Multi Picture System/ Picture System 2
Diagnostics Software Manual
Evans & Sutherland Internal Document

Prepared by Bill Roach

August 15, 1980

This document is intended as a guide for E&S employees or customers in the development, maintenance, or conversion of MPS and PS2 (hereafter collectively referred to as Picture System) diagnostics. Familiarity with Picture System hardware, as documented in the MPS/PS2 Hardware Reference Manual, and with assembly language software development, preferably under RSX-11M, are presupposed.

TABLE OF CONTENTS

APPENDICES

A. MIXIT Language Description

B. Machine Dependent Subroutines

C. Diagnostic Operating Systems Table

D. MIXIT Utility Subroutines

E. Diagnostics Generation Flowchart

F. LINK.DOC

G. Sample MIXIT Program, QSD002

1. INTRODUCTION AND USER'S INFORMATION

1.1. DESIGN PHILOSOPHY

a. Machine Independence

b. Standard Operator Interface

c. Specified by Design Engineer

d. Assume Minimal Hardware Configuration

e. Simple Standard Operation Combined
With Flexible Non-Standard Operation

Reference: MPS/PS2 Diag Manual, Sept. '80, Ch 1
or Nov. '78, Ch 5

1.2. SUPPORTED OPERATING SYSTEMS

a. RT-11 Single-User: fast; convenient batch mode;
can use 11/04 and floppy diskette.

b. Mapped RSX-11M: System used by most customers;
multi-user operation, but no DMA, interrupts, or
batch mode.

c. Unmapped RSX-11M: For Mastape distribution to
RSX customers; supports DMA and interrupts.

d. VAX/VMS: batch mode, DMA, and interrupts. Communicate
with PS through MPS Driver Diagnostic QIO's. In some
cases much slower than RSX-11M. These will work on
PS2, though E&S does not provide Graphics Software
for PS2 on the VAX.

e. Interdata 8/32: E&S developed the utility subroutines
and converted the diagnostics, but no longer has an
Interdata system in-house. In this and the two following
cases, Assembler source files for the desired system
may be generated under RSX or VMS.

f. NORD-10, Melcom-70: Utility subroutines developed by
OEM's in Norway and Japan respectively.

Reference: MPS/PS2 Diag Manual, Sept. '80, Ch 5 ff

1.3. OPERATION OF STANDARD DIAGNOSTICS

H - Help
M - Modify
D - Do Phases
P - Pass Count

X - Execute
S - Stop on Error
L - Loop on Error
C - Loop on Error, and Continue

1.4. INTRODUCTION TO QSDDT

General purpose tool, useful in hardware troubleshooting and diagnostic debugging. See Diag Manual Nov '78, Ch 6, or Sept '80 Ch 4. Commands not documented as of Nov '78:

Y - Sense Interrupts
! - Search Operator (P100 = Search Mask)

2. DIAGNOSTICS SOFTWARE DEVELOPMENT

2.1. MIXIT PROGRAMMING LANGUAGE

See Appendix A

2.2. DIAGNOSTIC UTILITY SUBROUTINES

2.2.1. MACHINE DEPENDENT SUBROUTINES

Contained in files PSIO.MAC, IOSUBS.MAC, QSDDT1.MAC, and RTI.MAC or their counterparts. See Appendix B for descriptions of constituent subroutines, and Appendix C for table of corresponding files.

2.2.2. MACHINE INDEPENDENT SUBROUTINES

Written in MIXIT; files commonly used by several diagnostics. See Appendix D for specifications.

ARFS.MIX,ARF2.MIX - Subrs RTWT,ARFS; ARF2 used for double-buffering.

RNDM.MIX - Subr RNDM; Random Number Generator

CODE.MIX,NCODE.MIX - Subr CODE is MNEMONIC Interpreter. NCODE.MIX respects PSTB; CODE.MIX does not.

2.3. PROGRAMMING GUIDELINES

a. Use INIT and DPCH for standard operator interface

b. Assume minimal hardware configuration.

c. Use <76,PSTB> thru <101,PSTB> for non-default parameters.

d. Use ARFS for auto-referesh.

e. Minimize number of RDPS and WRPS calls to improve performance. Better one block transfer than many single-word transfers. Avoid DMA calls, to run under Mapped RSX. To improve performance, DMA may be an option controlled by e.g. <100,PSTB>. Default should be DMA disabled.

f. Use tables and indexing profusely. Use indexed dispatch rather than chained branches. Better a 16-word table and indexed reference than four instructions.

g. Do not race with the hardware. Always assume the hardware will win, and if necessary, ensure this by a call to WAIT. If waiting 1/8 second or more does not make the program intolerably slow, always call WAIT

rather than write a timeout loop. In a Multi-user system, this will relinquish the cpu. Calling WAIT guarantees AT LEAST the specified delay. Whether WAIT is called or not, in a Multi-user system the diagnostic might be suspended at any time for any length of time.

h. Start refresh buffers on an even address, and ensure that all data segments have an even word count.

i. Never use DATA to define two character codes in one word. Use CDATA or BDATA.

j. Do not use interrupts (CINT, SINT, and DINT) except in interrupt diagnostics. At present, only QSD100, QSD117, and QSDDT do.

k. Access the interface registers via READ and WRITE. DMA transfers may only be initiated by the DMA subroutine, and will cause the task to abort under Mapped RSX.

l. No recursive routines.

m. Place constants and variables in one area of the program, in an easily searched fashion. Constants should have identifying names such as X10 for octal 10, K16 for 16K, etc.

n. All SCB addresses are obtained from PSTB after INIT has been called.

o. A table-processing technique, assuming a 4 by n table:

```
L1:  CALL    SUB1,<IX1,TAB1> ;PROCESS NEXT ENTRY IN TABLE 1
     ADD     X4,IX1          ;BUMP POINTER
     CMPA    IX1,T1SZ        ;COMPARE WITH TABLE SIZE
     BRN     L1              ;STILL NEG, LOOP
     .
     .
     SUBR    SUB1,1
     .
             Here    .1  = col. 1, present table entry
                  <1,.1> = col. 2, etc.
```

2.4. PROCEDURE FOR COMPILING AND LINKING

2.4.1. PREPROCESSING

```
>RUN $PREPR
*************: FILE.MIX
*************: FILE.PRE
```

or, for Preprocessing with indirect file,

```
>RUN $PREPRI
.............: FILE.PRC

where FILE.PRC contains (b = blank):

FILE1.MIXb
FILE1.PREb
FILE2.MIXb
FILE2.PREb
etc.
```

2.4.2. MIXIT COMPILATION

Preferred method, even for single file:

```
>RUN $MIX11 (or MIX832, etc.)
.
*@FILE.MXC
```

Where file FILE.MXC contains:

```
FILE1.MAC,TI:/L:OFF=FILE1.PRE
.
$EOF$
```

Or a method which has buss:

```
>RUN $MIX11
.
*FILE1.MAC=FILE1.PRE
(CR)
```

2.4.3. MACRO ASSEMBLY

It is possible, but not recommended, to use register names as labels, e.g.:

```
SP: DATA 0 ;A MEMORY LOC NAMED 'SP', NOT RECOMMENDED
```

To assemble such a file under RSX, use the /DS:REG switch.

The MIXIT 'STOP' instruction generates, for PDP's:

```
.MCALL .EXIT
.EXIT
```

To accomodate this in RSX, create file EXIT.MAC, containing:

```
.MACRO .EXIT
.MCALL EXIT$S
EXIT$S
.ENDM  .EXIT
```

RT11

.MACRO .EXIT
MONIT = 0
JMP @MONIT

And assemble any file which uses 'STOP' (usually just the main file), as follows.

```
>MAC FILE=EXIT,FILE
```

NOTE: This will NOT work in conjunction with /DS:REG

The MAC file generated by MIX11 is voluminous, and it is usually a waste of time and paper to obtain a listing. A symbol-table only listing, however, can be useful for debugging, and is obtained as follows:

```
>MAC FILE,FILE/NL:=FILE
```

or another case:

```
>MAC ,FILE/NL:=EXIT,FILE
```

Be sure, of course, that the LST file corresponds to the OBJ file being debugged.

2.4.4. LINKING

File LINK.DOC contains linking instructions for all distributed diagnostics. To convert it to an executable RT-11 batch file:

```
.R TECO
*ERRTLINK.TEC$YHXXHKERLINK.DOC$EWLINK.BAT$
YMX$$
```

To convert it to a TKB CMD file for Unmapped RSX (options are ASG=TI:1 and PAR=PAR14K:40000:70000):

```
>RUN $TEC/INC=8192.
*ERUNMAPD.TEC$YHXXHKERLINK.DOC$Y
EWUNMAPD.LNK$MX$$
```

Thereafter, to generate a TKB CMD file for Mapped RSX (options are ASG=TI:1 and COMMON=PSDEV0:RW):

```
>RUN $TEC/INC=8192.
*ERMAPD.TEC$YHXXHKERUNMAPD.LNK$Y
EWMAPD.LNK$MX$$
```

Options for VMS are ASG=TI:1 and ASG=MPA0:4

2.5. DIAGNOSTIC DEBUGGING TECHNIQUES

2.5.1. DEBUGGING WITH QSDDT

Examine the hardware after running the diagnostic. Is the state as expected? Examine and search memory buffers. Modify refresh buffers; reset (R); specify F0,F1; start autorefresh (G). If garbage appears on the screen, K, change F1, G to do binary search and locate the garbage.

2.5.2. DEBUGGING WITH SIMPIO

Link with SIMPIO (simulated PSIO) instead of PSIO. All PSIO calls and arguments are reported at the terminal. For read operations, the programmer may specify the received data. This is useful for verification of diagnostic error messages. Is the sequence of writes and reads as expected?

2.5.3. DEBUGGING WITH ODT

Link with ODT (RSX /DA switch). Obtain link map for global addresses and relocation bias for each module, and assembly symbol table listings for local addresses as needed. Load relocation registers (R). Set breakpoints (B). Examine variables. Abort after a breakpoint and run QSDDT to examine the Picture System.

2.5.4. TRACING CHANGES IN SOURCE FILES WITH CMP

Be slow to purge .MIX files (and quick to purge everything else and delete .PRE's). Use RSX CMP, VMS DIFF, or RT-11 SRCCOM to track the changes made.

3. DIAGNOSTICS SOFTWARE MAINTENANCE

3.1. INTRODUCTION

This chapter deals with procedures for revising distributable diagnostics, incorporating new diagnostics into the distributable group, and preparing a new diagnostics release for distribution to CE's, customers, the E&S Test Department, and other users.

