M9301/9400

BOOTSTRAP
MD-11-DZM9A-B

EP-DZM9A-B-DL-B

FICHE 1 OF 1

DEC 1976
digital
MADE IN USA

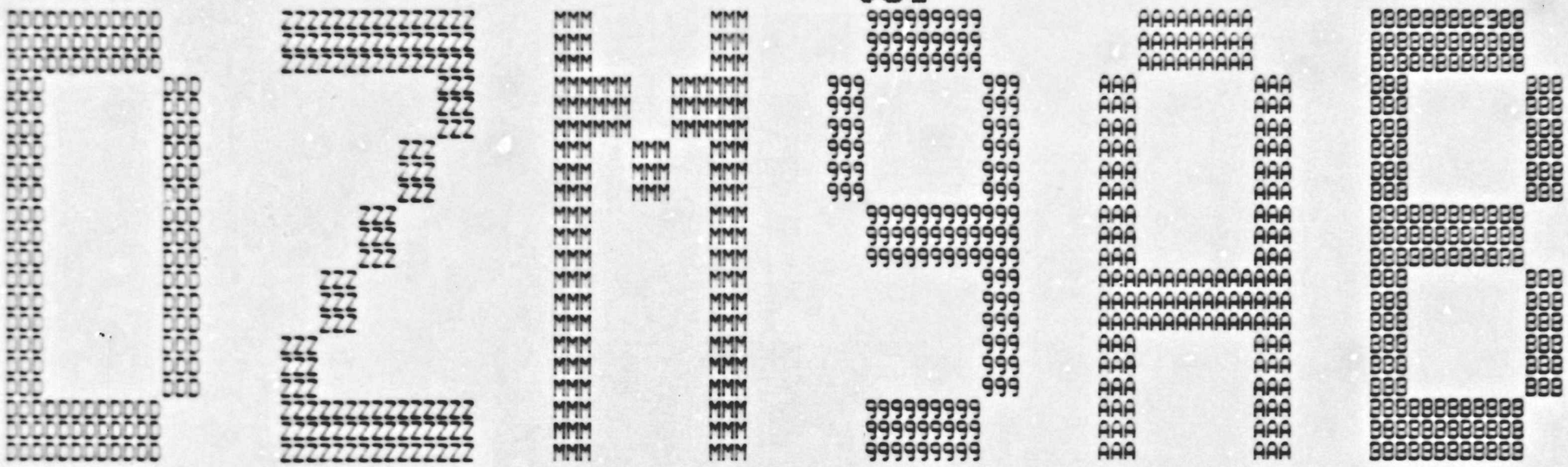
B01

```
3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012LPT
DSTART*Adder DAVIES,TOM[400,2204] 3456 DZM9AB SEQ 3410 DATE 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] *START*
D   D  A   A  V   V   I    E      S                  T     O   O  MM MM
D   D  A   A  V   V   I    E      S                  T     O   O  M M M
D   D  A   A  V   V   I    EEEE    SSS               T     O   O  M   M
D   D  AAAAA  V   V   I    E          S              T     O   O  M   M
D   D  A   A   V V    I    E          S              T     O   O  M   M
DDDD   A   A    V    III   EEEEE  SSSS    ..         T      OOO   M   M


MTO:ML21-4:DAVIES2-- distribution to ML21-4, slot 134      1     333    4   4     >
MM MM  L        2   2    11              4   4      <       11    3   3   4   4      >
M M M  L            2     1              4   4     <         1        3   4   4       >
M   M  L           2      1    -----     44444    <          1       3    44444        >
M   M  L          2       1                  4     <         1        3       4       >
M   M  L         2        1                  4      <        1    3   3       4      >
M   M  LLLLL    22222    111                 4       <      111    333        4     >


/OSTART*Adder DAVIES,TOM[400,2204] JOB 2100 SEQ 3410 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] *START*
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000111111111111111111111111111111110
000000000111111111122222222223333333333444444444455555555556666666666777777777788888888889999999999000000000011111111112222222222333312
3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012 *S
```



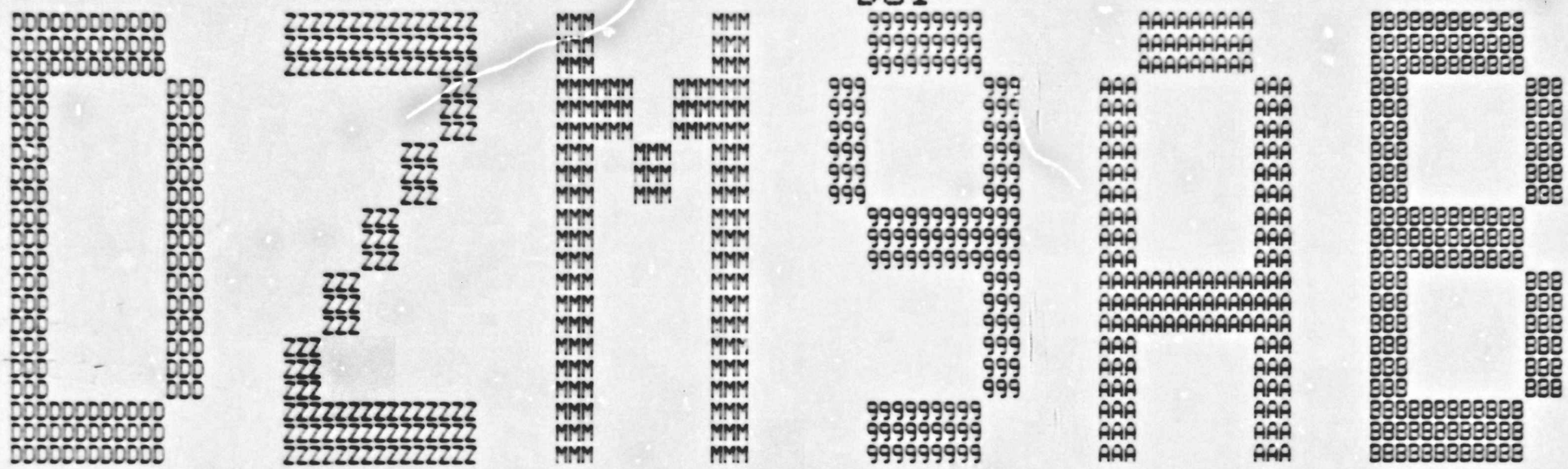
```
D   D  A   A  V   V   I    E      S                  T     O   O  MM MM
D   D  A   A  V   V   I    E      S                  T     O   O  M M M
D   D  A   A  V   V   I    EEEE    SSS               T     O   O  M   M
D   D  AAAAA  V   V   I    E          S              T     O   O  M   M
D   D  A   A   V V    I    E          S              T     O   O  M   M
DDDD   A   A    V    III   EEEEE  SSSS    ..         T      OOO   M   M


MTO:ML21-4:DAVIES2-- distribution to ML21-4, slot 134      1     333    4   4     >
MM MM  L        2   2    11              4   4      <       11    3   3   4   4      >
M M M  L            2     1              4   4     <         1        3   4   4       >
M   M  L           2      1    -----     44444    <          1       3    44444        >
M   M  L          2       1                  4     <         1        3       4       >
M   M  L         2        1                  4      <        1    3   3       4      >
M   M  LLLLL    22222    111                 4       <      111    333        4     >


/OSTART*Adder DAVIES,TOM[400,2204] JOB 2100 SEQ 3410 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] *START*
```

