INSTRUCTION.
(3)
(3)                                ;       ROUTINE
(3)                                ;       -------
(3)  004624  004612                $TRPAD:   .WORD   $TRAP2
(3)  004626  003424                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
(3)  004630  004210                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
(3)  004632  004164                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
(3)  004634  004224                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
(1)
(1)
(3)  004636  002640                          $RDCHR  ;;CALL=RDCHR    TRAP+5(104405)  TTY TYPEIN CHARACTER ROUTINE
(3)  004640  002760                          $RDLIN  ;;CALL=RDLIN    TRAP+6(104406)  TTY TYPEIN STRING ROUTINE
(3)  004642  003264                          $RDOCT  ;;CALL=RDOCT    TRAP+7(104407)  READ AN OCTAL NUMBER FROM TTY
671
672  004644  177777                $TXCRC:   -1
673  004646  000571                          571             ;M9301 - YA VERSION
674  004650  000457                          457             ;M9301 - YB VERSION
675  004652  000243                          243             ;M9301 - YC VERSION
```

```
 676  004654  000635                          635          ;M9400 - YA(OR YC) VERSION
 677  004656  000207                          207          ;M9301 - YF VERSION
 678  004660  000670                          670          ;M7942 - YB VERSION
 679  004662  000132                          132          ;M9301- YD VERSION
 680  004664  000374                          374          ;M9400 - YH (OR YK) VERSION
 681  004666  000533                          533          ;M9311 VERSION
 682  004670  000536                          536          ;M9301 - YH VERSION
 683  004672  177777                          -1
 684  004674  177777                          -1
 685  004676  177777                          -1
 686  004700  177777                          -1
 687  004702  177777                          -1
 688  004704  177777                          -1
 689
 690  004706  177777                TXLPC:    -1
 691  004710  133725                          133725       ;M9301 - YA VERSION
 692  004712  017563                          17563        ;M9301 - YB VERSION
 693  004714  141744                          141744       ;M9301 - YC VERSION
 694  004716  047613                          47613        ;M9400 - YA(OR YC) VERSION
 695  004720  114175                          114175       ;M9301 - YF VERSION
 696  004722  146126                          146126       ;M7942 - YB VERSION
 697  004724  132161                          132161       ;M9301 - YD VERSION
 698  004726  143466                          143466       ;M9400 - YH(OR YK) VERSION
 699  004730  036104                          036104       ;M9311 VERSION
 700  004732  125411                          125411       ;M9301 - YH VERSION
 701  004734  177777                          -1
 702  004736  177777                          -1
 703  004740  177777                          -1
 704  004742  177777                          -1
 705  004744  177777                          -1
 706  004746  177777                          -1
 707  004750  177777                          -1
 708
 709  004752  177777                TDLN1:    -1
 710  004754  001000                          1000         ;M9301 - YA VERSION
 711  004756  001000                          1000         ;M9301 - YB VERISION
 712  004760  001000                          1000         ;M9301 - YC VERSION
 713  004762  001000                          1000         ;M9400 - YA(OR YC) VERSION
 714  004764  001000                          1000         ;M9301 - YF VERSION
 715  004766  004000                          4000         ;M7942 - YB VERSION
 716  004770  001000                          1000         ;M9301 - VD VERSION
 717  004772  001000                          1000         ;M9400 - YH(OR YK) VERSION
 718  004774  001000                          1000         ;M9311 VERSION
 719  004776  000734                          734          ;M9301 - YH VERSION
 720  005000  177777                          -1
 721  005002  177777                          -1
 722  005004  177777                          -1
 723  005006  177777                          -1
 724  005010  177777                          -1
 725  005012  177777                          -1
 726
 727  005014  177777                TRMSA1:   -1
 728  005016  173000                          173000       ;M9301 - YA VERSION
 729  005020  173000                          173000       ;M9301 - YB VERSION
 730  005022  173000                          173000       ;M9301 - YC VERSION
 731  005024  173000                          173000       ;M9400 - YA(OR YC) VERSION
```