3.2. DIAGNOSTIC REVISIONS

a. Verify the revisions; see Appendix G.

b. Place the new .MIX files on the DGDEVP (Diagnostic Development) RL01 in area [220,10]. Edit LINK.DOC. The version number (e.g. S02) must be bumped in the .MIX file header, in the message output by the diagnostic, in LINK.DOC, and in the writeup in the Diagnostics Manual. Also, review the writeup.

c. Record the revision for inclusion in the release memo.

3.3. DIAGNOSTIC RELEASE PROCEDURE

a. Prepare the Release Memo, summarizing all revisions and new diagnostics incorporated into the release.

b. Transfer all .MIX files, LINK.DOC, and applicable .MAC files from the DGDEVP disk to MT: to VAX [VAXMPS.DIAG.MIX]

c. Edit LINK.DOC to produce Batch file(s) to Preprocess and MIXIT compile the .MIX files into .MAC files. Delete .PRE files when done.

d. Use Batch files to assemble, link, and run all diagnostics under [VAXMPS.DIAG.NEWEXE]. Verify the operation of all diagnostics, particularly those which have been revised.

e. If any .MIX files have been revised as a result of the foregoing steps, update them on DGDEVP. Put all generated .MAC files on MT:/DO. Put them from there onto DGMAC RL02.

f. Assemble, Link, and Verify for Mapped RSX-11M. Put resultant tasks on new DLRSX RL01 [220,14] with PSDEV0, ESDIAG.CMD, etc.

g. Assemble, Link, and Verify for Unmapped RSX-11M. Put on DLRSX [220,15].

h. Copy via FLX the DGMAC files to DL:/RT. Assemble, link, and verify under RT-11, using RT-11 Diag Devp RL01.

i. Prepare the following Master Copies of distributable diagnostics: RSX MT, RK05, and RL01 (make RL02 from RL01); VMS MT and put new .EXE files into [VAXMPS.DIAG], deleting old; RT-11 RK05 (make RL01 and RL02 copies from RK05), RX02, RX01, and TU58.

j. Send Release Memo to all CE's, and begin shipping media to CE's and customers as needed.

k. On VAX, use Batch, MIX832, and Interdata Mastape Program to put .CAL files on specially formatted tape, along with .CSS files, QSI000.CAL, QSI001.CAL, QSI100.CAL, old QSD117, PSIO.CAL, IOSUBS.CAL, QSDDT1.CAL, and copies of PSIO.MAC, IOSUBS.MAC, RTI.MAC, QSDDT1.MAC. INIT.CAL and QSDDT.CAL require Interdata-specific edits. Backup on MT:/DO a copy of everything distributed to the Interdata customers.

APPENDIX A

MIXIT -- A Machine-Independent Assembly Language

MIXIT is a machine-independent assembly language which can be processed on the PDP-11 to produce an ASCII assembly language file for a target machine. The assumptions built into MIXIT about the target machine are:

1. 16-bit word machine
2. 2's complement
3. Word addressable only[1]
4. No stack operations[1]
5. No re-entrant or recursive routines[1]

Instructions for MIXIT are of the form:

```
LABL:       .INS       arg1, arg2,...        ;com
```

where:

LABL is an optional 4-character label

.INS is the MIXIT instruction (the preceding period is optional)

arg1[2],
arg2,... are the arguments required (if any) for the instruction specified (.INS). Arguments are of the form:

a or <X,a> where X is a value to be used as an index such that c(X)+a = the effective address of the the argument. a is the argument.

---

[1]Refer to the language; the target machine may have different specifications, but these will be invisible to the programmer.
[2]For a more complete description, see the section on arguments.

General Instructions

```
.MOV     a,b      ;b←a
.ADD     a,b      ;b←a+b
.ADD2    a,b      ;<b,b+1>←<a,a+1>+<b,b+1>
.SUB     a,b      ;b←b-a
.SUB2    a,b      ;<b,b+1>←<b,b+1>-<a,a+1>
.INC     a        ;a←a+1
.DEC     a        ;a←a-1
.CLR     a        ;a←0
.COM     a        ;a←~a
.AND     a,b      ;b←a·b
.OR      a,b      ;b←a∨b
.SLS     a        ;a←a*2
.SRS     a        ;a←a/2, a<15> undisturbed
.SLD     a        ;<a,a+1>←<a,a+1>*2
.SRD     a        ;<a,a+1>←<a,a+1>/2, a<15>undisturbed
```

Test and Branch Instructions

```
.CMPL    a,b      ;logically compare a to b
.CMPA    a,b      ;arithmatic compare a to b
.TST     a        ;condition ← -,0,+,≠
                  ;note condition is not set by the
                  ; general instructions
.JMP     a        ;unconditional branch to "a"
.BRZ     a        ;branch to "a" if condition 0
.BNZ     a        ;branch to "a" if condition not 0
.BRN     a        ;branch to "a" if condition negative
.BRP     a        ;branch to "a" if condition not negative
```

* Following a CMPL or CMPA instruction, condition code represents (a-b) unsigned for CMPL or two's complement for CMPA. Only CMPL, CMPA and TST set condition.

Data Storage Instructions

```
.BLOCK      n                   ;reserve n words of storage
.DATA       <a,b,c,...>         ;define data words a,b,c,...(a,b,c,...
                                ;may be names or numbers)
.CDATA      <string>            ;define character string, using
                                ;characters in the 64 ASCII set
                                ;which generate octal values 40-137
.DIFF       a,b                 ;define a word of data ←b-a (offset
                                ;in words, a and b must be names)
.BDATA      <a,b,c,d,...>       ;define numeric byte
                                ;data a,b,c,d...
                                ;this instruction packs each pair of
                                ;bytes into a data word according
                                ;to the machine-specific byte sequence.
                                ;an even number of unsigned
                                ;octal arguments is required.
```

Subrountine Instructions

```
.CALL[1]    a or a,<b,c,...>    ;call subroutine "a" with optional
                                ;arguments b,c...
.SUBR       a,n                 ;define subroutine entry point a
                                ;with n arguments (both subroutine
                                ;name and argument count are optional).
.RTRN                           ;return to calling routine
.HERE       <a,b,...>           ;define global entry points
.THERE      <a,b,...>           ;defines external globals
```

Miscellaneous Instructions

```
.LABEL      a                   ;defines label "a"
.STOP                           ;terminates execution of program
                                ;and return to monitor
.HALT                           ;stops CPU execution
.FIN                            ;end of Program Segment (Finish)
.REM        <-------->          ;Remarks--all subsequent characters
                                ;on the line are comments (this
                                ;instruction is not really necessary,
                                ;since each instruction may contain
                                ;its own comment
```

[1] Subroutines in MIXIT are not reentrant.

```
.HEAD     <--------->          ;generates a page eject directive
                               ;and supplies heading information to
                               ;the assembler of the target machine.
.NAME     ----                 ;optional title, must be the first
                               ;statement in the program if present
```

Program Test Word

When a .CMPA, .CMPL or .TST instruction is specified, the resulting zero/nonzero, positive/negative value is placed in the Program Test Word, defined at the beginning of each program segment as:

```
.HEAD       < MIXIT ASSEMBLY >
.REM        < ;PROGRAM TEST WORD >
.LABEL      TTTT[1]
.DATA       0
```

When a .BRP, .BRN, .BRZ or .BNZ instruction is given, the associated transfer of command is conditional on the contents of the Program Test Word (the PTW).

There is a unique PTW defined at the beginning of each program segment. Therefore, if a subroutine is called which is defined in a separately assembled program segment, the PTW remains undisturbed upon return to the current program segment. Note also that the current PTW is not reflected in the PTW of the external segment.

---

[1]Undefined results will occur if TTTT is used as an argument to .CMPA, .CMPL or .TST instructions.

Arguments

Except for the specific exceptions discussed in previous sections, arguments to MIXIT instructions are of five general types. Each is discussed in detail below.

1. Names -- all MIXIT names represent actual memory addresses, and may be assigned either as statement labels, or as externally-defined locations via the THERE directive. All names must be four characters or less in length, must contain only alphabetic or numeric characters, and must begin with a letter of the alphabet.

2. Numbers -- these may be in either decimal (denoted by the presence of an eight, a nine, and/or a trailing decimal point) or octal radix. They may be either positive or negative (as signified by a leading minus sign). Numbers, however, may be used only as index values (see Para. 4 below) or as constants in a DATA statement.

3. Subroutine arguments -- these are used within the bounds of a subroutine (i.e. anywhere after a SUBR directive). Such an argument consists of a period followed by a pure number, which will be interpreted in decimal radix (e.g. ".13") and which represents the ith (e.g. 13th) parameter in the parameter list of the associated CALL statement. This construct may appear wherever a name may appear (within a subroutine), except as labels, or in name- or data-defining contexts such as arguments to HERE, THERE, DIFF or DATA statements. These arguments may, of course, be used as parameters to subroutine calls to achieve further nesting of subroutine levels.

4. Indexed arguments -- when it is desired to specify an offset, in words, from a defined location or subroutine argument (for example, in the case of arrays) this construct is used. In the place of a name or subroutine argument, one writes "<arg,arg>" where the first argument may be any of the above types (name, number or subroutine argument), and signifies the offset in words; and the second argument may be either

a name or subroutine argument, and signifies the base address (i.e. the name of the array). Note that to determine the number of words in an array, the DIFF directive should be employed, rather than an execution-time subtraction of two addresses, in order to avoid address complications arising from running MIXIT on byte machines.

5. Indirect addressing -- since indirect addressing is simply a special form of indexing in which the base address is zero, the format for this construct is simply "<arg,>" where the second argument is omitted. Because absolute addresses are prohibited in MIXIT, numbers may not be used as the argument here, and although a location may contain any value, care should be taken to indirectly reference only those locations which were assigned as named locations via a DATA statement.

An example of the use of both indexing and indirect addressing appears below. This is a dispatch table and the dispatch code associated with it.

```
      MOV   <DEX,TABL>,TEMP
      JMP   <TEMP,>
TABL: DATA <RTNA,RTNB,RTNC,...>
TEMP: BLOCK 1
```

APPENDIX B
MACHINE DEPENDENT SUBROUTINES

B.1. Introduction

The following Machine Dependent Subroutines are described in this appendix. Each must be rewritten for a specific CPU and Operating System.

PSIO.MAC PICTURE SYSTEM I/O (Sec B.2.)