C01



LPTSPL Version 6(100344) Running on MTA041
 START User DAVIES,TOM [400,2704] Job DZM9AB Seg. 10 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] *START*
Request created: 03-Dec-76 09:26:29   /TO:ML21-4:DAVIES -- distribution to ML21-4, slot 134

[illegible]
File will be RENAMEd to <057> protection




```
LPTSPL Version 6(100344)  Running on MTA041
 *START* User DAVIES,TOM [400,2704] Job  DZM9AB Seq. 10 Date 03-Dec-76 09:49:19 Monitor IPC-D 602 [680] *START*
Request created: 03-Dec-76  09:26:29     /TO:ML21-4:DAVIES -- distribution to ML21-4, slot 134

GtEdE S5KZ8tDZM9A2PSE0T408R2ZOMEICEeAS&II Z8PiE57% Z$PB2I80:&1BZIMIPr1W4edF0B3SDN6A7%L09:49:28
File will be RENAMEd to <057> protection
```

```
                        000000                          .REPT 0
```

IDENTIFICATION

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

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


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

                        ;OPERATING PROCEDURE:

                        ;       1.  LOAD THE PROGRAM USING NORMAL PROCEDURES.

                        ;       2.  LOAD ADDRESS 200

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

                        ;       4.  PRESS START

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

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

1.0 GENERAL PROGRAM INFORMATION

1.1 PROGRAM PURPOSE

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

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

1.2 SYSTEM REQUIREMENTS

1.2.1 HARDWARE

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

1.2.2 SOFTWARE

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

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

1.3 RELATED DOCUMENTS AND STANDARDS

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

APT INTERFACE SPECIFICATION, REV. 13

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

NONE, HOWEVER THE CPU IS ASSUMED TO BE FUNCTIONING

1.5 FAILURE ASSUMPTIONS

THE PROCESSOR IS ASSUMED TO BE FUNCTIONING PROPERLY.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOADING

USE NORMAL PROCEDURES FOR LOADING DIAGNOSTIC PROGRAMS.

2.1.2 NORMAL START

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

2.1.3 OPTIONAL START

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

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

2.2 SPECIAL ENVIRONMENTS

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

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

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

2.3 PROGRAM OPTIONS

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

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

2.3.1 SWITCH SELECTION

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

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

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

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

2.3.2 TELETYPE DIALOG

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

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

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

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

2. TYPE LPC VALUE:
THIS REQUESTS THE 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.

6.0 LISTING

.ENDR

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

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

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

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

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

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

BASIC DEFINITIONS

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

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

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

```
.MAIN.  MACY11 27(1006)  27-SEP-76  14:46  PAGE 10
DZM9AB.P11    27-SEP-76 14:43          BASIC DEFINITIONS

                                       TKVEC=  60              ;;TTY KEYBOARD VECTOR
                                       TPVEC=  64              ;;TTY PRINTER VECTOR
                                       PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
                                               .=0
                                       .SBTTL  TRAP CATCHER

                                               .=0
                                       ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                       ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                       ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                               .=174
000174  000000                         DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                         SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                               .=200
000200  005067  000624                         CLR     TYPOUT
000204  000167  000666                         JMP     START
000210  012767  000001  000612               MOV     #1,TYPOUT
000216  000167  000654                         JMP     START

                                               PS=177776
                                               TRAPVEC=34

                                       .=1000
001000  177570                         SWR:    177570
001002  177570                         DISPLAY:        177570
001004  173000                         ROMSA1: 173000
001006  001000                         DATLN1: 512.
001010  165000                         ROMSA2: 165000
001012  001000                         DATLN2: 512.
001014  000000                         XORS:   0
001016  000000                         EXCRC:  0
001020  000000                         EXLPC:  0
001022  000000                         ACTCRC: 0
001024  000000                         ACTLPC: 0
001026  000000                         PARCNT: 0
001030  000000                         TYPOUT: 0

                                       .SBTTL  ACT11 HOOKS

                                       ;;*****************************************************************
                                       ;HOOKS REQUIRED BY ACT11
001032                                         $SVPC=.                 ;SAVE PC
000046                                         .=46
000046  002042                                $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052                                         .=52
000052  000000                               .WORD   0                 ;;2)SET LOC.52 TO ZERO
001032                                         .=$SVPC                 ;; RESTORE PC
                                       .SBTTL  APT PARAMETER BLOCK

                                       ;;*****************************************************************
                                       ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                       ;;*****************************************************************
001032                                         .$X=.   ;;SAVE CURRENT LOCATION
000024                                         .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024  000200                               200     ;;FOR APT START UP
000044                                         .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
```

```
                                        APT PARAMETER BLOCK

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

001032                                  $APTHD:
001032  000000                         $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001034  001046                         $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001036  000002                         $TSTM:  .WORD   2       ;;RUN TIM OF LONGEST TEST
001040  000002                         $PASTM: .WORD   2       ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001042  000000                         $UNITM: .WORD   0       ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001044  000014                                 .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                        .SBTTL  APT MAILBOX-ETABLE

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

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

```
                                                                ;;AND A HARDWARE DISPLAY REGISTER
                                                                ;;TRY TO REFERENCE HARDWARE SWR
                                                                ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
                                                                ;;BRANCH IF NO TIMEOUT
                                                                ;;SET UP FOR TRAP RETURN
                                                                ;;POINT TO SOFTWARE SWR
                                                                ;;RESTORE ERROR VECTOR