```
   732  005026  173000                          173000          ;M9301 - YF VERSION
   733  005030  170000                          170000          ;M7942 - YB VERSION
   734  005032  173000                          173000          ;M9301 - YD VERSION
   735  005034  173000                          173000          ;M9400 - YH(OR YK) VERSION
   736  005036  163000                          163000          ;M9311 VERSION
   737  005040  173000                          173000          ;M9301 - YH VERSION
   738  005042  177777                          -1
   739  005044  177777                          -1
   740  005046  177777                          -1
   741  005050  177777                          -1
   742  005052  177777                          -1
   743  005054  177777                          -1
   744
   745  005056  177777                TDLN2:    -1
   746  005060  001000                          1000            ;M9301 - YA VERSION
   747  005062  001000                          1000            ;M9301 - YB VERSION
   748  005064  001000                          1000            ;M9301 - YC VERSION
   749  005066  001000                          1000            ;M9400 -YA(OR YC) VERSION
   750  005070  001000                          1000            ;M9301 -YF VERSION
   751  005072  000000                          0               ;M7942 - YB VERSION
   752  005074  001000                          1000            ;M9301 - YD VERSION
   753  005076  001000                          1000            ;M9400 - YH(OR YK) VERSION
   754  005100  001000                          1000            ;M9311 VERSION
   755  005102  000764                          764             ;M9301 - YH VERSION
   756  005104  177777                          -1
   757  005106  177777                          -1
   758  005110  177777                          -1
   759  005112  177777                          -1
   760  005114  177777                          -1
   761  005116  177777                          -1
   762
   763  005120  177777                TRMSA2:   -1
   764  005122  165000                          165000          ;M9301 - YA VERSION
   765  005124  165000                          165000          ;M9301 - YB VERSION
   766  005126  165000                          165000          ;M9301 - YC VERSION
   767  005130  165000                          165000          ;M9400 - YA(OR YC) VERSION
   768  005132  165000                          165000          ;M9301 - YF VERSION
   769  005134  000000                          0               ;M7942 - YB VERSION
   770
   771  005136  165000                          165000          ;M9301 - YD VERSION
   772  005140  165000                          165000          ;M9400 - YH(OR YK) VERSION
   773  005142  166000                          166000          ;M9311 VERSION
   774  005144  165000                          165000          ;M9301 - YH VERSION
   775  005146  177777                          -1
   776  005150  177777                          -1
   777  005152  177777                          -1
   778  005154  177777                          -1
   779  005156  177777                          -1
   780  005160  177777                          -1
   781
   782  005162  177777                TMSG:     -1
   783  005164  006034                          MSG1
   784  005166  006051                          MSG2
   785  005170  006066                          MSG3
   786  005172  006103                          MSG4
   787  005174  006123                          MSG5
```

```
   788  005176  006140                              MSG6            ;M7942 - YB VERSION
   789  005200  006155                              MSG7
   790  005202  006172                              MSG10
   791  005204  006216                              MSG11
   792  005206  006226                              MSG12           ;M9301 - YH VERSION
   793  005210  177777                              -1
   794  005212  177777                              -1
   795  005214  177777                              -1
   796  005216  177777                              -1
   797  005220  177777                              -1
   798  005222  177777                              -1
   799
   800
   801  005224  005015  030122  020115  TITL:   .ASCIZ  <15><12>/ROM TEST/
        005232  042524  052123     000
   802  005237     015  052012  050131  GETCRC: .ASCIZ  <15><12>/TYPE CRC VALUE:  /
        005244  020105  051103  020103
        005252  040526  052514  035105
        005260  020040     000
   803  005263     015  052012  050131  GETLPC: .ASCIZ  <15><12>/TYPE LPC VALUE:  /
        005270  020105  050114  020103
        005276  040526  052514  035105
        005304  020040     000
   804  005307     015  042412  050130  EXCRMG: .ASCIZ  <15><12>/EXPECTED CRC =  /
        005314  041505  042524  020104
        005322  051103  020103  020075
        005330  000040
   805  005332  005015  042412  050130  EXLPMG: .ASCIZ  <15><12><12>/EXPECTED LPC =  /
        005340  041505  042524  020104
        005346  050114  020103  020075
        005354  000040
   806  005356  005015  047503  050115  ACCRMG: .ASCIZ  <15><12>/COMPUTED CRC =  /
        005364  052125  042105  041440
        005372  041522  036440  020040
        005400     000
   807  005401     015  041412  046517  ACLPMG: .ASCIZ  <15><12>/COMPUTED LPC =  /
        005406  052520  042524  020104
        005414  050114  020103  020075
        005422  000040
   808  005424  005015  042412  042116  EOTST:  .ASCIZ  <15><12><12>/END OF TEST/
        005432  047440  020106  042524
        005440  052123     000
   809  005443     015  052012  050131  SA1:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 1ST ROM ADDR. SPACE:  /
        005450  020105  052123  051101
        005456  044524  043516  040440
        005464  042104  027122  047440
        005472  020106  051461  020124
        005500  047522  020115  042101
        005506  051104  020056  050123
        005514  041501  035105  020040
        005522     000
   810  005523     015  052012  050131  SA2:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 2ND ROM ADDR. SPACE:  /
        005530  020105  052123  051101
        005536  044524  043516  040440
        005544  042104  027122  047440
        005552  020106  047062  020104
```