RSPS	RESET PS2
RSIO	RESET DIO
RSDM	RESET DMA
WRIT	PS2 INTERFACE WRITE
READ	PS2 INTERFACE READ
DMA	INITIATE DMA DATA TRANSFER
TOUT	DMA OR DIO TIMEOUT DETECTION
RDPS	READ VIA DIO
WRPS	WRITE VIA DIO
GPSA	GET DIOPSA
LPSA	LOAD DIOPSA
CINT	CONNECT INTERRUPT
DINT	DISCONNECT INTERRUPT
SINT	SENSE INTERRUPTS

IOSUBS.MAC TERMINAL I/O (Sec. B.3.)

SOCT	SEND OCTAL NUMBER
SMES	SEND MESSAGE OR STRING
GETS	GET STRING
GETN	GET OCTAL NUMBER
WAIT	DELAY N/8 SECONDS

QSDDT1.MAC STRING PARSING AND BYTE PACKING ROUTINES (Sec. B.4.)

GETC	GET NUMBER AND DELIMITER FROM STRING
BYWD	PACK TWO BYTES INTO A CG WORD

RTI.MAC REMOTE TERMINAL INTERFACE I/O (Sec. B.5.)

ROT	READ OTHER TERMINAL
WOT	WRITE OTHER TERMINAL
ROTC	CLEAR OTHER TERMINAL READ BUFFER

```
        B.2. PSIO.MAC, Picture System I/O
;
; THESE SUBROUTINES PROVIDE THE STANDARD I/O INTERFACE FOR ALL
; PICTURE SYSTEM DIAGNOSTICS TO AND FROM THE PICTURE SYSTEM.
;
;
; SUBROUTINE RSPS:
;
; THIS SUBROUTINE (RESET PS) IS CALLED TO INITIALIZE ALL PS REGISTERS
; AND I/O INTERFACE CONTROL ELEMENTS TO THEIR NORMAL POWER-UP STATE.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   RSPS
;
; SUBROUTINE RSIO:
;
; THIS SUBROUTINE RESETS THE DIRECT I/O PORTION OF THE INTERFACE
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   RSIO
;
; SUBROUTINE RSDM:
;
; THIS SUBROUTINE RESETS THE DMA PORTION OF THE INTERFACE.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   RSDM
;
;
; SUBROUTINE WRIT:
;
; THIS SUBROUTINE IS CALLED TO LOAD A CPU REGISTER.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   WRIT,<VALU,LOC>
;
; WHERE:
;
;       VALU    SPECIFIES THE VALUE TO BE WRITTEN.
;
;       LOC     SPECIFIES WHICH CPU REGISTER TO WRITE.
;                       LOC=0 FOR PSDATA
;                       LOC=1 FOR DIOPSA
;                       LOC=2 FOR DMAWC
;                       LOC=3 FOR DMABA
;                       LOC=4 FOR IOST
;
;
; SUBROUTINE READ:
```

```
;
; THIS SUBROUTINE IS CALLED TO READ A CPU REGISTER.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   READ,<LOC,VALU>
;
; WHERE:
;
;       LOC     SPECIFIES WHICH CPU REGISTER TO READ.
;                       LOC=0 FOR PSDATA
;                       LOC=1 FOR DIOPSA
;                       LOC=2 FOR DMAWC
;                       LOC=3 FOR DMABA
;                       LOC=4 FOR IOST
;
;       VALU    SPECIFIES WHERE TO STORE THE VALUE READ.
;
;
; SUBROUTINE DMA
;
; THIS SUBROUTINE IS CALLED TO TRANSFER A WORD (OR
; BLOCK OF WORDS) TO OR FROM THE PICTURE SYSTEM
; VIA THE PDP-11 DMA INTERFACE.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   DMA,<PSA,N,PDPA,XADR,MODE,WAIT>
;
; WHERE:
;
;       PSA     SPECIFIES THE PICTURE SYSTEM ADDRESS
;               WHERE THE FIRST WORD TO BE TRANSFERED
;               FROM THE PICTURE SYSTEM TO THE PDP-11
;               RESIDES OR WHERE THE FIRST WORD TRANSFERED
;               FROM THE PDP-11 TO THE PICTURE SYSTEM
;               SHOULD BE STORED.
;       N       SPECIFIES THE NUMBER OF SEQUENTIAL WORDS
;               TO BE TRANSFERED.
;       PDPA    SPECIFIES THE PDP-11 ADDRESS WHERE THE
;               FIRST WORD TO BE TRANSFERED FROM THE
;               PDP-11 TO THE PICTURE SYSTEM RESIDES OR
;               WHERE THE FIRST WORD TRANSFERED FROM THE
;               PICTURE SYSTEM TO THE PDP-11 SHOULD BE
;               STORED.
;       XADR    SPECIFIES THE TWO EXTENDED ADDRESS BITS
;               OF PDPA.
;       MODE    SPECIFIES WHETHER THE DMA WILL DO ACTIVE
;               OUTPUT, ACTIVE INPUT OR PASSIVE INPUT
;               TRANSFERS.
;               IF MODE=0 THEN THE DMA WILL DO ACTIVE OUTPUT
;               AND N SEQUENTIAL WORDS WILL BE TRANSFERED
;               FROM PDP-11 MEMORY BEGINNING AT THE ADDRESS
;       WAIT    DO YOU WANT CONTROL RETURNED TO THE USER
;               IMMEDIATELY BEFORE OR AFTER THE DMA IS
;               COMPLETED. 0 = BEFORE, 1 = AFTER.
```

```
;                SPECIFIED BY PDPA TO THE PICTURE SYSTEM.
;                IF MODE=1 THEN THE DMA WILL DO ACTIVE INPUT
;                AND N WORDS WILL BE TRANSFERED FROM THE
;                PICTURE SYSTEM TO PDP-11 MEMORY AND WILL
;                BE STORED IN SEQUENTIAL LOCATIONS BEGINNING
;                AT THE ADDRESS SPECIFIED BY PDPA.
;                IF MODE=2 THEN THE DMA WILL DO PASSIVE INPUT
;                AND N WORDS WILL BE ACCEPTED FROM THE PICTURE
;                SYSTEM AND WILL BE STORED IN SEQUENTIAL PDP-11
;                MEMORY LOCATIONS BEGINNING AT THE ADDRESS
;                SPECIFIED BY PDPA.
;        WAIT    SPECIFIES WHETHER OR NOT TO WAIT FOR DMAREADY TO
;                BE SET BEFORE RETURNING TO CALLER.
;                IF WAIT=0 THEN DMAREADY BIT WILL NOT BE CHECKED
;                BEFORE RETURN TO CALLER.
;                IF WAIT=1 THEN DMAREADY BIT WILL BE CHECKED AND
;                MUST BE SET BEFORE RETURN TO CALLER.
;
;
; SUBROUTINE TOUT:
;
; THIS SUBROUTINE (TIMEOUT) IS CALLED TO TIMEOUT, TEST THE DIO
; DMA READY BITS AND SET RESPECTIVE FLAGS IF THEY ARE SET.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   TOUT,<IRDY,DRDY>
;
; WHERE:
;
;       IRDY    IS A VARIABLE WHICH IS SET IF DIOREADY IS SET
;               AFTER THE TIMEOUT, CLEARED OTHERWISE.
;       DRDY    IS A VARIABLE WHICH IS SET IF DMAREADY IS SET
;               AFTER THE TIMEOUT, CLEARED OTHERWISE.
;
;
; SUBROUTINE RDPS:
;
; THIS SUBROUTINE (READ PS) IS CALLED TO TRANSFER A WORD (OR
; A BLOCK OF WORDS) FROM THE PICTURE SYSTEM BACK INTO THE
; PDP-11.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   RDPS,<PSA,N,PDPA,HOLD>
;
; WHERE:
;
;       PSA     SPECIFIES THE PICTURE SYSTEM MEMORY ADDRESS THAT
;               THE FIRST WORD IS TO BE READ FROM.
;       N       SPECIFIES THE NUMBER OF WORDS THAT ARE TO BE READ
;               FROM THE PICTURE SYSTEM.
;       PDPA    SPECIFIES THE PDP-11 ADDRESS WHERE THE FIRST WORD
```

```
;               READ FROM THE PICTURE SYSTEM IS TO BE WRITTEN.
;               FOR BLOCK TRANSFERS, N.CONSECUTIVE WORDS WILL BE
;               WRITTEN INTO PDP-11 MEMORY BEGINING AT THIS ADDRESS.
;       HOLD    SPECIFIES WHETHER THE PICTURE SYSTEM ADDRESS REG-
;               ISTER (DIOPSA) SHOULD BE INCREMENTED AFTER EACH READ
;               OPERATION.
;               IF HOLD=0 THEN N SEQUENTIAL WORDS WILL BE READ BE-
;               GINNING AT THE PICTURE SYSTEM MEMORY LOCATION SPEC-
;               IFIED BY PSA.
;               IF HOLD NOT=0 THEN THE CONTENTS OF PICTURE SYSTEM
;               MEMORY LOCATION SPECIFIED BY PSA WILL BE READ AND
;               TRANSFERED TO THE PDP-11 N TIMES.
;
;
; SUBROUTINE WRPS:
;
; THIS SUBROUTINE (WRITE PS) IS CALLED TO TRANSFER A WORD (OR
; A BLOCK OF WORDS) FROM THE PDP-11 TO THE PICTURE SYSTEM.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   WRPS,<PSA,N,PDPA,HOLD>
;
; WHERE:
;
;       PSA     SPECIFIES THE PICTURE SYSTEM MEMORY ADDRESS THAT
;               THE FIRST WORD IS TO BE WRITTEN INTO.
;       N       SPECIFIES THE NUMBER OF WORDS THAT ARE TO BE TRANS-
;               FERED TO THE PICTURE SYSTEM.
;       PDPA    SPECIFIES THE PDP-11 ADDRESS WHERE THE FIRST WORD
;               TO BE TRANSFERED TO THE PICTURE SYSTEM RESIDES.
;               FOR BLOCK
TRANSFERS, N CONSECUTIVE WORDS WILL BE
;               TRANSFERED FROM PDP-11 MEMORY BEGINING AT THIS
;               ADDRESS.
;       HOLD    SPECIFIES WHETHER THE PICTURE SYSTEM ADDRESS REG-
;               ISTER (DIOPSA) SHOULD BE INCREMENTED AFTER EACH
;               WRITE OPERATION.
;               IF HOLD=0 THEN N SEQUENTIAL WORDS WILL BE WRITTEN
;               BEGINNING AT THE PICTURE SYSTEM MEMORY LOCATION
;               SPECIFIED BY PSA.
;               IF HOLD NOT=0 THEN THE PICTURE SYSTEM MEMORY
;               LOCATION SPECIFIED BY PSA WILL BE WRITTEN N TIMES.
;
;
; SUBROUTINE GPSA:
;
; THIS SUBROUTINE (GET PICTURE SYSTEM ADDRESS) IS CALLED TO GET THE
; LOCATION IN PICTURE SYSTEM MEMORY THAT IS CURRENTLY BEING ADDRESSED
; BY THE PICTURE SYSEM DIRECT I/O INTERFACE (DIOPSA).
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   GPSA,<PSA>
```

```
;
; WHERE:
;
;       PSA     IS A VARIABLE IN WHICH THE CURRENT CONTENTS OF THE
;               DIOPSA REGISTER IS RETURNED.
;
; SUBROUTINE LPSA:
;
; THIS SUBROUTINE (LOAD PICTURE SYSTEM ADDRESS) IS CALLED TO LOAD THE
; PICTURE SYSTEM ADDRESS POINTER WITHOUT DOING ANY I/O
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   LPSA,<PSA>
;
; WHERE:
;
;       PSA     IS A VARIABLE WHICH WILL BE LOADED INTO THE DIOPSA
;               REGISTER.
;
;
; SUBROUTINE CINT:
;
; THIS SUBROUTINE IS CALLED TO CONNECT TO A PICTURE SYSTEM
; INTERUPT PROCESS.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   CINT,<N>
;
; WHERE:
;
;       N       SPECIFIES THE INTERRUPT PROCESS TO CONNECT TO.
;                       N=1 FOR RTC INTERRUPT
;                       N=2 FOR SYSTEM INTERRUPT
;                       N=3 FOR DEVICE INTERRUPT
;                       N=4 FOR DMA INTERRUPT
;
;
; SUBROUTINE DINT:
;
; THIS SUBROUTINE IS CALLED TO DISCONNECT FROM A PREVIOUSLY CON-
; NECTED INTERRUPT PROCESS.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL DINT,<N>
;
; WHERE:
;
;       N       SPECIFIES THE INTERRUPT PROCESS THAT IS TO BE DIS-
;               CONNECTED FROM.
;                       N=1 FOR RTC INTERRUPT
```

```
;                       N=2 FOR SYSTEM INTERRUPT
;                       N=3 FOR DEVICE INTERRUPT
;                       N=4 FOR DMA INTERRUPT
;
;
;
; SUBROUTINE SINT:
;
; THIS SUBROUTINE IS CALLED TO SENSE INTERRUPTS. A MASK IS RETURNED
; TO INDICATE WHICH TYPE OF INTERRUPT OCCURED.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL SINT,<MASK>
;
; WHERE:
;
;       MASK    SPECIFIES THE RETURN MASK.
;               BIT 0=1 RTC INTERRUPT
;               BIT 1=1 SYSTEM INTERRUPT
;               BIT 2=1 DEVICE INTERRUPT
;               BIT 3=1 DMA INTERRUPT
;
;
```

