COURSE OUTLINE

PICTURE SYSTEM II
MAINTENANCE TRAINING COURSE

CUSTOMER ENGINEERING DEPT.

EVANS & SUTHERLAND COMPUTER CORP.

WEEK #1

		MONDAY 9/17	AFTERNOON TUESDAY - 9/18	PROGRAM [illegible] WEDNESDAY - 9/19	THURSDAY - 9/20	FRIDAY - 9/21
8:30 9:20	1	COMPUTER SYSTEMS REVIEW Ed Wild	Introduction to PDP-11 Operating Systems Bill Roach	The Picture Controller Interface Vince Risalvato	Introduction to PS-2 Processor Ed Allred	Theory of Operation: PSMEM Real Time Clock Refresh Controller Vince Risalvato
9:30 10:20	2				The Matrix Arithmetic Processor Vince Risalvato & Ed Wild	
10:30 11:20	3	Introduction to PDP-11 Computer Bill Roach				
11:20 1:00		LUNCH	LUNCH	LUNCH 2:00	LUNCH	LUNCH
1:00 1:50	4	Introduction to PDP-11 -continued - Bill Roach	Review of TTL Logic Concepts Ed Wild	PROGRAMMING EXERCISE DEMO PS-2 INTERFACE REGISTERS LAB SESSION Vince Risalvato	Matrix Arithmetic Processor Vince Risalvato	Introduction to Line Generator Ed Wild
2:00 2:50	5		Introduction to E&S Design Techniques Ed Wild			Introduction to PS-2 Diagnostic System Becky Spitz (Software [illegible]
3:00 3:50	6	GRAPHICS FUNDAMENTALS Ed Wild	UNIBUS Theory and Practices Brian McBride	PSBUS Theory and Structure Vince Risalvato		
4:00 4:50	7	INTRODUCTION TO PS-2 Vince Risalvato	PDP-11 - LAB Session Leon Soren & Vince Risalvato		MAP LAB SESSION Vince Risalvato & Ed Allred	Diagnostic LAB Session Becky Spitz & Bill R.
					To MAURAK	DINNER SESSION CLASS ET-ALL

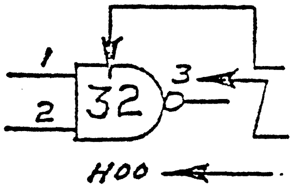
WEEK #2

SESSION	MONDAY - 9/24	TUESDAY - 9/25	WEDNESDAY - 9/26	THURSDAY - 9/27	FRIDAY - 9/28
8:30 1 9:20	Interface Troubleshooting LAB Session Vince Risalvato and Ed Wild	Light Pen and Remote Terminal Interface Vince Risalvato	Line Generator Arithmetic Logic Ed Wild	Dispaly Drive Card Ed Wild	Character Generator Theory of Operation Ed Wild
9:30 2 10:20				Picture System Displays B/W - Color Ed Wild	
10:30 3 11:20					Line Generator Tuning Procedures Ed Wild
11:20 1:00	LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
1:00 4 1:50	PS-2 Peripheral Devices Theory of Operation Vince Risalvato	Picture Processor Troubleshooting LAB Session Ed Wild and Vince Risalvato	Review of Analog Circuitry Ed Wild	Line Generator Control Logic Ed Wild	Line Generator Tuning and Alignment LAB Session Ed Wild and Vince Risalvato
2:00 5 2:50			Line Generator Analog and Display Driver Card Ed Wild		
3:00 6 3:50				Line Generator LAB Session Ed Wild and Vince Risalvato	
4:00 7 4:50	PS-2 Configurations Vince Risalvato	Question and Answer Vince R. & Ed W.	Line Generator LAB Session Ed Wild and Vince R.		

WEEK #3

SESSION	MONDAY - 10/1	TUESDAY - 10/2	WEDNESDAY - 10/3	THURSDAY - 10/4	FRIDAY - 10/5
8:30 1 9:20	PS-2 Diagnostics QSDDT Becky Spitz	QSDDT Homework Problem Debug Bill Roach and Becky Spitz	Multi-User Refresh Controller Theory of Operation Vince Risalvato	PS-2 MPS Graphics Software Package Leon Soren	Acceptance Test LAB Session Leon Soren and Brian McBride
9:30 2 10:20					
10:30 3 11:20	Programming with QSDDT Becky Spitz	QSDDT Solution Presentation Becky S./Bill R.		Driver Configuration LAB Session Leon S./Becky S.	
11:20 1:00	LUNCH	LUNCH	LUNCH	LUNCH	LUNCH
1:00 4 1:50	Line Generator Troubleshooting LAB Session Ed Wild and Vince Risalvato	Remote Terminal Interface Light Pen LAB Session Ed Wild and Vince Risalvato	Multi-User Refresh Controller Vince Risalvato	System Troubleshooting LAB Session Vince Risalvato and Leon Soren	Course Critique Class
2:00 2:50			System Troubleshooting LAB Session Vince Risalvato and Ed Wild		
3:00 3:50					
4:00 4:50		Multi-User Operating Systems Bill Roach			