```
                005560  047522  020115  042101
                005566  051104  020056  050123
                005574  041501  035105  020040
                005602      000
811             005603      015  052012  050131  SIZE1:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 1ST ROM ADDR. SPACE:  /
                005610  020105  042514  043516
                005616  044124  024040  054502
                005624  042524  024523  047440
                005632  020106  051461  020124
                005640  047522  020115  042101
                005646  051104  020056  050123
                005654  041501  035105  020040
                005662      000
812             005663      015  052012  050131  SIZE2:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 2ND ROM ADDR. SPACE:  /
                005670  020105  042514  043516
                005676  044124  024040  054502
                005704  042524  024523  047440
                005712  020106  047062  020104
                005720  047522  020115  042101
                005726  051104  020056  050123
                005734  041501  035105  020040
                005742      000
813             005743      015  000012          CARLF:  .ASCIZ  <15><12>
814             005746  020040      000         SP2:    .ASCIZ  /  /
815             005751      015  040412  042104  TYPHDR: .ASCIZ  <15><12>/ADDRESS                DATA/
                005756  042522  051523  020040
                005764  020040  020040  020040
                005772  020040  020040  020040
                006000  042040  052101  000101
816             006006  020072  000040          COLON:  .ASCIZ  /:  /
817             006012  005015  047125  047113  AUTERM: .ASCIZ  <15><12>/UNKNOWN MODULE /
                006020  053517  020116  047515
                006026  052504  042514  000040
818
819             006034  005015  034515  030063  MSG1:   .ASCIZ  <15><12>/M9301 - YA/
                006042  020061  020055  040531
                006050      000
820             006051      015  046412  031471  MSG2:   .ASCIZ  <15><12>/M9301 - YB/
                006056  030460  026440  054440
                006064  000102
821             006066  005015  034515  030063  MSG3:   .ASCIZ  <15><12>/M9301 - YC/
                006074  020061  020055  041531
                006102      000
822             006103      015  046412  032071  MSG4:   .ASCIZ  <15><12>/M9400 - YA,YC/
                006110  030060  026440  054440
                006116  026101  041531      000
823             006123      015  046412  031471  MSG5:   .ASCIZ  <15><12>/M9301 - YF/
                006130  030460  026440  054440
                006136  000106
824             006140  005015  033515  032071  MSG6:   .ASCIZ  <15><12>/M7942 - YB/
                006146  020062  020055  041131
                006154      000
825             006155      015  046412  031471  MSG7:   .ASCIZ  <15><12>/M9301 - YD/
                006162  030460  026440  054440
                006170  000104
826             006172  005015  034515  030064  MSG10:  .ASCIZ  <15><12>/M9400 - YH(OR YK)/
```

```
        006200  020060  020055  044131
        006206  047450  020122  045531
        006214  000051
  827   006216  005015  034515  030463  MSG11:  .ASCIZ  <15><12>/M9311/
        006224  000061
  828   006226  005015  034515  030063  MSG12:  .ASCIZ  <15><12>/M9301 - YH/
        006234  020061  020055  044131
        006242     000
  829           000001                  .END
```