B.3. IOSUBS.MAC Terminal I/O

```
;
; THESE SUBROUTINES PROVIDE THE STANDARD I/O INTERFACE FOR
; ALL PICTURE SYSTEM DIAGNOSTICS TO ANC FROM THE TERMINAL DEVICE.
;
;
; SUBROUTINE SOCT:
;
; THIS ROUTINE IS CALLED TO OUTPUT AN OCTAL NUMBER TO THE TERMINAL
; BUFFER AND THEN TO OUTPUT THE BUFFER TO THE TERMINAL IF SPECIFIED.
; IF THE TERMINAL BUFFER IS OUTPUT, A CR AND LF ARE APPENDED TO THE
; END OF THE BUFFER.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   SOCT,<FLAG,NUM>
;
; WHERE:
;
;       FLAG    SPECIFIES WHETHER THE TERMINAL BUFFER IS TO BE
;               OUTPUT TO THE TERMINAL.
;               FLAG<0:  DO NOT OUTPUT TERMINAL BUFFER.
;               FLAG= OR >:  OUTPUT TERMINAL BUFFER.
;       NUM     IS THE BINARY NUMBER WHICH WILL BE OUTPUT AS AN
;               OCTAL NUMBER.  LEADING ZEROS WILL NOT BE OUTPUT.
;
;
; SUBROUTINE SMES:
; THIS ROUTINE IS CALLED TO OUTPUT AN ASCII STRING OF CHARACTERS
; TO THE TERMINAL BUFFER AND THEN TO OUTPUT THE BUFFER TO THE
; TERMINAL IF SPECIFIED.  IF THE TERMINAL BUFFER IS OUTPUT, A CR
; AND LF ARE APPENDED TO THE END OF THE BUFFER.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   SMES,<N,CHRS>
;
; WHERE:
;
;       N       SPECIFIES THE NUMBER OF CHARACTER TO OUTPUT.  IF N
;               IS NEGATIVE, THE ABSOLUTE VALUE OF N SPECIFIES THE
;               NUMBER OF CHARACTERS.  N ALSO SPECIFIES WHETHER THE
;               TERMINAL BUFFER IS TO BE OUTPUT TO THE TERMINAL.
;               N<0:  DO NOT OUTPUT TERMINAL BUFFER.
;               N= OR >:  OUTPUT TERMINAL BUFFER.
;
;       CHRS    IS THE ADDRESS OF THE FIRST CHARACTER OF THE CHAR-
;               ACTER STRING TO OUTPUT.
;
;
; SUBROUTINE GETS:
; THIS ROUTINE IS CALLED TO INPUT A STRING OF CHARACTERS FROM THE
```

```
; CONSOLE TERMINAL.  THE STRING INPUT BY THE OPERATOR IN THE ARRAY
; SPECIFIED.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   GETS,<N,BUFF>
;
; WHERE:
;
;       N       SPECIFIES THE NUMBER OF CHARACTERS TO BE INPUT (IF
;               N IS NEGATIVE, NO NEW LINE IS READ IN, BUT INSTEAD
;               THE REMAINING CHARACTERS IN THE OLD LINE ARE USED).
;               IF FEWER CHARACTERS THAN THE ABSOLUTE VALUE OF N ARE
;               INPUT (UP TO THE CARRIAGE RETURN, BUT NOT INCLUDING)
;               THEN THE BUFFER IS FILLED TO THE END (END=ABS(N)) WITH
;               NULLS (0).
;
;       BUFF    SPECIFIES THE ADDRESS OF THE BUFFER THAT THE STRING
;               IS TO BE RETURNED IN.  NOTE THAT THE STRING WILL
;               HAVE NO RUBOUTS, CR OR LF.
;
;
; SUBROUTINE GETN:
; THIS ROUTINE IS CALLED TO INPUT ASTRING OF CHARACTERS FROM THE
; CONSOLE TERMINAL AND TO CONVERT THE CHARACTERS TO A BINARY NUMBER
; WHICH IS RETURNED TO THE CALLING ROUTINE.  A MAXIMUM VALUE IS SPEC-
; IFIED WHICH IS USED TO CHECK THE NUMBER INPUT.  IF THE ABSOLUTE VALUE
; OF THE NUMBER INPUT IS GREATER THAN THE MAXIMUM SPECIFIED, THEN THE
; NUMBER IS REQUESTED TO BE RE-INPUT BY THE OPERATOR.
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   GETN,<NMAX,N>
; WHERE:
;
;       NMAX    SPECIFIES THE MAXIMUM THAT THE NUMBER INPUT CAN BE.
;               IF NMAX<0 OR NMAX=0, NO VALUE CHECKING OCCURS.  ALSO,
;               IF NMAX<0, THE REMAINING CONTENTS OF THE PREVIOUS
;               LINE ARE USED, WHEREAS FOR NMAX=0 OR GREATER, A NEW
;               LINE IS INPUT.  NOTE THAT FOR NMAX>0, ONLY A
;               SINGLE NUMBER, WITH NO DELIMITERS, IS ALLOWED ON THE
;               LINE.
;
;       N       IS THE VARIABLE IN WHICH THE NUMBER INPUT IS RE-
;               TURNED.  IT SHOULD BE NOTED THAT ONCE GETN IS CALLED
;               WHERE NMAX>0, THE ROUTINE IS NOT RETURNED FROM UNTIL
;               A VALID NUMBER HAS BEEN INPUT.
;
```

B.4. QSDDT1.MAC; STRING PARSING AND BYTE PACKING

```
;
;
; GETC SCANS AN INPUT STRING TO THE FIRST CHARACTER
; OTHER THAN THE FIRST THROUGH SIXTH DIGITS AND RETURNS THE
; FOLLOWING:
;
;               NDIG:   NEGATIVE NUMBER OF DIGITS IN N
;               NUMB:   N IF NON-NULL, OTHERWISE 0
;               DLIM:   DELIMITER OR 7TH CHAR
;
; GETC IS CALLED IN MIXIT AS FOLLOWS:
;
;       CALL    GETC,<STRN,NDIG,NUMB,DLIM,RADX>
;
;                       STRN IS A PTR TO THE STRING TO BE PARSED,
;                       AND WILL BE MODIFIED BY GETC TO POINT JUST
;                       PAST THE DELIMITER.
;                       RADX = RADIX 8. OR 10.
;
;                       INPUT VALUES MAY BE NEGATIVE, BEGINNING WITH -,
;                       OR SET-BIT EXTENDED, BEGINNING WITH '
;                       E.G. -1 = 177777, AND '53=177753
;
;       GETC FACILITATES PARSING OF AN INPUT COMMAND STRING BY
;       RETURNING THE NEXT OCTAL OR DECIMAL VALUE, IF ANY, AND
;       THE DELIMITER FOLLOWING.  THE STRING TO BE PARSED WOULD
;       NORMALLY BE OBTAINED BY A PRIOR CALL TO SUBROUTINE 'GETS'
;       IN IOSUBS
;
;
; SUBROUTINE BYWD COMBINES TWO 8-BIT VALUES ('BYTES')
; INTO A 16-BIT VALUE ACCORDING TO THE PDP-11 BYTE SEQUENCE.
; MIXIT CALLING SEQUENCE:
;
;       CALL    BYWD,<WRD1,WRD2,WRD3>
;
;                       WRD1 AND WRD2 ARE THE
;                       1ST AND 2ND 8-BIT VALUES
;                       (IN THE SEQUENCE THEY ARE TO BE
;                       TRANSMITTED TO THE CG)
;                       WRD3 IS THE 16-BIT TARGET VALUE
;
```

```
        B.5. RTI.MAC; REMOTE TERMINAL INTERFACE I/O

;       SUBROUTINE ROT:
;
;
;THIS SUBROUTINE TAKES LENGTH CHARACTERS FROM THE HOST'S SECONDARY
 SERIAL
;INTERFACE AND PUTS THEM IN THE CHARACTER ARRAY STRING
;
;
;       MIXIT CALLING SEQUENCE:
;
;       .CALL   ROT,<STRING,LENGTH>
;
;       WHERE:
;
;               STRING = ADDR OF STRING INPUT BUFFER
;
;               LENGTH = STRING LENGTH
;
;SUBROUTINE ROTC:
;
;THIS SUBROUTINE CLEARS THE HOST'S SECONDARY SERIAL INTERFACE OF ANY
;CHARACTERS THAT MAY ALREADY BE PRESENT
;
;       MIXIT CALLING SEQUENCE:
;
;       .CALL   ROTC
;
;SUBROUTINE WOT:
;
;THIS SUBROUTINE TAKES LENGTH CHARACTERS FROM CHARACTER ARRAY STRING
;AND SENDS THEM TO THE HOST'S SECONDARY SERIAL INTERFACE
;
;       MIXIT CALLING SEQUENCE:
;
;       .CALL   WOT,<STRING,LENGTH>
;
;       WHERE:
;
;               STRING = OUTPUT STRING BUFFER
;
;               LENGTH = STRING LENGTH
```