                                                                ;;CLEAR PASS COUNT
                                                                ;;TEST USER SIZE UNDER APT
                                                                ;;YES,USE NON-APT SWITCH
                                                                ;;NO,USE APT SWITCH REGISTER

                                                                ;ACT AUTO MODE?
                                                                ;YIF SO: BR


                                                                ;GET SWR
                                                                ;IF ZERO: GET INPUT

                                                                ;FETCH EXPECT. LPC
                                                                ;FETCH EXPECTED CRC
                                                                ;FETCH 1ST LENGTH
                                                                ;FETCH 1ST STARTING ADDR.
                                                                ;FETCH 2ND LENGTH
                                                                ;FETCH 2ND STARTING ADDR
                                                                ;GO COMPUTE
                                                                ;UNDER ACT AUT ACCEPT?
                                                                ;IF SO; BR USE DEFAULT PARAMETERS
                                                                ;UNDER APT?
                                                                ;IF SO: BR
                                                                ;ROM TYPE OPTION
                                                                ;IF SO:  BR


                                                                ;STORE EXPECT. CRC
                                                                ;TYPE LPC INPUT REQUEST


                                                                ;STORE EXPECTED LPC

                                                                ;REQUEST 1ST ADDRESS SPACE
                                                                ;INPUT SA

                                                                ;REQUEST LENGTH OF 1ST ADDR. SPACE

                                                                ;INPUT LENGTH

                                                                ;REQUEST START ADDR. FOR 2ND SPACE

                                                                ;INPUT SA
```

```
                                                MOV     (SP)+,ROMSA2
                                                TYPE
                                                SIZE2                   ;REQUEST LENGTH OF 2ND SPACE
                                                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
                                        1$:     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
                                                MOV     EXLPC,-(SP)     ;PUT EXPECTED LPC ON STACK
                                                TYPOC                   ;TYPE EXPECTED LPC
```

```
   674  001742  104401                          TYPE                                    ;TYPE ACTUAL LPC MESSG.
   675  001744  005203                          ACLPMG
   676  001746  016746  177052                  MOV     ACTLPC,-(SP)                    ;PUT ACTUAL LPC ON STACK
   677  001752  104402                          TYPOC                                   ;TYPE ACTUAL LPC
   678  001754  026727  176062  002042          CMP     42,#$ENDAD                      ;UNDER ACT AUTO MODE?
   679  001762  001404                          BEQ     1$                              ;IF SO:  BR
   680  001764  122767  000001  177074          CMPB    #1,$ENV                         ;UNDER APT?
   681  001772  001007                          BNE     CK2                             ;IF NOT:  BR
   682  001774  012767  000003  177046  1$:     MOV     #3,$FATAL                       ;MOVE TO MAILBOX ERROR NO.  ****  3  ****
   683  002002  012767  000001  177036          MOV     #1,$MSGTYP                      ;SET MAILBOX FOR FATAL ERROR
   684  002010  000000                          HALT                                    ;LPC ERROR
   685
   686  002012  026727  176024  002042  CK2:    CMP     42,#$ENDAD                      ;ACT AUTO ACCEPT?
   687  002020  001402                          BEQ     1$                              ;IF SO:  BR
   688  002022  104401                          TYPE                                    ;TYPE END OF TEST
   689  002024  005226                          EOTST
   690  002026  005267  177022          1$:     INC     $PASS                           ;BUMP PASS COUNT
   691  002032  013700  000042                  MOV     @#42,R0                         ;CHECK APT
   692  002036  001405                          BEQ     GOAGIN                          ;KEEP GOING
   693  002040  000005                          RESET
   694  002042  004710                  $ENDAD: JSR     PC,(R0)                         ;ACT HOOKS
   695  002044  000240                          NOP
   696  002046  000240                          NOP
   697  002050  000240                          NOP
   698  002052  000167  177216          GOAGIN: JMP     RESTRT          ;DO AGAIN
   699
   700  002056  016767  176740  176730  CRC:    MOV     ACTCRC,XORS
   701  002064  111104          CLO:    MOVB    (R1),R4                         ;GET CHAR.
   702  002066  022701  173024                  CMP     #173024,R1              ;LOCATION EFFECTED BY SWITCHES
   703  002072  001004                          BNE     CL3                     ;IF NOT:  BR
   704  002074  005300                          DEC     R0                      ;FIX COUNTERS
   705  002076  005300                          DEC     R0
   706  002100  005721                          TST     (R1)+                   ;FIX POINTER
   707  002102  000770                          BR      CLO                     ;CONTINUE
   708  002104  004767  000114          CL3:    JSR     PC,PARITY               ;GO GET PARITY
   709  002110  004767  000166                  JSR     PC,XOR                  ;XOR CHAR
   710  002114  000241                          CLC
   711  002116  006004                          ROR     R4                      ;ROTATE 1 POS. RIGHT
   712  002120  103014                          BCC     CL2                     ;IF NO CARRY: BR
   713  002122  052704  000400                  BIS     #400,R4                 ;SET BIT NINE
   714  002126  000241                          CLC
   715  002130  010405          CL1:    MOV     R4,R5                           ;SAVE CHAR
   716  002132  042705  177703                  BIC     #177703,R5
   717  002136  005105                          COM     R5
   718  002140  042705  177703                  BIC     #177703,R5
   719  002144  042704  000074                  BIC     #74,R4
   720  002150  050504                          BIS     R5,R4
   721  002152  010467  176636          CL2:    MOV     R4,XORS
   722  002156  005300                          DEC     R0
   723  002160  001402                          BEQ     CLLAST                  ;IF LAST CAR.:  BR
   724  002162  000167  177676                  JMP     CLO                     ;GET NEXT CHAR.
   725  002166  016704  176622          CLLAST: MOV     XORS,R4
   726  002172  005167  176616                  COM     XORS
   727  002176  042767  177050  176610          BIC     #177050,XORS
   728  002204  042704  177727                  BIC     #177727,R4              ;COMPLEMENT ALL BUT BITS 3 & 5
   729  002210  050467  176600                  BIS     R4,XORS
```