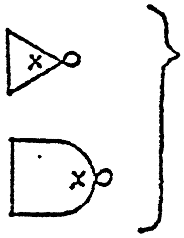
EVANS & SUTHERLAND DRAWING SYMBOLOGY



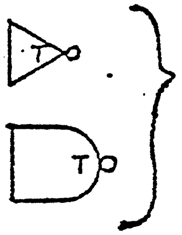
I.C. Position on Card

I.C. Pin Number

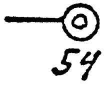
I.C. Type Number (74 HOO)



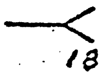
Indicates open collector



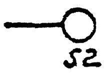
Indicates TRI STATE output



Circuit Card Input or Output pin (54) to Backpanel



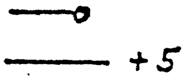
Circuit Card Test Point; Usually designates connector on rear of card. Test points may be located at any physical point on analog cards.



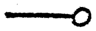
Reference to signal also shown on sheet 2 of logic diagram

*

Designates signal which is active during low voltage state



VCC Connection



GROUND Connection

ENGINEERING CHANGE REQUEST/ORDER

EVANS & SUTHERLAND COMPUTER CORP.

ECO A-9

PAGE
OF

DOCUMENT TITLE
Picture Processor Assembly

BASIC DOC NO.
149101-100 A-8

REASON FOR CHANGE To add segmentation to the R.M. and add the NOP instructions.

COMPLETE WORK B
DATE ______

OTHER DOCUMENTS AFFECTED

149131-101-NC	149131-102-NC

EFFECTIVITY As required if purchased.

ENG DOCUMENT AFFECTED		COMP
TOP ASSY ---- -100	□	
MECH ASSY---- -200	□	
SCHEMATIC DIAG ---300	☒	
MFG BREAKOUT---400	□	
ARTWORK ------500	□	
LOGIC DIAG------600	☒	
TEST PROC ENG TEST---700	☒	
WIRE LIST------800	☒	
OCK DIAG-----900	□	
ALGORITHM---- -950	□	
DETAIL PART----00	□	
E & S PARTS LIST---	□	
GUZZINTA--------	☒	
LOGIC DECK ------	☒	
TOOLING --------	□	
DRILL TAPE ENG E.S.------	□	
MAINT MANUAL-----	□	
REF MANUAL-----	□	
OTHER (SPECIFY)		

DESCRIPTION OF CHANGE

This ECO adds segmentation capability to the Refresh Memory and implements the NOP instructi
One of two other ECO's must be implemented dep ing if the system is 8K or 16K

8K - 149131-101 - A1
16K - 149131-102 - A1

This ECO changes a Picture Processor 149101-10 A8 to a 149101-100 - A9 which is identical to 149101-101 - NC.

The following two cable assemblies must be changed.

149209-006-NC replaces 149123-006 - NC
149210-006-NC replaces 149124-006 - NC
149113-101-NC replaces 149113-100 - A3
149114-101-NC replaces 149114-100 - A3

ANCED COPY
DATE..........
BY..........

REQUESTED BY Craig R | DATE [illegible]
PREPARED BY EJ A. | DATE [illegible]

DISTR	SIGNATURES REQD	DATE	NO OF COPIES
☒	ENG SER		1
☒	PROJ ENG		1
☒	MANUF		1
	C		1
☒	CUSTOMER ENG		1
☒	MARKETING		1

ECO CLASS
AFFECT FORM FIT OR F

PARTS DISPOSITION

	USE AS IS	REWORK	REPLACE	SCRAP
IN STOCK				
IN PROCESS				

,THERLAND
,ER CORP.

ENGINEERING CHANGE REQUEST/
ENGINEERING CHANGE ORDER

PAGE 2 OF 3 | ECO A

DOCUMENT TITLE Picture Processor Assembly

BASIC DOC 149101-100

ECR/ECO CONTINUATION SHEET

ADD - DELETE WIRE LIST

DELETE

DC 48.62, 51.73, 55.22 $

*DC 51.75, 54.32 $

ADD

DC 48.62, 55.22, 54.32 $

DMAIN(0) 16.78,43.33,46.48,48.24,54.4

DMAIN(1) 16.59,43.30,46.47,48.25,54.6

DMAIN(2) 16.58,43.29,46.60,48.33,54.8

DMAIN(3) 16.57,43.25,46.54,48.34,54.1

DMAIN(4) 16.34,42.33,46.26,48.36,54.1

DMAIN(5) 16.35,42.30,46.53,48.37,54.1

DMAIN(6) 16.36,42.29,46.28,48.47,54.2

DMAIN(7) 16.37,42.25,46.59.,48.48,54.2

DMAIN(8) 11.33,16.30,46.32,48.55,54.3

DMAIN(9) 11.30,16.29,46:64,48.56,54.4

DMAIN(10) 11.29,16.28,46.30,48.57,54.