APPENDIX C

The following table contains miscellaneous information about operating systems which support Picture System Diagnostics

PICTURE SYSTEM DIAGNOSTIC OPERATING SYSTEMS

15-Aug.-80

OPERATING SYSTEMS AND CPU	MACHINE DEPENDENT FILES	MIXIT COMPLIER & TARGET FILES	LINKER OPTIONS	BATCH CAPABILITY	QSD11Ø IMPLEMENTATION	1st BYTE	COMMENTS RESTRICTIONS
RT-11 PDP-11	PSIO.MAC IOSUBS.MAC QSDDT1.MAC RTI.MAC	MIX11 .MAC		RT-11 BATCH .BAT	RTI.MAC modify H3-H5	R	Single-user Fast Floppy Diskette Test Station- Batch
MAPPED RSX-11M PDP-11 with MEMORY MANAGEMENT	PSIO2.MAC IOSUB2.MAC QSDDT1.MAC RTIQIO.MAC	MIX11 .MAC	ASG=TI:1 COMMON= PSDEVØ:RW	IOSUB3 .CMD (limited) ZAP .CMD (limited)	RTIQIO.MAC >ASN TTn:=RT:	R	No DMA or interrupts >SET /MAIN=PSDEVØ: 7676:2:DEV >INS PSDEVØ >ALL MP:
MAPPED RSX-11M PDP-11	PSIO.MAC IOSUB2.MAC QSDDT1.MAC RTI.MAC	MIX11 .MAC	ASG=TI:1 PAR=PAR14K: 40000:70000	NO	same as RT-11	R	Can be on same disk with Mapped RSX. Allows DMA and interrupts single user
VMS VAX-11/780	PSIOV.MAC IOSUB2.MAC QSDDT1.MAC RTIQIO.MAC	MIX11 .MAC	ASG=TI:1 ASG=MPAØ:4	.COM	same as MAPPED RSX	R	MPDRV must be loaded $ ASSIGN MPBØ: MPAØ Multi-user protection cannot set I4 bit Ø
OS32 INTERDATA 8-32	PSIO.CAL IOSUBS.CAL QSDDT1.CAL	MIX832 .CAL QSI000,001, 100.MIX QSD11701.MIX	see .CSS FILES	.CSS?	NONE	L	Edit QSDDT for byte sequence and disable "y" cmd. Special MT:format
SINTRAN NORD-1Ø	OEM-C.I.I.R. OSLO	MIXNOR .NOR QSD117Ø1.MIX			NONE	L	Edit QSDDT FLX MT:/DO Hard copy MT:Directory Supply EMBL
MELCOM-7Ø	OEM-RIKEI JAPAN	MIXMEL. .MEL NEWMEL. .MER QSD117Ø1.MIX			NONE	L R	Edit QSDDT FLX MT:/DO

APPENDIX D
MIXIT (MACHINE INDEPENDENT) UTILITY SUBROUTINES

D.1. INTRODUCTION

The Followins Subroutines are Described in this Appendix:

INIT.MIX (Sec. D.2)

INIT	STANDARD COMMAND INTERPRETER
DPCH	PHASE DISPATCHER CONTROLLED BY INIT
ERRL	ERROR LOOPING CONTROL

ARF2.MIX OR ARFS.MIX AUTOREFRESH CONTROL,
SINGLE- OR MULTI-USER RFC (USE ARF2 FOR DOUBLE BUFFERING)
(Sec. D.3)

RTWT	REAL TIME WAIT
ARFS	AUTOREFRESH CONTROL

RNDM.MIX (Sec. D.4.)

RNDM	RANDOM NUMBER GENERATOR

D.2. INIT.MIX, Standard Operator Interface

```
;
; SUBROUTINE INIT:
;
; THIS SUBROUTINE INTERPRETS THE STANDARD OPERATING COMMANDS
; DESCRIBED IN THE PS2 DIAGNOSTICS MANUAL:
;
;               H HELP
;               P NUMBER OF PASSES
;               D PHASE SELECTION ('DO')
;               L LOOP ON ERROR
;               C LOOP ON ERROR AND CONTINUE
;               M MODIFY TABLES
;               X EXECUTE (EXIT INIT)
;               S STOP ON ERROR
;
; MIXIT CALLING SEQUENCE:
;
;       MOV     (number of phases),DOPH
;       MOV     (max error identification),PHAZ
;       CALL    INIT,<MSGS>
;               where MSGS is a table as follows:
;               MSGS:   DATA    (number of salutation messages)
;                       DATA    <(mss ptr 1),(char cnt 1)>
;                       .
;                       DATA    (number of help messages)
;                       DATA    <(mss ptr n+1),(char cnt n+1)>
;                       .
;
;
; SUBROUTINE DPCH:
;
; THIS SUBROUTINE CALLS PHASES OF A DIAGNOSTIC IN ACCORDANCE WITH
; D AND P COMMANDS PREVIOUSLY INTERPRETED BY INIT
;
; MIXIT CALLING SEQUENCE:
;
;       CALL    DPCH,<PTBL>
;               where PTBL is a table of pointers to
;               phases of a diagnostic as follows:
;               PTBL:   DATA    <0,PH1,PH2,....>
;                       .
;               and each phase is a subroutine declared
;               as follows:
;               SUBR    PH1,1   ;.1=PHASE NUMBER
;
;
```

```
; SUBROUTINE ERRL:
;
; THIS SUBROUTINE CONTROLS ERROR LOOPING IN ACCORDANCE WITH
; L, C, AND S COMMANDS PREVIOUSLY INTERPRETED BY INIT
;
; MIXIT CALLING SEQUENCE:
;
;       CALL    ERRL,<ERRN,ERSB>
;               where ERRN is the error identification number
;               and ERSB is an error subroutine as follows:
;
;               SUBR    ERSB,1  ;.1=RETURN ERROR STATUS
;               .
;               (repeat test which has previously failed,
;               but do not output an error message)
;               .
;               CMPL    (expected value),(received value)
;               MOV     TTTT,.1         ;RETURN TEST RESULT
;               RTRN
;
;
```

D.3. ARFS.MIX,ARF2.MIX, Autorefresh Control

```
;
; GLOBAL ROUTINES RTWT (REAL TIME WAIT)
; AND ARFS (AUTOREFRESH) CAN ACCOMODATE
; TWO VERSIONS OF REFRESH CONTROLLER,
; SINGLE USER (TYPE 1), OR MULT-USER (TYPE 2)
; LOCAL ROUTINE RFIN IS CALLED BY BOTH ARFS AND RTWT
; TO DETERMINE THE TYPE OF REFRESH CONTROLLER
; AVAILABLE, AND TO INITIALIZE IT.
;
;
;
; SUBROUTINE RTWT, WAIT FOR REAL TIME CLOCK REQUEST
; CALLING SEQUENCE:
;
;       CALL    RTWT
;
;
; SUBROUTINE ARFS CONTROLS AUTOREFRESH
; CALLING SEQUENCE:
;
;       CALL ARFS,<MINA,MAXA,RATE,FUNC>
;
; WHERE:
;
;       .1 (MINA) = START ADDRESS OF REF BUF
;               .2 (MAXA) = REFRESH LIMIT
;               .3 (RATE) = NUMBER OF 120./SEC PER CLK REQ
;                           TWO'S COMP 1-17 (IF 0, NO CHANGE)
;               .4 (FUNC) = 1 TO STOP A.R., 2 TO START A.R., 0 TO
;                           LOAD CLK RATE ONLY, 4 TO RE-INIT
;                           (4 IS VALID ONLY AFTER PSRESET)
;
; NOTE: RSPS MUST BE CALLED PRIOR TO THE FIRST CALL TO ARFS
;
;
```

Refresh table starts at 37600 and is filled with no-ops 1st test command address is 37604, next is 37605 etc.