```
   730  002214  016767  176574  176600          MOV     XORS,ACTCRC
   731  002222  000207                          RTS     PC
   732  002224  005067  176576          PARITY: CLR     PARCNT          ;CLEAR BIT COUNTER
   733  002230  012703  000010                  MOV     #10,R3          ;SET NO. OF BITS
   734  002234  032704  000001          CLP0:   BIT     #1,R4           ;SEE IF ONE BIT
   735  002240  001402                          BEQ     CLP1            ;IF NOT:  BR
   736  002242  005267  176560                  INC     PARCNT          ;BUMP COUNTER
   737  002246  000241                  CLP1:   CLC
   738  002250  006004                          ROR     R4              ;ROTATE TO NEXT BIT
   739  002252  005303                          DEC     R3
   740  002254  001367                          BNE     CLP0            ;CONTINUE FOR ALL BITS
   741  002256  112104                          MOVB    (R1)+,R4
   742  002260  042704  177400                  BIC     #177400,R4
   743  002264  032767  000001  176534          BIT     #1,PARCNT       ;SEE IF ODD # OF ONE BITS
   744  002272  001002                          BNE     CLP2            ;IF SO:  BR
   745  002274  052704  000400                  BIS     #400,R4         ;SET PARITY BIT
   746  002300  000207                  CLP2:   RTS     PC              ;EXIT
   747
   748  002302  010446                  XOR:    MOV     R4,-(SP)        ;XOR SUBROUTINE: R4 WITH XOPS
   749  002304  046716  176504                  BIC     XORS,(SP)
   750  002310  040467  176500                  BIC     R4,XORS
   751  002314  052667  176474                  BIS     (SP)+,XORS
   752  002320  016704  176470                  MOV     XORS,R4
   753  002324  000207                          RTS     PC
   754
   755  002326  016767  176472  176460  LPC:    MOV     ACTLPC,XORS
   756  002334  012104                  LPC1:   MOV     (R1)+,R4
   757  002336  022701  173026                  CMP     #173026,R1      ;LOCATION EFFECTED BY SWITCHES
   758  002342  001402                          BEQ     LPC2            ;IF SO:  SKIP LOC. BY BRANCHING
   759  002344  004767  177732                  JSR     PC,XOR
   760  002350  005300                  LPC2:   DEC     R0
   761  002352  001370                          BNE     LPC1
   762  002354  016767  176434  176442          MOV     XORS,ACTLPC
   763  002362  000207                          RTS     PC
   764
   765  002364  104401                  TYPROM: TYPE                    ;TYPE HEADER
   766  002366  005553                          TYPHDR
   767  002370  016700  176410                  MOV     ROMSA1,R0       ;POINT OT 1ST ROM SPACE
   768  002374  016701  176406                  MOV     DATLN1,R1       ;PUT LENGTH IN R1
   769  002400  006201                          ASR     R1              ;CONTERT TO WORDS
   770  002402  001402                          BEQ     TYPR1           ;BRANCH IF 1ST ROM SPACE NOT USED
   771  002404  004767  000026                  JSR     PC,TYP          ;GO TYPE 1ST ADDR. SPACE
   772  002410  016700  176374          TYPR1:  MOV     ROMSA2,R0       ;POINT TO 2ND ADDR. SPACE
   773  002414  016701  176372                  MOV     DATLN2,R1       ;PUT LENGTH IN R1
   774  002420  006201                          ASR     R1              ;CONVERT TO WORDS
   775  002422  001402                          BEQ     ENDOT           ;BR IF 2ND ADDR. SPACE NOT USED
   776  002424  004767  000006                  JSR     PC,TYP          ;GO TYPE 2ND ADDR. SPACE
   777  002430  104401                  ENDOT:  TYPE
   778  002432  005226                          EOTST
   779  002434  000000                          HALT
   780
   781  002436  104401                  TYP:    TYPE
   782  002440  005545                          CARLF
   783  002442  000403                          BR      TYP3
   784  002444  032700  000003          TYP0:   BIT     #3,R0           ;ADDRESS MULTIPLE OF 4?
   785  002450  001006                          BNE     TYP2            ;IF NOT:  BR
```

```
                                        INITIALIZE THE COMMON TAGS

 786  002452  104401                    TYP3:   TYPE
 787  002454  005545                            CARLF
 788  002456  010046                            MOV     R0,-(SP)                ;PUT ADDRESS ON STACK
 789  002460  104402                            TYPOC                           ;TYPE ADDR.
 790  002462  104401                            TYPE
 791  002464  005610                            COLON
 792  002466  012046                    TYP2:   MOV     (R0)+,-(SP)             ;PUT DATA ON STACK
 793  002470  104402                            TYPOC                           ;TYP DATA
 794  002472  104401                            TYPE                            ;TYPE 2 SPACES
 795  002474  005550                            SP2
 796  002476  005301                            DEC     R1                      ;FINISHED?
 797  002500  001361                            BNE     TYP0                    ;IF NOT:  BR
 798  002502  000207                            RTS     PC                      ;RETURN
 799  002504  005000                    AUTACT: CLR     R0
 800  002506  062700  000002            AUT1:   ADD     #2,R0                   ;BUMP TALBE INDEX
 801  002512  026027  005004  177777            CMP     TMSG(R0),#-1            ;CHECKED ALL KNOWN VERSIONS?
 802  002520  001425                            BEQ     AUTERR                  ;IF SO:  BR
 803  002522  026067  004626  176272            CMP     TXCRC(R0),ACTCRC        ;DOES THIS CRC AGREE?
 804  002530  001366                            BNE     AUT1                    ;IF NOT:  KEEP LOOKING
 805  002532  023727  000042  002042            CMP     @#42,#$ENDAD            ;UNDER ACT AUTO ACCEPT?
 806  002540  001405                            BEQ     AUT3                    ;IF SO:  BR
 807  002542  016067  005004  000002            MOV     TMSG(R0),AUT2           ;SET UP VERSION MESSAGE
 808  002550  104401                            TYPE
 809  002552  000000                    AUT2:   0
 810  002554  016067  004626  176234    AUT3:   MOV     TXCRC(R0),EXCRC         ;SET EXPECTED CRC
 811  002562  016067  004650  176230            MOV     TXLPC(R0),EXLPC         ;SET EXPECTED LPC
 812  002570  000167  177124                    JMP     CK1                     ;CHECK LPC
 813
 814  002574  104401                    AUTERR: TYPE
 815  002576  005614                            AUTERM
 816  002600  012767  000001  176242            MOV     #1,$FATAL               ;MOVE TO MAILBOX ERROR NO.  ****  1  ****
 817  002606  012767  000001  176232            MOV     #1,$MSGTYP              ;SET MAILBOX FOR FATAL ERROR
 818  002614  000000                            HALT                            ;AUTO ACCEPT FAILED
 819
 820                                    .SBTTL  TTY INPUT ROUTINE
 821
 822                                    ;;**********************************************************************
 823  002616  177560                    $TKS:   .WORD   177560          ;;TTY KBD STATUS
 824  002620  177562                    $TKB:   .WORD   177562          ;;TTY KBD BUFFER
 825                                    .ENABL  LSB
 826
 827                                    .DSABL  LSB
 828
 829
 830                                    ;;**********************************************************************
 831                                    ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
 832                                    ;*CALL:
 833                                    ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
 834                                    ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
 835                                    ;*                              ;;WITH PARITY BIT STRIPPED OFF
 836                                    ;
 837
 838  002622  011646                    $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
 839  002624  016666  000004  000002            MOV     4(SP),2(SP)     ;;SAVE THE PS
 840  002632  105777  177760            1$:     TSTB    @$TKS           ;;WAIT FOR
 841  002636  100375                            BPL     1$              ;;A CHARACTER
```