```
ABASE = 000000            413
ACCRMG  005356            501     806#
ACDW1 = 000000            413
ACDW2 = 000000            413
ACLPMG  005401            519     807#
ACPUOP= 000000            413
ACTCRC  001022            406#    470*    493     502     544     574*    647
ACTLPC  001024            407#    471*    512     520     599     606*
ADDW0 = 000000            413
ADDW1 = 000000            413
ADDW10= 000000            413
ADDW11= 000000            413
ADDW12= 000000            413
ADDW13= 000000            413
ADDW14= 000000            413
ADDW15= 000000            413
ADDW2 = 000000            413
ADDW3 = 000000            413
ADDW4 = 000000            413
ADDW5 = 000000            413
ADDW6 = 000000            413
ADDW7 = 000000            413
ADDW8 = 000000            413
ADDW9 = 000000            413
ADEVCT= 000000            413
ADEVM = 000000            413
AENV  = 000000            413
AENVM = 000000            413
AFATAL= 000000            413
AMADR1= 000000            413
AMADR2= 000000            413
AMADR3= 000000            413
AMADR4= 000000            413
AMAMS1= 000000            413
AMAMS2= 000000            413
AMAMS3= 000000            413
AMAMS4= 000000            413
AMSGAD= 000000            413
AMSGLG= 000000            413
AMSGTY= 000000            413
AMTYP1= 000000            413
AMTYP2= 000000            413
AMTYP3= 000000            413
AMTYP4= 000000            413
APASS = 000000            413
APRIOR= 000000            413
APTCSU= 000040            666     667#
APTENV= 000001            666     667#
APTSIZ= 000200            418     667#
APTSPO= 000100            666     667#
ASWREG= 000000            413
ATESTN= 000000            413
AUNIT = 000000            413
AUSWR = 000000            413
AUTACT  002522            492     643#
AUTERM  006012            659     817#
```

```
AUTERR  002612            646     658#
AUT1    002524            644#    648
AUT2    002570            651*    653#
AUT3    002572            650     654#
AVECT1= 000000            413
AVECT2= 000000            413
BIT0  = 000001            383#
BIT00 = 000001            383#
BIT01 = 000002            383#
BIT02 = 000004            383#
BIT03 = 000010            383#
BIT04 = 000020            383#
BIT05 = 000040            383#
BIT06 = 000100            383#
BIT07 = 000200            383#
BIT08 = 000400            383#
BIT09 = 001000            383#
BIT1  = 000002            383#
BIT10 = 002000            383#
BIT11 = 004000            383#
BIT12 = 010000            383#
BIT13 = 020000            383#
BIT14 = 040000            383#
BIT15 = 100000            383#
BIT2  = 000004            383#
BIT3  = 000010            383#
BIT4  = 000020            383#
BIT5  = 000040            383#
BIT6  = 000100            383#
BIT7  = 000200            383#
BIT8  = 000400            383#
BIT9  = 001000            383#
BPTVEC= 000014            383#
CARLF   005743            626     631     813#
CHECK   001510            436     438     440     467#
CHO     001566            473     480#
CHI     001644            482     491     493#
CKI     001736            494     507     512#    656
CK2     002030            513     525     530#
CLLAST  002204            567     569#
CLPO    002252            578#    584
CLP1    002264            579     581#
CLP2    002316            588     590#
CLO     002102            545#    551     568
CL1     002146            559#
CL2     002170            556     565#
CL3     002122            547     552#
COLON   006006            635     816#
CR    = 000015            383#    666
CRC     002074            475     483     544#
CRLF  = 000200            383#    666
DATLN1  001006            400#    432*    458*    472     477     612
DATLN2  001012            402#    434*    466*    481     485     617
DDISP = 177570            383#    418
DISPLA  001002            398#    418*
DISPRE  000174            385#    418
```