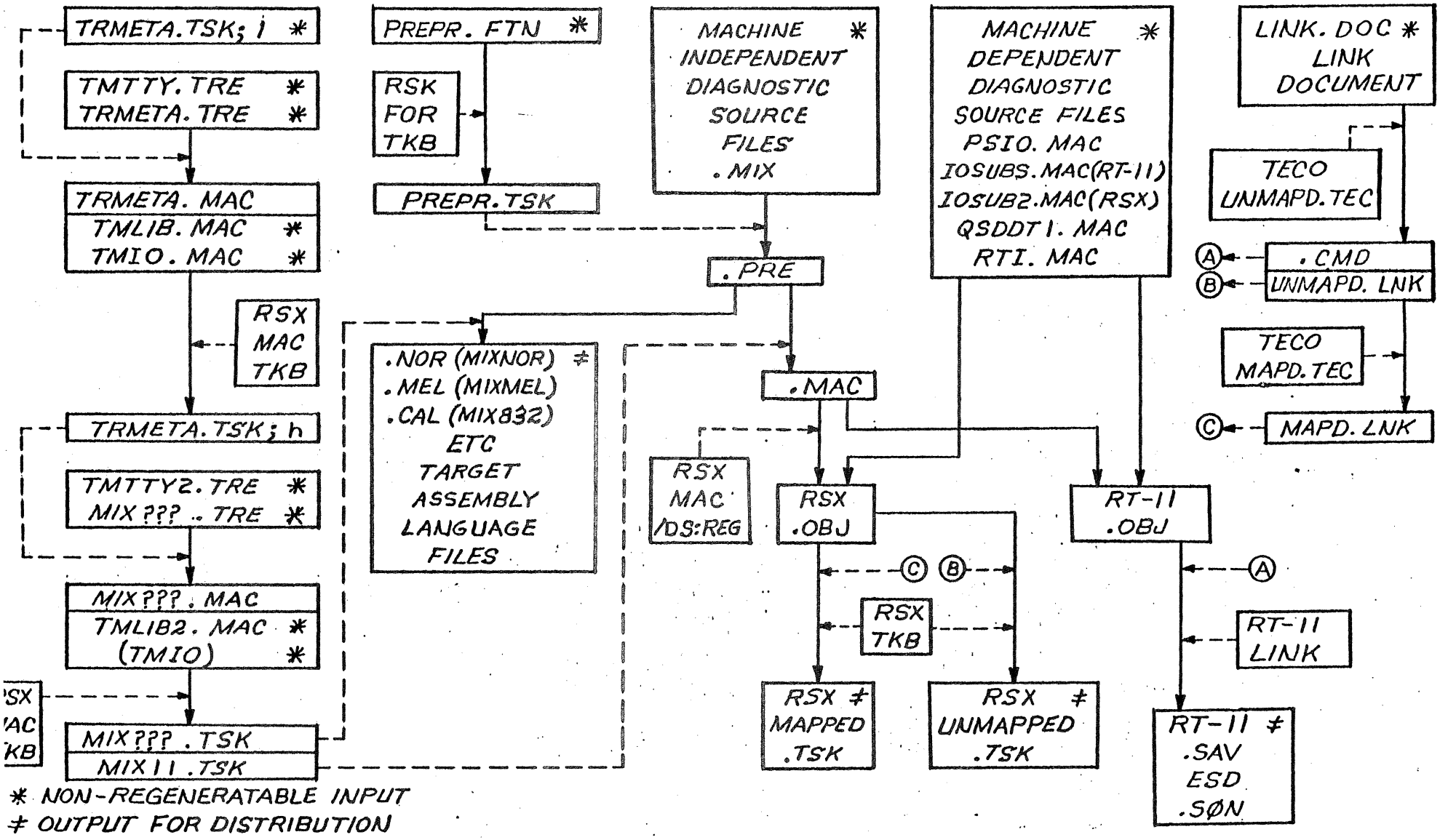
[illegible] RSD11 starts at 37000

D.4. RNDM.MIX, Random Number Generator

```
;       SUBROUTINE RNDM: RANDOM NUMBER GENERATOR
;
; MIXIT CALLING SEQUENCE:
;
;       .CALL   RNDM,<NUM,KEY>
;
; WHERE:
;
;       NUM = RANDOM VALUE RETURNED
;
;       KEY = RANDOM NUMBER KEY  Seed for number generation.
;
```

APPENDIX E

The following flowchart represents required files and procedures for generation of the Tree-Meta Compiler TRMETA, various MIXIT compilers, Diagnostic Source Files, and loadable RT-11 .SAV files and RSX-11M .TSK files.



PS2 DIAGNOSTICS GENERATION

2 OCT 1978

> ALL MPØ

> DEA MPØ

APPENDIX F

LINK.DOC contains linking instructions in RT-11 format for all distributable diagnostics. It is used as the master reference for all operating systems.

```
;FILE LINK.DOC REVISED 26-MAY-80
;PS2 DIAGNOSTICS VERSION 0313

*QSD000.S01=INIT,QSD000,PSIO,IOSUBS,RNDM

*QSD001.S01=INIT,QSD001,PSIO,IOSUBS,RNDM

*QSD002.S02=INIT,QSD002,PSIO,IOSUBS,RNDM,FTIME

*QSD003.S01=INIT,QSD003,PSIO,IOSUBS,CODE,RNDM

*QSD004.S01=INIT,QSD004,PSIO,IOSUBS,CODE,RNDM/C
*CROM1

*QSD004.S02=INIT,QSD004,PSIO,IOSUBS,CODE,RNDM/C
*CROM2

*QSD005.S01=INIT,QSD005,PSIO,IOSUBS,CODE,RNDM

*QSD006.S01=INIT,QSD006,PSIO,IOSUBS,CODE,RNDM

*QSD007.S01=INIT,QSD007,PSIO,IOSUBS,CODE,RNDM

*QSD008.S01=INIT,QSD008,PSIO,IOSUBS,CODE,RNDM

*QSD009.S01=INIT,QSD009,PSIO,IOSUBS,CODE,RNDM

*QSD010.S01=INIT,QSD010,PSIO,IOSUBS,CODE,RNDM

*QSD011.S01=INIT,QSD011,PSIO,IOSUBS,CODE,RNDM

*QSD012.S01=INIT,QSD012,PSIO,IOSUBS,CODE,RNDM

*QSD013.S01=INIT,QSD013,PSIO,IOSUBS,CODE,RNDM

*QSD014.S01=INIT,QSD014,PSIO,IOSUBS,CODE,RNDM/C
*RROM

*QSD015.S01=INIT,QSD015,PSIO,IOSUBS,CODE,RNDM/C
*MLSM

*QSD016.S01=INIT,QSD016,PSIO,IOSUBS,CODE,RNDM

*QSD017.S02=INIT,QSD017,IOSUBS,PSIO,RNDM
```

```
*QSD018.S04=INIT,QSD018,PSIO,IOSUBS,RNDM/C
*ARFS,INCM,NNEW

*QSD020.S06=INIT,QSD020,DAT020,IOSUBS,PSIO/C
*ARFS

*QSD021.S01=INIT,QSD021,IOSUBS,PSIO,ARFS

*QSD026.S04=INIT,QSD026,PSIO,IOSUBS,ARFS/C
*INCM,NNEW

*QSD027.S04=INIT,QSD027,DAT027,PSIO,IOSUBS,ARFS/C
*INCM,NNEW

*QSD028.S05=INIT,QSD028,Q02801,DD28,CHRAM/C
*B200,INCM,LOBF,ARFS/C
*PSIO,IOSUBS

*QSD031.S01=INIT,QSD031,PSIO,IOSUBS,NCODE,RNDM

*QSD033.S02=INIT,QSD033,PSIO,IOSUBS

*QSD034.S01=INIT,QSD034,PSIO,IOSUBS

*QSD035.S01=INIT,QSD035,PSIO,IOSUBS

*QSD036.S01=INIT,QSD036,DD036,PSIO,IOSUBS

*QSD037.S01=INIT,QSD037,DD037,PSIO,IOSUBS

*QSD040.S01=INIT,QSD040,DD40,PSIO,IOSUBS/C
*ARFS,LOBF

*QSD100.S02=INIT,QSD100,PSIO,IOSUBS,RNDM

*QSD102.S02=INIT,QSD102,PSIO,IOSUBS

*QSD103.S01=INIT,QSD103,PSIO,IOSUBS,RNDM

*QSD104.S02=INIT,QSD104,PSIO,IOSUBS,QSDDT1

*QSD105.S01=INIT,QSD105,QSE105,IOSUBS,PSIO

*QSD107.S01=INIT,QSD107,IOSUBS,PSIO/C
*TANSUB,TANLIB,RNDM

*QSD108.S07=INIT,QSD108,IOSUBS,PSIO,TANSUB/C
*TANLIB,RNDM,ARF2

*QSD109.S01=INIT,QSD109,QSE109,CROM3/C
*PSIO,IOSUBS
```

```
*QSD110.S04=INIT,QSD110,IOSUBS,PSIO,TANSUB/C
*TANLIB,RNDM,RTI

*QSD111.S02=INIT,QSD111,IOSUBS,PSIO/C
*RNDM

*QSD112.S02=INIT,QSD112,IOSUBS,PSIO,TANSUB/C
*TANLIB,RNDM,RFLIB,RFCODE

*QSD113.S03=INIT,QSD113,IOSUBS,PSIO/C
*RNDM,TANSUB,TANLIB,RFLIB

*QSD114.S01=INIT,QSD114,IOSUBS,PSIO/C
*RNDM,TANLIB,TANSUB,RFLIB

*QSD115.S01=INIT,QSD115,IOSUBS,PSIO/C
*RNDM,TANSUB,TANLIB,RFLIB

*QSD116.S02=INIT,QSD116,IOSUBS,PSIO/C
*TANSUB,TANLIB,RFLIB,RNDM

*QSD117.S02=INIT,QSD117,IOSUBS,PSIO,RFLIB

*QSD120.S03=INIT,QSD120,PSIO,IOSUBS,RNDM

*QSD130.S01=INIT,QSD130,QSE109,CROM6,PSIO,IOSUBS

*QSD131.S01=INIT,QSD130,QSE109,CROM7,PSIO,IOSUBS

*QSD136.S01=INIT,QSD136,PSIO,IOSUBS,CODE,RNDM

*RSD000.S03=INIT,RSD000,DD00,R000/C
*INCM,LOBF,ARFS/C
*PSIO,IOSUBS

*RSD001.S02=INIT,RSD001,R00101,DD01,INCM/C
*B360,LOBF,ARFS,PSIO/C
*IOSUBS

*RSD002.S03=INIT,RSD002,R002/C
*INCM,LOBF,ARFS,PSIO,IOSUBS

*RSD003.S02=INIT,RSD003,R00301,R00302,R00303/C
*DD03,B200,DOBF,WTSG/C
*INCM,LOBF,RTWT,PSIO,IOSUBS

*RSD004.S02=INIT,RSD004,R00401,R00402,DD04/C
*DOBF,INCM,B360,LOBF,ARFS/C
*DMR2,PSIO,IOSUBS
```

```
*RSD005.S03=INIT,RSD005,R00501,DD05,INCM/C
*B200,LOBF,ARFS,PSIO/C
*IOSUBS

*RSD006.S04=INIT,RSD006,R00602,DD06,DOBF/C
*INCM,B360,LOBF,ARFS/C
*PSIO,IOSUBS

*RSD007.S04=INIT,RSD007,R00701,DD07,B440/C
*INCM,LOBF,ARFS,PSIO/C
*IOSUBS

*RSD008.S04=INIT,RSD008,R00701,DD08,B200/C
*INCM,LOBF,ARFS/C
*PSIO,IOSUBS

*RSD009.S05=INIT,RSD009,R00901,DD09,B200/C
*CHROM,INCM,LOBF,ARFS/C
*DOBF,PSIO,IOSUBS

*RSD010.S01=INIT,RSD010,R10S1,INCM,ARF2/C
*PSIO,IOSUBS

*RSD011.S01=INIT,RSD011,R11S1,INCM,ARF2/C
*PSIO,IOSUBS

*RSD012.S01=INIT,RSD012,R12S1,ARFS/C
*PSIO,IOSUBS

*RSD013.S02=INIT,RSD013,DD13,ARFS,LOBF,RNDM/C
*PSIO,IOSUBS

*RSD014.S02=INIT,RSD014,DD14,ARFS,LOBF/C
*PSIO,IOSUBS

*QSDDT.S05=INIT,QSDDT,QSDDT2,QSDDT1,ARFS/C
*INCM,PSIO,IOSUBS

$EOJ
```