```
842  002640  117766  177754  000004          MOVB    @$TKB,4(SP)         ;;READ THE TTY
843  002646  042766  177600  000004          BIC     #^C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
844  002654  026627  000004  000023          CMP     4(SP),#23           ;;IS IT A CONTROL-S?
845  002662  001013                          BNE     3$                  ;;BRANCH IF NO
846  002664  105777  177726          2$:     TSTB    @$TKS               ;;WAIT FOR A CHARACTER
847  002670  100375                          BPL     2$                  ;;LOOP UNTIL ITS THERE
848  002672  117746  177722                  MOVB    @$TKB,-(SP)         ;;GET CHARACTER
849  002676  042716  177600                  BIC     #^C177,(SP)         ;;MAKE IT 7-BIT ASCII
850  002702  022627  000021                  CMP     (SP)+,#21           ;;IS IT A CONTROL-Q?
851  002706  001366                          BNE     2$                  ;;IF NOT DISCARD IT
852  002710  000750                          BR      1$                  ;;YES, RESUME
853  002712  026627  000004  000140  3$:     CMP     4(SP),#140          ;;IS IT UPPER CASE?
854  002720  002407                          BLT     4$                  ;;BRANCH IF YES
855  002722  026627  000004  000175          CMP     4(SP),#175          ;;IS IT A SPECIAL CHAR?
856  002730  003003                          BGT     4$                  ;;BRANCH IF YES
857  002732  042766  000040  000004          BIC     #40,4(SP)           ;;MAKE IT UPPER CASE
858  002740  000002                  4$:     RTI                         ;;GO BACK TO USER
859                                  ;;*****************************************************************
860                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
861                                  ;*CALL:
862                                  ;*      RDLIN                       ;;INPUT A STRING FROM THE TTY
863                                  ;*      RETURN HERE                 ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
864                                  ;*                                  ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
865
866  002742  010346                  $RDLIN: MOV     R3,-(SP)            ;;SAVE R3
867  002744  005046                          CLR     -(SP)               ;;CLEAR THE RUBOUT KEY
868  002746  012703  003176          1$:     MOV     #$TTYIN,R3          ;;GET ADDRESS
869  002752  022703  003206          2$:     CMP     #$TTYIN+8.,R3       ;;BUFFER FULL?
870  002756  101456                          BLOS    4$                  ;;BR IF YES
871  002760  104405                          RDCHR                       ;;GO READ ONE CHARACTER FROM THE TTY
872  002762  112613                          MOVB    (SP)+,(R3)          ;;GET CHARACTER
873  002764  122713  000177          10$:    CMPB    #177,(R3)           ;;IS IT A RUBOUT
874  002770  001022                          BNE     5$                  ;;BR IF NO
875  002772  005716                          TST     (SP)                ;;IS THIS THE FIRST RUBOUT?
876  002774  001007                          BNE     6$                  ;;BR IF NO
877  002776  112767  000134  000170          MOVB    #'\,9$              ;;TYPE A BACK SLASH
878  003004  104401  003174                  TYPE    ,9$
879  003010  012716  177777                  MOV     #-1,(SP)            ;;SET THE RUBOUT KEY
880  003014  005303                  6$:     DEC     R3                  ;;BACKUP BY ONE
881  003016  020327  003176                  CMP     R3,#$TTYIN          ;;STACK EMPTY?
882  003022  103434                          BLO     4$                  ;;BR IF YES
883  003024  111367  000144                  MOVB    (R3),9$             ;;SETUP TO TYPEOUT THE DELETED CHAR.
884  003030  104401  003174                  TYPE    ,9$                 ;;GO TYPE
885  003034  000746                          BR      2$                  ;;GO READ ANOTHER CHAR.
886  003036  005716                  5$:     TST     (SP)                ;;RUBOUT KEY SET?
887  003040  001406                          BEQ     7$                  ;;BR IF NO
888  003042  112767  000134  000124          MOVB    #'\,9$              ;;TYPE A BACK SLASH
889  003050  104401  003174                  TYPE    ,9$
890  003054  005016                          CLR     (SP)                ;;CLEAR THE RUBOUT KEY
891  003056  122713  000025          7$:     CMPB    #25,(R3)            ;;IS CHARACTER A CTRL U?
892  003062  001003                          BNE     8$                  ;;BR IF NO
893  003064  104401  003212                  TYPE    ,$CNTLU             ;;TYPE A CONTROL "U"
894  003070  000726                          BR      1$                  ;;GO START OVER
895  003072  122713  000022          8$:     CMPB    #22,(R3)            ;;IS CHARACTER A "^R"?
896  003076  001011                          BNE     3$                  ;;BRANCH IF NO
897  003100  105013                          CLRB    (R3)                ;;CLEAR THE CHARACTER
```