```
DSWR   = 177570            383#    418
EMTVEC= 000030             383#
ENDOT   002446             619     621#
EOTST   005424             533     622     808#
ERRVEC= 000004             383#    418#
EXCRC   001016             404#    431#    446#    493     498     654#
EXCRMG  005307             497     804#
EXLPC   001020             405#    430#    450#    512     516     655#
EXLPMG  005332             515     805#
GETCRC  005237             444     802#
GETIN   001370             428     437#
GETLPC  005263             448     803#
GET2    001440             442     451#
GNS    = ****** U          385     670
GOAGIN  002070             536     542#
HT     = 000011            383#    666
IOTVEC= 000020             383#
LF     = 000012            383#    666
LPC     002344             479     487     599#
LPC1    002352             600#    605
LPC2    002366             602     604#
MSG1    006034             783     819#
MSG10   006172             790     826#
MSG11   006216             791     827#
MSG12   006226             792     828#
MSG2    006051             784     820#
MSG3    006066             785     821#
MSG4    006103             786     822#
MSG5    006123             787     823#
MSG6    006140             788     824#
MSG7    006155             789     825#
PARCNT  001026             408#    576#    580#    587
PARITY  002242             552     576#
PIRQ   = 177772            383#
PIRQVE= 000240             383#
PR0    = 000000            383#
PR1    = 000040            383#
PR2    = 000100            383#
PR3    = 000140            383#
PR4    = 000200            383#
PR5    = 000240            383#
PR6    = 000300            383#
PR7    = 000340            383#
PS     = 177776            383#    393#
PSW    = 177776            383#
PWRVEC= 000024             383#    418#    669#
RDCHR  = 104405            664     670#
RDLIN  = 104406            665     670#
RDOCT  = 104407            445     449     453     457     461     465     670#
RESTRT  001274             420     424#    542
RESVEC= 000010             383#
ROMSA1  001004             399#    433#    454#    474     476     611
ROMSA2  001010             401#    435#    462#    480     484     616
SA1     005443             452     809#
SA2     005523             460     810#
SIZE1   005603             456     811#
```

```
SIZE2   005663            464     812#
SP2     005746            639     814#
STACK = 001100            383#
START   001076            389     391     415#
STKLMT= 177774            383#
ST2     001320            429#
SWR     001000            397#    418*    426*    427     669#
SWREG   000176            385#    418     426
SW0   = 000001            383#
SW00  = 000001            383#
SW01  = 000002            383#
SW02  = 000004            383#
SW03  = 000010            383#
SW04  = 000020            383#
SW05  = 000040            383#
SW06  = 000100            383#
SW07  = 000200            383#
SW08  = 000400            383#
SW09  = 001000            383#
SW1   = 000002            383#
SW10  = 002000            383#
SW11  = 004000            383#
SW12  = 010000            383#
SW13  = 020000            383#
SW14  = 040000            383#
SW15  = 100000            383#
SW2   = 000004            383#
SW3   = 000010            383#
SW4   = 000020            383#
SW5   = 000040            383#
SW6   = 000100            383#
SW7   = 000200            383#
SW8   = 000400            383#
SW9   = 001000            383#
TBITVE= 000014            383#
TDLN1   004752            432     709#
TDLN2   005056            434     745#
TITL    005224            422     801#
TKVEC = 000060            383#
TMSG    005162            645     651     782#
TPVEC = 000064            383#
TRAPVE= 000034            383#    394#    418#
TRMSA1  005014            433     727#
TRMSA2  005120            435     763#
TRTVEC= 000014            383#
TXCRC   004644            431     647     654     672#
TXLPC   004706            430     655     690#
TYP     002454            615     620     625#
TYPE  = 104401            421     443     447     451     455     459     463     496     500     514     518     532     609
                          621     625     630     634     638     652     658     664     665     666     668     669     670#
TYPHDR  005751            610     815#
TYPOC = 104402            499     503     517     521     633     637     670#
TYPON = 104404            670#
TYPOS = 104403            670#
TYPOUT  001030            388*    390*    409#    441     467
TYPROM  002402            469     609#
```