APPENDIX G
SAMPLE MIXIT PROGRAM - QSD002.MIX

```
        JMP     STRT        ;FOR MANUAL STARTUPS
;
;
; PROGRAM:  QSD002.MIX
;
; AUTHOR:  STEPHEN N. MCALLISTER
;
; DATE WRITTEN:  5/14/76
;
;       VER S02 TIMEOUT AND MESSAGE CHANGES 24-NOV-79
;
; DESCRIPTION:  THIS PROGRAM PROVIDES THE PICTURE SYSTEM MEMORY
;       TESTS.  THERE ARE SEVEN TESTS, INCLUDING DATA PATH, ADDRESS/
;       DATA, AND MEMORY CONTENT CHECKS.
;
;
        HEAD    <MESSAGE SECTION>
MSGS:   DATA    2
        DATA    <MSG1,10.>
        DATA    <MSG2,33.>
        DATA    12.
        DATA    <MS10,44.>
        DATA    <MS11,34.>
        DATA    <MS12,26.>
        DATA    <MS13,29.>
        DATA    <MS14,28.>
        DATA    <MS15,29.>
        DATA    <MS16,12.>
        DATA    <MS17,17.>
        DATA    <MS18,11.>
        DATA    <MS19,20.>
        DATA    <MS20,22.>
        DATA    <MS21,52.>
MSG1:   CDATA   <QSD002.S02>
MSG2:   CDATA   <PICTURE SYSTEM MEMORY DIAGNOSTICS>
MS10:   CDATA   <THIS DIAGNOSTIC TESTS PICTURE SYSTEM MEMORY.>
MS11:   CDATA   <THERE ARE SEVEN TESTS, AS FOLLOWS:>
MS12:   CDATA   <1.  MEMORY DATA PATH CHECK>
MS13:   CDATA   <2.  MEMORY ADDRESS/DATA CHECK>
MS14:   CDATA   <3-7.  MEMORY CONTENTS CHECKS>
MS15:   CDATA   <THE FIVE CONTENTS CHECKS ARE:>
MS16:   CDATA   <3.  ZERO/ONE>
MS17:   CDATA   <4.  RANDOM NUMBER>
MS18:   CDATA   <5.  REFRESH>
MS19:   CDATA   <6.  BIT DISTURB ONES>
MS20:   CDATA   <7.  BIT DISTURB ZEROES>
MS21:   CDATA   <P100 = REFRESH TEST DELAY, DEFLT 740 FOR 60 SEC.>
MSGA:   DATA    -22.
        CDATA   <1: DATA PATH ERR;PORT=>
```

```
MSGB:   DATA    -6.
        CDATA   < ADDR=>
MSGC:   DATA    -11.
        CDATA   < DATA SENT=>
MSGD:   DATA    -11.
        CDATA   < DATA RECD=>
MSGE:   DATA    -20.
        CDATA   <2: ADDRESS ERR;PORT=>
MSGF:   DATA    -16.
        CDATA   <3: ZERO/ONE ERR;>
MSGG:   DATA    -18.
        CDATA   <: RANDOM DATA ERR;>
MSGH:   DATA    -18.
        CDATA   <: BIT DISTURB ERR;>
MSGJ:   DATA    39.
        CDATA   <GROUND 195141-100 PIN 62 -- CARR. RTRN.>
MSGM:   DATA    46.
        CDATA   <ENTER MEMORY SIZE (1=16K, 2=32K, 3=48K, 4=64K)>
MSGU:   DATA    28.
        CDATA   <REMOVE JUMPER -- CARR. RTRN.>
MSGR:   DATA    7
        CDATA   <RUNNING>
CTLG:   DATA    -1
        BDATA   <7,0>
MS99:   DATA    21.
        CDATA   <MEMORY TESTS COMPLETE>
        HEAD    <CONSTANTS AND TEMPORARY STORAGE>
X0:     DATA    0
X1:     DATA    1
X2:     DATA    2
X3:     DATA    3
X4:     DATA    4
X5:     DATA    5
X6:     DATA    6
X7:     DATA    7
RFDL:   DATA    480.  ;WAIT PARAM = 60 SEC. (OCTAL 740)
X100:   DATA    100
X200:   DATA    200
X400:   DATA    400
X4K:    DATA    10000
X12K:   DATA    30000
X16K:   DATA    40000
XHI:    DATA    177377              ;HIGHEST POSSIBLE MEMORY LOCATION
COMP:   BLOCK   202.
CDIF:   DIFF    COMP,CDIF
MSK1:   DATA    77
MSK2:   DATA    7700
MSK3:   DATA    170000
N:      BLOCK   1
I:      BLOCK   1
I2:     BLOCK   1
M:      BLOCK   1
M2:     BLOCK   1
```

```
PRTB:   BLOCK   1
TEMP:   BLOCK   1
TMP2:   BLOCK   1
ADDR:   BLOCK   1
MSIZ:   DATA    0
JTBL:   DATA    <0,PH1,PH2,PH3,PH4,PH4,PH6,PH6>
DATA:   DATA    <0,177777,125252,52525,123456>
        HEAD    <DISPATCHER>
        THERE   <INIT,SMES,SOCT,GETS,GETN,WRPS,RDPS,WAIT,RNDM>
        THERE   <DPCH,ERRL,PHAZ,DOPH,PSTB>
STRT:   MOV     X7,PHAZ
        MOV     X7,DOPH
        CALL    INIT,<MSGS> ;INITIALIZE
ST1:    TST     MSIZ        ;GET MEMORY SIZE?
        BNZ     ST3         ;ALREADY GOT
        CALL    SMES,<MSGM,<1,MSGM>>    ;GET MEMORY SIZE
        CALL    GETN,<X4,TTTT>
        BRZ     ST1         ;MAKE SURE IT'S LEGAL
        BRN     ST1
        CLR     MSIZ
        DEC     MSIZ
ST2:    ADD     X16K,MSIZ
        DEC     TTTT
        BNZ     ST2
        CMPL    MSIZ,XHI    ;TOO HIGH?
        BRN     ST3         ;NO
        MOV     XHI,MSIZ    ;YES, FIX IT UP
ST3:    TST     <100,PSTB>  ;USER SPECIFIED REFR DELAY?
        BRZ     ST4         ;NO
        MOV     <100,PSTB>,RFDL   ;YES, GET IT
ST4:    CALL    DPCH,<JTBL> ;CALL THE DISPATCHER
        CALL    SMES,<MS99,<1,MS99>>    ;SAY DONE
        STOP                ;QUIT...
        HEAD    <ERROR PROCEDURE PROCESSOR>
        SUBR    ERDO,1              ;ENTRY POINT
        CALL    SMES,<MSGB,<1,MSGB>>    ;FINISH THE MSG.
        CALL    SOCT,<MSGB,ADDR>
        CALL    SMES,<MSGC,<1,MSGC>>
        CALL    SOCT,<MSGC,TMP2>
        CALL    SMES,<MSGD,<1,MSGD>>
        CALL    SOCT,<X1,TEMP>
        CALL    ERRL,<.1,ERPT>    ;CALL ERROR LOOP PROCESSOR
        RTRN                ;RETURN TO ERROR PLACE
        HEAD    <ERPT -- RECREATE ERRORS>
        SUBR    ERPT,1
        CALL    WRPS,<ADDR,X1,TMP2,X1>  ;REPEAT THE TEST
        CALL    RDPS,<ADDR,X1,TEMP,X1>
        CMPL    TEMP,TMP2   ;MAKE COMPARISON
        MOV     TTTT,.1             ;RETURN WITH RESULT
        RTRN
        HEAD    <PHASE 1 -- MEMORY DATA PATH CHECK>
        SUBR    PH1,1       ;ENTRY POINT
        CLR     PRTB        ;INITIALIZE
```

```
        CLR     N
P1A:    CLR     I               ;VALUE TEST
P1B:    CLR     M               ;4K MEMORY BOUNDARY
P1C:    MOV     M,ADDR                  ;ADDRESS = M + N
        ADD     N,ADDR
        CALL    WRPS,<ADDR,X1,<I,DATA>,X1> ;WRITE
        CALL    RDPS,<ADDR,X1,TEMP,X1>  ;READ
        CMPL    TEMP,<I,DATA>           ;ERROR?
        BNZ     P1M             ;YES
P1D:    CMPA    I,X4            ;NO, I = 4?
        BRZ     P1E             ;YES
        INC     I               ;NO, I = I + 1
        JMP     P1B             ;LOOP ON M
P1E:    ADD     X16K,N                  ;N = N + 16K
        TST     N               ;WRAP-AROUND?
        BRZ     P1F             ;YES
        CLR     M               ;NO, M = 0
        CMPL    MSIZ,N                  ;IS THERE N MEMORY?
        BRP     P1A             ;YES
P1F:    TST     PRTB            ;PRTB SET YET?
        BNZ     P1L             ;YES
        CALL    SMES,<CTLG,<1,CTLG>>    ;NO, RING BELL AND
        CALL    SMES,<MSGJ,<1,MSGJ>>    ;ASK FOR JUMPER
        CALL    GETS,<X1,TEMP>  ;WAIT FOR DONE
        CLR     N               ;N = 0
        INC     PRTB            ;SET PRTB
        JMP     P1A
P1L:    CALL    SMES,<MSGU,<1,MSGU>>    ;REMOVE JUMPER
        CALL    GETS,<X1,TEMP>
        RTRN
P1M:    CMPL    M,X12K                  ;M .GE. 12K?
        BRP     P1N             ;YES
        ADD     X4K,M           ;NO, M = M + 4K
        JMP     P1C
P1N:    CALL    SMES,<MSGA,<1,MSGA>>    ;OUTPUT ERROR MSG.
        CALL    SOCT,<MSGA,PRTB>
        MOV     <I,DATA>,TMP2
        CALL    ERDO,X1                 ;GO DO ERROR TEST
        JMP     P1D
        HEAD    <PHASE 2 -- MEMORY ADDRESS/DATA CHECK>
        SUBR    PH2,1           ;ENTRY POINT
        CLR     PRTB            ;INITIALIZE
        CLR     ADDR
P2A:    CALL    WRPS,<ADDR,X1,ADDR,X1>  ;WRITE ONE OUT
        CMPL    ADDR,MSIZ       ;LAST ADDRESS?
        BRP     P2B
        INC     ADDR
        JMP     P2A             ;BUMP N AND LOOP
P2B:    CLR     ADDR            ;PREPARE TO READ BACK
P2C:    CALL    RDPS,<ADDR,X1,TEMP,X1>  ;READ BACK
        CMPL    ADDR,TEMP       ;RESULTS AGREE?