```
.MAIN.  MACY11 27(1006)  27-SEP-76  14:46  PAGE 18
DZM9AB.P11    27-SEP-76 14:43          TTY INPUT ROUTINE

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

READ AN OCTAL NUMBER FROM THE TTY

```
                                        BGT     4$              ;;IS AN OCTAL DIGIT
003312  122716  000067                  CMPB    #'7,(SP)
003316  002423                          BLT     4$
003320  006301                          ASL     R1              ;;*2
003322  006102                          ROL     R2
003324  006301                          ASL     R1              ;;*4
003326  006102                          ROL     R2
003330  006301                          ASL     R1              ;;*8
003332  006102                          ROL     R2
003334  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
003340  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
003342  000756                          BR      2$              ;;LOOP
003344  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
003346  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
003352  010267  000026                  MOV     R2,$HIOCT
003356  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
003360  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
003362  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
003364  000002                          RTI                     ;;RETURN
003366  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
003370  105010                          CLRB    (R0)            ;;SET A TERMINATOR
003372  104401                          TYPE                    ;;TYPE UP THRU THE BAD CHAR.
003374  000000                  5$:     .WORD   0
003376  104401  003206                  TYPE    ,$QUES          ;;"?" "CR" & "LF"
003402  000730                          BR      1$              ;;TRY AGAIN
003404  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
                                .SBTTL  TYPE ROUTINE

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

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

```
1010  003460  132767  000040  175401  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
1011  003466  001003                  BNE     60$             ;;YES,SKIP TYPE OUT
1012  003470  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1013  003472  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
1014  003474  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
1015  003476  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
1016  003500  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
1017  003504  000002                  RTI                     ;;RETURN
1018  003506  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
1019  003512  001430                  BEQ     8$
1020  003514  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
1021  003520  001006                  BNE     5$
1022  003522  005726                  TST     (SP)+           ;;POP  <CR><LF> EQUIV
1023  003524  104401                  TYPE                    ;;TYPE A CR AND LF
1024  003526  003207                  $CRLF
1025  003530  105067  000130          CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
1026  003534  000755                  BR      2$              ;;GET NEXT CHARACTER
1027  003536  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
1028  003542  126726  000130          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
1029  003546  001350                  BNE     2$              ;;IF NO GO GET NEXT CHAR.
1030  003550  016746  000120          MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1031                                                          ;;AND THE NULL CHAR.
1032  003554  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
1033  003560  002770                  BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
1034  003562  004767  000032          JSR     PC,$TYPEC       ;;GO TYPE A NULL
1035  003566  105367  000072          DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
1036  003572  000770                  BR      7$              ;;LOOP
1037
1038                                  ;HORIZONTAL TAB PROCESSOR
1039
1040  003574  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
1041  003600  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
1042  003604  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
1043  003612  001372                  BNE     9$              ;;TAB STOP
1044  003614  005726                  TST     (SP)+           ;;POP SPACE OFF STACK
1045  003616  000724                  BR      2$              ;;GET NEXT CHARACTER
1046  003620  105777  000044          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
1047  003624  100375                  BPL     $TYPEC
1048  003626  116677  000002  000036          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
1049  003634  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
1050  003642  001003                  BNE     1$              ;;BRANCH IF NO
1051  003644  105067  000014          CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1052  003650  000406                  BR      $TYPEX          ;;EXIT
1053  003652  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1054  003660  001402                  BEQ     $TYPEX          ;;BRANCH IF YES
1055  003662  105227                  INCB    (PC)+           ;;COUNT THE CHARACTER
1056  003664  000000          $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
1057  003666  000207          $TYPEX: RTS     PC
1058
1059  003670  177564          $TPS:   .WORD   177564          ;;TTY PRINTER STATUS REG. ADDRESS
1060  003672  177566          $TPB:   .WORD   177566          ;;TTY PRINTER BUFFER REG. ADDRESS
1061  003674     000          $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
1062  003675     002          $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1063  003676     012          $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
1064  003677     000          $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1065                                  .SBTTL  APT COMMUNICATIONS ROUTINE
```

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

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

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

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

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

POWER DOWN AND UP ROUTINES

```
1234  004530  004542                  $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
1235  004532  000002                          RTI
1236  004534  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
1237  004536  000776                          BR      .-2             ;;  BEFORE THE POWER DOWN WAS COMPLETE
1238  004540  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
1239  004542  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
1240  004550  000122
1241                                          .EVEN
1242                                  .SBTTL  TRAP DECODER
1243
1244                                  ;;**************************************************************************
1245                                  ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1246                                  ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1247                                  ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1248                                  ;*GO TO THAT ROUTINE.
1249
1250  004552  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
1251  004554  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
1252  004560  005740                          TST     -(R0)           ;;BACKUP BY 2
1253  004562  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
1254  004564  006300                          ASL     R0              ;;POSITION FOR INDEXING
1255  004566  016000  004606                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
1256  004572  000200                          RTS     R0              ;;GO TO ROUTINE
1257
1258
1259                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
1260
1261  004574  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
1262  004576  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
1263  004604  000002                          RTI                     ;;RESTORE THE PSW
1264
1265                                  .SBTTL  TRAP TABLE
1266
1267                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
1268                                  ;*BY THE "TRAP" INSTRUCTION.
1269
1270                                  :       ROUTINE
1271                                  :       -------
1272  004606  004574                  $TRPAD: .WORD   $TRAP2
1273  004610  003406                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
1274  004612  004172                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
1275  004614  004146                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
1276  004616  004206                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
1277
1278
1279  004620  002622                          $RDCHR  ;;CALL=RDCHR    TRAP+5(104405)  TTY TYPEIN CHARACTER ROUTINE
1280  004622  002742                          $RDLIN  ;;CALL=RDLIN    TRAP+6(104406)  TTY TYPEIN STRING ROUTINE
1281  004624  003246                          $RDOCT  ;;CALL=RDOCT    TRAP+7(104407)  READ AN OCTAL NUMBER FROM TTY
1282
1283  004626  177777                  TXCRC:  -1
1284  004630  000571                          571             ;M9301 - YA VERSION
1285  004632  000457                          457             ;M9301 - YB VERSION
1286  004634  000243                          243             ;M9301 - YC VERSION
1287  004636  000635                          635             ;M9400 - YA(OR YC) VERSION
1288  004640  000420                          420             ;M9301 - YF VERSION
1289  004642  177777                          -1
```