```
TYPR1   002426            614     616#
TYPO    002462            628#    641
TYP2    002504            629     636#
TYP3    002470            627     630#
XOR     002320            553     592#    603
XORS    001014            403#    544#    565#    569     570#    571#    573#    574     593     594#    595#    596     599#
                          606
$APTHD  001032            412#
$ASTAT= ****** U          667
$ATYC   003742            667#
$ATY1   003716            667#
$ATY3   003724            666     667#
$ATY4   003734            667#
$CHARC  003702            666#*
$CKSWR= ****** U          670
$CMTAG= ****** U          418
$CNTLG  003235            664#
$CNTLU  003230            664#
$CPUOP  001074            413#
$CRLF   003225            664#    665     666
$DEVCT  001056            413#
$ENDAD  002060            411     419     504     522     530     538#    649
$ENV    001066            413#    439     488     506     524     666     667
$ENVM   001067            413#    418     666     667
$ETABL  001066            413#
$ETEND  001076            412     413#
$FATAL  001050            413#    415#    508#    526#    660#    667#
$FFLG   004162            667#*
$ILLC   003714            666#
$ILLS   003713            666#
$GTSWR= ****** U          670
$HIBTS  001032            412#
$HIOCT  003422            665#*
$ILLUP  004552            669#
$LF     003226            664#    665     666
$LFLG   004161            667#*
$MAIL   001046            412     413#    418     666
$MBADR  001034            412#
$MFLG   004160            667#*
$MNEW   003253            664#
$MSGAD  001062            413#    667*
$MSGLG  001064            413#    667*
$MSGTY  001046            413#    416#    509#    527#    661#    667#
$MSWR   003242            664#
$NULL   003712            666#
$OCNT   004406            668#*
$OMODE  004410            668#*
$PASS   001054            413#    418#    534#
$PASTM  001040            412#
$POWER  004560            669#
$PWRON  004412            418     669#
$PWRMG  004546            669#
$PWRUP  004464            669#
$QUES   003224            664#    665     666
$RDCHR  002640            664#    670
$RDDEC= ****** U          670
```

```
$RDLIN  002760            664#    670
$RDOCT  003264            665#    670
$RDSZ = 000010            664#
$R2A  = ****** U          670
$SAVRE= ****** U          670
$SAVR6  004556            669#*
$SETUP= 000014            386#    418     664
$STUP = 177777            386#
$SVPC = 001032            411#
$SWR  = 000000            382#    669
$SWREG  001070            413#    418
$TESTN  001052            413#    417*
$TKB    002636            664#
$TKS    002634            664#
$TPB    003710            666#*
$TPFLG  003715            666#
$TPS    003706            666#
$TRAP   004570            418     670#
$TRAP2  004612            670#
$TRP  = 000010            670#
$TRPAD  004624            670#
$TSTM   001036            412#
$TTYIN  003214            664#
$TYPBN= ****** U          670
$TYPDS= ****** U          670
$TYPE   003424            666#    667     670
$TYPEC  003636            666#
$TYPEX  003704            666#
$TYPOC  004210            668#    670
$TYPON  004224            668#    670
$TYPOS  004164            668#    670
$UNIT   001060            413#
$UNITM  001042            412#
$USWR   001072            413#
$OFILL  004407            668#*
.     = 006243            384#    385#    387#    396#    411#    412#    664#    665     666     667#    669
.$ASTA= ****** U          667
.$X   = 001032            412#
```

```
COMMEN    383#
ENDCOM    383#
ERROR     383#
ESCAPE    383#
GETPRI    383#
GETSWR    383#
MULT      383#
NEWTST    383#
POP       380#    383#    665     667     669
PUSH      380#    383#    665     667     669
REPORT    383#
SCOPE     383#
SETPRI    383#
SETTRA    670#
SETUP     380#    383#    418
SKIP      383#
SLASH     383#
SPACE     383#
STARS     380#    383#    411     412     413     664     665     666     667     668     669     670
SWRSU     383#    418#
TRMTRP    670#
TYPBIN    383#
TYPDEC    383#
TYPNAM    383#
TYPNUM    383#
TYPOCS    383#
TYPOCT    383#
TYPTXT    383#
$$ESCA    383#
$$NEWT    383#
$$SET     670#
$$SETM    418#
$$SKIP    383#
.EQUAT    380#    383
.SETUP    380#    386
.SACTI    381#    411
.SAPTB    381#    413
.SAPTH    381#    412
.SAPTY    381#    667
.SCATC    379#    385
.SPOWE    379#    669
.SRDOC    379#    665
.SREAD    379#    664
.STRAP    379#    670
.STYPE    379#    666
.STYPO    379#    668

. ABS.  006243     000

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DZM9AC.BIN,DZM9AC.LST/CRF/NL:TOC=DZM9AC.P11
 RUN-TIME:  11  4  .5 SECONDS
```

M03

RUN-TIME RATIO: 272/16=16.8
CORE USED: 20K (40 PAGES)

N03