        BNZ     P2M             ;NO
P2D:    CMPL    ADDR,MSIZ       ;YES, LAST ADDRESS?
```

```
          BRP     P2E
          INC     ADDR          ;NO, BUMP N
          JMP     P2C           ;AND LOOP
P2E:      TST     PRTB          ;PRTB SET YET?
          BNZ     P2L           ;YES
          CALL    SMES,<CTLG,<1,CTLG>>    ;NO, RING BELL AND
          CALL    SMES,<MSGJ,<1,MSGJ>>    ;ASK FOR JUMPER
          CALL    GETS,<X1,TEMP>    ;WAIT FOR ANSWER
          CALL    SMES,<MSGR,<1,MSGR>>    ;SAY 'RUNNING'
          CLR     ADDR          ;CLEAR N
          INC     PRTB          ;SET PRTB
          JMP     P2A
P2L:      CALL    SMES,<CTLG,<1,CTLG>>    ;RING BELL
          CALL    SMES,<MSGU,<1,MSGU>>    ;REMOVE JUMPER
          CALL    GETS,<X1,TEMP>
          RTRN                  ;RETURN
P2M:      CALL    SMES,<MSGE,<1,MSGE>>    ;OUTPUT ERROR MSG.
          CALL    SOCT,<MSGE,PRTB>
          MOV     ADDR,TMP2
          CALL    ERDO,X2             ;GO DO ERR TEST
          JMP     P2D
          HEAD    <PHASE 3 -- ALTERNATING ZERO/ONE TEST>
          SUBR    PH3,1         ;ENTRY POINT
          CLR     TMP2          ;INITIALIZE
          COM     TMP2
          CALL    P3A,<X0,X0,X0>    ;LOAD WITH ONES
          CALL    P3A,<X0,X1,X0>    ;CHECK IT
          CLR     TMP2
          CALL    P3A,<X2,X0,X0>    ;COMPLIMENT EVEN LOCS.
          CALL    P3A,<X0,X1,X1>    ;CHECK FOR 0,1,0, ETC.
          CALL    P3A,<X1,X0,X0>    ;COMPLIMENT ODD LOCS.
          CALL    P3A,<X0,X1,X0>    ;CHECK FOR ZEROES
          CLR     TMP2
          COM     TMP2
          CALL    P3A,<X2,X0,X0>    ;COMPLIMENT EVEN LOCS.
          CALL    P3A,<X0,X1,X1>    ;CHECK FOR 1,0,1, ETC.
          RTRN                  ;RETURN
;
;
;         EXECUTIVE SUBROUTINE
;
;         .1:     0 = ALL LOCATIONS
;                 1 = ODD     '
;                 2 = EVEN    '
;
;         .2:     0 = WRITE C(TMP2)
;                 1 = READ & COMPARE WITH C(TMP2)
;
;         .3:     0 = DO NOT MODIFY C(TMP2)
;                 1 = COMPLIMENT C(TMP2) AFTER ACCESS
;
;
          SUBR    P3A,3         ;SUBROUTINE TO DO IT
```

```
        CLR     ADDR            ;START AT THE START
P3B:    MOV     .1,TTTT                 ;ALL LOCS?
        BRZ     P3C             ;YES
        ADD     ADDR,TTTT       ;NO, MASK ALL BUT UNITS BIT
        AND     X1,TTTT                 ;IS IT US?
        BNZ     P3X             ;NO
P3C:    TST     .2              ;YES, WRITE?
        BNZ     P3E             ;NO
P3D:    CALL    WRPS,<ADDR,X1,TMP2,X1>  ;WRITE
        JMP     P3G
P3E:    CALL    RDPS,<ADDR,X1,TEMP,X1>  ;READ
        CMPL    TEMP,TMP2       ;ERROR?
        BRZ     P3G             ;NO
        CALL    SMES,<MSGF,<1,MSGF>>    ;YES, SAY SO
        CALL    ERDO,X3                 ;DO ERROR STUFF
P3G:    TST     .3              ;COMPLIMENT TEST VALUE?
        BRZ     P3X             ;NO
        COM     TMP2            ;YES, DO IT
P3X:    CMPL    ADDR,MSIZ       ;DONE?
        BRP     P3Z             ;YES
        INC     ADDR            ;NO, BUMP ADDRESS
        JMP     P3B             ;AND LOOP
P3Z:    RTRN                    ;RETURN
        HEAD    <PHASES 4 & 5 -- RANDOM DATA>
        SUBR    PH4,1           ;ENTRY POINT
        CLR     ADDR            ;INITIALIZE ADDRESS
        MOV     I2,I            ;INITIALIZE RANDOM KEY
P4A:    CALL    RNDM,<TMP2,I>           ;GET A NUMBER
        CALL    WRPS,<ADDR,X1,TMP2,X1>  ;WRITE IT
        CMPL    ADDR,MSIZ       ;LAST ONE?
        BRP     P4B             ;YES
        INC     ADDR            ;NO, BUMP ADDRESS
        JMP     P4A
P4B:    CMPL    .1,X5           ;TIME-DELAY?
        BRN     P4C
        CALL    WAIT,RFDL       ;GIVE REFRESH TIME TO FAIL
P4C:    CLR     ADDR            ;INIT
        MOV     I2,I
P4D:    CALL    RNDM,<TMP2,I>           ;GET A NUMBER
        CALL    RDPS,<ADDR,X1,TEMP,X1>  ;READ
        CMPL    TEMP,TMP2       ;SAME?
        BRZ     P4E             ;YES
        CALL    SOCT,<MSGA,.1>          ;NO, WRITE PHASE NUMBER
        CALL    SMES,<MSGG,<1,MSGG>>    ;WRITE ERROR MSG.
        CALL    ERDO,.1
P4E:    CMPL    ADDR,MSIZ       ;LAST ONE?
        BRZ     P4F             ;YES
        INC     ADDR            ;NO, BUMP ADDRESS
        JMP     P4D             ;LOOP
P4F:    MOV     I,I2            ;BUILD NEW KEY
        CALL    RNDM,<TMP2,I2>
        RTRN                    ;RETURN
        HEAD    <PHASES 6 & 7 -- BIT DISTURB 1'S & 0'S>
```

```
        SUBR    PH6,1           ;ENTRY POINT
        MOV     CDIF,TTTT
        CLR     TEMP
        CLR     TMP2
        COM     TMP2
P6A:    DEC     TTTT
        MOV     TMP2,<TTTT,COMP>
        DEC     TTTT
        MOV     TEMP,<TTTT,COMP>
        BNZ     P6A
        MOV     CDIF,N                  ;GET ACTUAL USABLE SIZE
        DEC     N
        CMPL    .1,X6           ;THIS TEST BIT DISTURB 1'S?
        BRZ     P6B
        CLR     TMP2            ;(DISTURB 0'S, CLEAR TEST WD)
P6B:    CALL    P3A,<X0,X0,X0>      ;FILL WITH 1'S (OR 0'S)
        CLR     ADDR            ;FOR EACH MEMORY LOCATION, DO:
P6C:    TST     TMP2            ;DISTURB 1'S?
        BRZ     P6D
        CALL    WRPS,<ADDR,N,COMP,X1>    ;YES, WRITE COMPL TBL
        JMP     P6E
P6D:    CALL    WRPS,<ADDR,N,<1,COMP>,X1> ;(NO, "     )
P6E:    MOV     ADDR,I                  ;SET UP FOR
        MOV     ADDR,I2                 ;HOUSE-TO-HOUSE SEARCH
        ADD     X2,I
        ADD     X200,I2
        AND     MSK1,I
        AND     MSK2,I2
        MOV     ADDR,PRTB       ;SAVE ADDRESS
        MOV     I2,M2           ;GET INITIAL COLUMN
        SUB     X400,M2
P6F:    ADD     X100,M2                 ;BUMP COLUMN
        AND     MSK2,M2                 ;MASK OFF EXCESS
        CMPL    M2,I2           ;DONE?
        BRZ     P6H
        MOV     I,M             ;NO, GET INITIAL ROW
        SUB     X4,M
P6G:    INC     M               ;BUMP ROW
        AND     MSK1,M                  ;MASK OFF EXCESS
        CMPL    M,I             ;DONE?
        BRZ     P6F
        AND     MSK3,ADDR       ;NO, MASK ROWS & COLUMNS
        OR      M2,ADDR                 ;RE-CONSTRUCT ADDRESS
        OR      M,ADDR
        CMPL    ADDR,PRTB       ;THIS THE TEST LOC?
        BRZ     P6G             ;YES, SKIP IT
        CMPL    MSIZ,ADDR       ;NO, WITHIN RANGE?
        BRN     P6G             ;NO, SKIP IT
        CALL    RDPS,<ADDR,X1,TEMP,X1>  ;NO, SEE IF 1 (OR 0)
        CMPL    TEMP,TMP2
        BRZ     P6G             ;YES
        CALL    SOCT,<MSGA,.1>      ;NO, WRITE PHASE NUMBER
        CALL    SMES,<MSGH,<1,MSGH>>    ;WRITE ERROR MSG.
```

```
        CALL    ERDO,.1             ;DO ERROR PROCESSING
        CALL    WRPS,<ADDR,X1,TMP2,X1>  ;RESTORE BAD LOCATION
        JMP     P6G             ;NEXT NEIGHBOR
P6H:    MOV     PRTB,ADDR       ;DONE, RESTORE TEST LOC.
        CALL    WRPS,<ADDR,X1,TMP2,X1>
        CMPL    ADDR,MSIZ       ;LAST LOC. IN MEMORY?
        BRZ     P6X             ;YES
        INC     ADDR            ;NO, ADDR = ADDR + 1
        JMP     P6C             ;DO NEXT LOC.
P6X:    RTRN                    ;DONE
        FIN     STRT
```