```
 1290  004644  177777                          -1
 1291  004646  177777                          -1
 1292
 1293  004650  177777              TXLPC:      -1
 1294  004652  133725                          133725          ;M9301 - YA VERSION
 1295  004654  017563                          17563           ;M9301 - YB VERSION
 1296  004656  141744                          141744          ;M9301 - YC VERSION
 1297  004660  047613                          47613           ;M9400 - YA(ORYC) VERSION
 1298  004662  075747                          75747           ;M9301 - YF VERSION
 1299  004664  177777                          -1
 1300  004666  177777                          -1
 1301  004670  177777                          -1
 1302  004672  177777                          -1
 1303
 1304  004674  177777              TDLN1:      -1
 1305  004676  001000                          1000            ;M9301 - YA VERSION
 1306  004700  001000                          1000            ;M9301 - YB VERISION
 1307  004702  001000                          1000            ;M9301 - YC VERSION
 1308  004704  001000                          1000            ;M9400 - YA(OR YC) VERSION
 1309  004706  001000                          1000            ;M9301 - YF VERSION
 1310  004710  177777                          -1
 1311  004712  177777                          -1
 1312  004714  177777                          -1
 1313
 1314  004716  177777              TRMSA1:     -1
 1315  004720  173000                          173000          ;M9301 - YA VERSION
 1316  004722  173000                          173000          ;M9301 - YB VERSION
 1317  004724  173000                          173000          ;M9301 - YC VERSION
 1318  004726  173000                          173000          ;M9400 - YA(OR YC) VERSION
 1319  004730  173000                          173000          ;M9301 - YF VERSION
 1320  004732  177777                          -1
 1321  004734  177777                          -1
 1322  004736  177777                          -1
 1323
 1324  004740  177777              TDLN2:      -1
 1325  004742  001000                          1000            ;M9301 - YA VERSION
 1326  004744  001000                          1000            ;M9301 - YB VERSION
 1327  004746  001000                          1000            ;M9301 - YC VERSION
 1328  004750  001000                          1000            ;M9400 -YA(OR YC) VERSION
 1329  004752  001000                          1000            ;M9301 -YF VERSION
 1330  004754  177777                          -1
 1331  004756  177777                          -1
 1332  004760  177777                          -1
 1333
 1334  004762  177777              TRMSA2:     -1
 1335  004764  165000                          165000          ;M9301 - YA VERSION
 1336  004766  165000                          165000          ;M9301 - YB VERSION
 1337  004770  165000                          165000          ;M9301 - YC VERSION
 1338  004772  165000                          165000          ;M9400 - YA(OR YC) VERSION
 1339  004774  165000                          165000          ;M9301 - YF VERSION
 1340  004776  177777                          -1
 1341  005000  177777                          -1
 1342  005002  177777                          -1
 1343
 1344  005004  177777              TMSG:       -1
 1345  005006  005636                          MSG1
```

```
1346  005010  005653                                MSG2
1347  005012  005670                                MSG3
1348  005014  005705                                MSG4
1349  005016  005725                                MSG5
1350  005020  177777                                -1
1351  005022  177777                                -1
1352  005024  177777                                -1
1353
1354
1355  005026  005015  030122  020115  TITL:   .ASCIZ  <15><12>/ROM TEST/
1356  005034  042524  052123     000
1357  005041     015  052012  050131  GETCRC: .ASCIZ  <15><12>/TYPE CRC VALUE:  /
1358  005046  020105  051103  020103
1359  005054  040526  052514  035105
1360  005062  020040     000
1361  005065     015  052012  050131  GETLPC: .ASCIZ  <15><12>/TYPE LPC VALUE:  /
1362  005072  020105  050114  020103
1363  005100  040526  052514  035105
1364  005106  020040     000
1365  005111     015  042412  050130  EXCRMG: .ASCIZ  <15><12>/EXPECTED CRC =  /
1366  005116  041505  042524  020104
1367  005124  051103  020103  020075
1368  005132  000040
1369  005134  005015  042412  050130  EXLPMG: .ASCIZ  <15><12><12>/EXPECTED LPC =  /
1370  005142  041505  042524  020104
1371  005150  050114  020103  020075
1372  005156  000040
1373  005160  005015  047503  050115  ACCRMG: .ASCIZ  <15><12>/COMPUTED CRC =  /
1374  005166  052125  042105  041440
1375  005174  041522  036440  020040
1376  005202     000
1377  005203     015  041412  046517  ACLPMG: .ASCIZ  <15><12>/COMPUTED LPC =  /
1378  005210  052520  042524  020104
1379  005216  050114  020103  020075
1380  005224  000040
1381  005226  005015  042412  042116  EOTST:  .ASCIZ  <15><12><12>/END OF TEST/
1382  005234  047440  020106  042524
1383  005242  052123     000
1384  005245     015  052012  050131  SA1:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 1ST ROM ADDR. SPACE:  /
1385  005252  020105  052123  051101
1386  005260  044524  043516  040440
1387  005266  042104  027122  047440
1388  005274  020106  051461  020124
1389  005302  047522  020115  042101
1390  005310  051104  020056  050123
1391  005316  041501  035105  020040
1392  005324     000
1393  005325     015  052012  050131  SA2:    .ASCIZ  <15><12>/TYPE STARTING ADDR. OF 2ND ROM ADDR. SPACE:  /
1394  005332  020105  052123  051101
1395  005340  044524  043516  040440
1396  005346  042104  027122  047440
1397  005354  020106  047062  020104
1398  005362  047522  020115  042101
1399  005370  051104  020056  050123
1400  005376  041501  035105  020040
1401  005404     000
```

```
1402  005405     015  052012  050131  SIZE1:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 1ST ROM ADDR. SPACE:  /
1403  005412  020105  042514  043516
1404  005420  044124  024040  054502
1405  005426  042524  024523  047440
1406  005434  020106  051461  020124
1407  005442  047522  020115  042101
1408  005450  051104  020056  050123
1409  005456  041501  035105  020040
1410  005464     000
1411  005465     015  052012  050131  SIZE2:  .ASCIZ  <15><12>/TYPE LENGTH (BYTES) OF 2ND ROM ADDR. SPACE:  /
1412  005472  020105  042514  043516
1413  005500  044124  024040  054502
1414  005506  042524  024523  047440
1415  005514  020106  047062  020104
1416  005522  047522  020115  042101
1417  005530  051104  020056  050123
1418  005536  041501  035105  020040
1419  005544     000
1420  005545     015  000012          CARLF:  .ASCIZ  <15><12>
1421  005550  020040     000          SP2:    .ASCIZ  /  /
1422  005553     015  040412  042104  TYPHDR: .ASCIZ  <15><12>/ADDRESS              DATA/
1423  005560  042522  051523  020040
1424  005566  020040  020040  020040
1425  005574  020040  020040  020040
1426  005602  042040  052101  000101
1427  005610  020072  000040          COLON:  .ASCIZ  /:  /
1428  005614  005015  047125  047113  AUTERM: .ASCIZ  <15><12>/UNKNOWN MODULE /
1429  005622  053517  020116  047515
1430  005630  052504  042514  000040
1431
1432  005636  005015  034515  030063  MSG1:   .ASCIZ  <15><12>/M9301 - YA/
1433  005644  020061  020055  040531
1434  005652     000
1435  005653     015  046412  031471  MSG2:   .ASCIZ  <15><12>/M9301 - YB/
1436  005660  030460  026440  054440
1437  005666  000102
1438  005670  005015  034515  030063  MSG3:   .ASCIZ  <15><12>/M9301 - YC/
1439  005676  020061  020055  041531
1440  005704     000
1441  005705     015  046412  032071  MSG4:   .ASCIZ  <15><12>/M9400 - YA,YC/
1442  005712  030060  026440  054440
1443  005720  026101  041531     000
1444  005725     015  046412  031471  MSG5:   .ASCIZ  <15><12>/M9301 - YF/
1445  005732  030460  026440  054440
1446  005740  000106
1447          000001                  .END
```

F03


```
ABASE = 000000
ACCRMG  005160
ACDW1 = 000000
ACDW2 = 000000
ACLPMG  005203
ACPUOP= 000000
ACTCRC  001022
ACTLPC  001024
ADDW0 = 000000
ADDW1 = 000000
ADDW10= 000000
ADDW11= 000000
ADDW12= 000000
ADDW13= 000000
ADDW14= 000000
ADDW15= 000000
ADDW2 = 000000
ADDW3 = 000000
ADDW4 = 000000
ADDW5 = 000000
ADDW6 = 000000
ADDW7 = 000000
ADDW8 = 000000
ADDW9 = 000000
ADEVCT= 000000
ADEVM = 000000
AENV  = 000000
AENVM = 000000
AFATAL= 000000
AMADR1= 000000
AMADR2= 000000
AMADR3= 000000
AMADR4= 000000
AMAMS1= 000000
AMAMS2= 000000
AMAMS3= 000000
AMAMS4= 000000
AMSGAD= 000000
AMSGLG= 000000
AMSGTY= 000000
AMTYP1= 000000
AMTYP2= 000000
AMTYP3= 000000
AMTYP4= 000000
APASS = 000000
APRIOR= 000000
APTCSU= 000040
APTENV= 000001
APTSIZ= 000200
APTSPO= 000100
ASWREG= 000000
ATESTN= 000000
AUNIT = 000000
AUSWR = 000000
AUTACT  002504
AUTERM  005614
AUTERR  002574
AUT1    002506
AUT2    002552
AUT3    002554
AVECT1= 000000
AVECT2= 000000
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BPTVEC= 000014
CARLF   005545
CHECK   001472
CH0     001550
CH1     001626
CK1     001720
CK2     002012
CLLAST  002166
CLP0    002234
CLP1    002246
CLP2    002300
CL0     002064
CL1     002130
CL2     002152
CL3     002104
COLON   005610
CR    = 000015
CRC     002056
CRLF  = 000200
DATLN1  001006
DATLN2  001012
DDISP = 177570
DISPLA  001002
DISPRE  000174
DSWR  = 177570
EMTVEC= 000030
ENDOT   002430
EOTST   005226
ERRVEC= 000004
EXCRC   001016
EXCRMG  005111
EXLPC   001020
EXLPMG  005134
GETCRC  005041
GETIN   001352
GETLPC  005065
GET2    001422
GOAGIN  002052
HT    = 000011
IOTVEC= 000020
LF    = 000012
LPC     002326
LPC1    002334
LPC2    002350
MSG1    005636
MSG2    005653
MSG3    005670
MSG4    005705
MSG5    005725
PARCNT  001026
PARITY  002224
PIRQ  = 177772
PIRQVE= 000240
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
PWRVEC= 000024
RDCHR = 104405
RDLIN = 104406
RDOCT = 104407
RESTRT  001274
RESVEC= 000010
ROMSA1  001004
ROMSA2  001010
R6    =%000006
R7    =%000007
SA1     005245
SA2     005325
SIZE1   005405
SIZE2   005465
SP2     005550
STACK = 001100
START   001076
STKLMT= 177774
ST2     001302
SWR     001000
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
TBITVE= 000014
TDLN1   004674
TDLN2   004740
TITL    005026
TKVEC = 000060
TMSG    005004
TPVEC = 000064
TRAPVE= 000034
TRMSA1  004716
TRMSA2  004762
TRTVEC= 000014
TXCRC   004626
TXLPC   004650
TYP     002436
TYPE  = 104401
TYPHDR  005553
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
TYPOUT  001030
TYPROM  002364
TYPR1   002410
TYPO    002444
TYP2    002466
TYP3    002452
XOR     002302
XORS    001014
$APTHD  001032
$ATYC   003724
$ATY1   003700
$ATY3   003706
$ATY4   003716
$CHPRC  003664
$CNTLG  003217
$CNTLU  003212
$CPUOP  001074
$CRLF   003207
$DEVCT  001056
$ENDAD  002042
$ENV    001066
$ENVM   001067
$ETABL  001066
$ETEND  001076
$FATAL  001050
$FFLG   004144
$FILLC  003676
$FILLS  003675
$HIBTS  001032
$HIOCT  003404
$ILLUP  004534
$LF     003210
$LFLG   004143
$MAIL   001046
$MBADR  001034
$MFLG   004142
$MNEW   003235
$MSGAD  001062
$MSGLG  001064
$MSGTY  001046
$MSWR   003224
$NULL   003674
$OCNT   004370
$OMODE  004372
$PASS   001054
$PASTM  001040
$POWER  004542
$PWRDN  004374
```

```
.MAIN.  MACY11 27(1006)  27-SEP-76  14:46  PAGE 30
DZM9AB.P11    27-SEP-76 14:43          SYMBOL TABLE

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

. ABS.  005742     000


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DZM9AB,DZM9AB/SOL=DZM9AB
 RUN-TIME: 37 13 1 SECONDS
 RUN-TIME RATIO: 117/52=2.2
 CORE USED:  20K  (39 PAGES)
```

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

[illegible] 2410 Date 03-Dec-76 09:30:09 Monitor IPC-D 602 [680] **END**