DMAIN(11) 11.25,16.27,46.63,48.58,54.

DMAIN(12) 10.33,16.26,46.13,48.59,54.

DMAIN(13) 10.30,16.25,46.16,48.60,51.

FSM1(0) 9.6,12.74,45.15,48.38,47.35

FSM1(1) 9.5,45.13,48.54,47.36 $

FSM1(2) 9.4,12.75,48.53,47.43 $

FSM2(0) 13.38,45.14,48.40,47.25 $

FSM2(1) 13.37,45.12,48.39,47.26 $

WADR(0) 47.3,54.3 $

WADR(1) 47.4,54.5 $

WADR(2) 47.6,54.7 $

WADR(3) 47.12,54.9 $

WADR(4) 47.17,54.13 $

WADR(5) 47.39,54.15 $

WADR(6) 47.45,54.17 $

WADR(7) 47.46,54.19 $

WADR(8) 47.57,54.23 $

WADR(9) 47.58, 54.27 $

WADR(10) 47.67, 54.33 $

WADR(11) 47.68, 54.45 $

WADR(12) 47.69, 54.49 $

CUSTOMER ENGINEERING

INSTRUCTION 1

EQUIPMENT SERVICE LOG

The Customer Engineer is required to maintain an Equipment Service Log for each system/equipment he is responsible for servicing. It is recommended that the logs for each system/equipment be kept in a master loose leaf type book.

1.0 PAGE FORMAT

Each page of the log shall have in the upper right hand corner the following information in the format shown.

Customer/Site ______________________

System/Equipment ______________________

Serial No. ______________________

Date of Service ______________________

Customer Engineer ______________________

2.0 INFORMATION LOGGED

The information should be entered as follows:

2.1 Symptoms of Failure -

a) Customers description of the failure

b) Your description of the symptoms if different and/or more detailed.

2.2 Time Required to Locate Problem -

The time required by you to diagnose and locate the cause.

2.3 How I Found the Problem -

The diagnostics you used. Were you lead astray by anything? Generally the difficulty you had in finding the cause.

2.4 What Failed -

The card type and bug, component, back panel, wire, etc.

2.5 What Was Repaired -

What was replaced, adjusted, etc. Include what you replaced or repaired in attempting to find the problem even if it didn't fix the problem.

2.6 Difficulty and Time Required to Make Repair -

For example: "ten minutes to replace bug. Two hours to remove fan, filter and bracket so I could reach it."

2.7 Suggestions -

What you found deficient in the diagnostics, documentation, equipment design, tools, etc.

Suggestions for making the task easier.

2.8 General -

Any observations you made or conversations you had with the customer which would be of value to E&S. What is he doing with the system; what is he planning to do, his complaints, compliments, etc.

2.9 Report to Headquarters -

The Customer Engineer is responsible for sending a copy of all additions to the log for those equipments which had service performed on them during a calendar month. The data for a given month will be sent to the Director of Customer on or before the 5th of the following month.

CE-2.1
Orig. 2/19/75
Rev. ________

CUSTOMER ENGINEERING
INSTRUCTION 2
ECO FIELD CONFIGURATION CONTROL

This instruction sets up a procedure for establishing system configuration at the time of shipment from the factory and for maintaining a continuous record of configuration.

1.0 ECO FIELD CONFIGURATION CONTROL CARD SET

A set of ECO FIELD CONFIGURATION CONTROL cards, CONFIG CARDS, will become and remain a part of all systems or subsystems that are shipped from the factory. This includes equipment that are sold, loaned or shipped for use by E&S personnel at shows and demonstrations. The CONFIG CARD sets will be physically located within the cabinet containing the system or subsystem. A sample copy of a CONFIG CARD with instructions for establishing and maintaining the card is given in section 2.4 below.

2.0 INITIAL PREPARATION AND VERIFICATION

2.1 Preparation

Prior to the factory acceptance test, FAT, Quality Control will have a set of CONTROL CONFIG cards prepared based upon their equipment configuration summaries.

2.2 Verification

After FAT and prior to shipment the Customer Engineer responsible for conducting the FAT will verify that the CONTROL CONFIG cards are correct. He will make a copy of the card set for C.E. Dept. records.

3.0 MAINTENANCE OF THE CARD SET

The Customer Engineer responsible for installation, maintenance and servicing of the equipment is responsible for keeping the cards up to date. All changes which effect equipment configuration will be recorded.

4.0 FILLING OUT THE CARDS

A sample ECO FIELD CONFIGURATION CONTROL form CED50030 is shown below. It lists the system components by slot number or by an identifying name. The second column contains the E&S part number and name. The first ECO/DATE column contains the ECO level and data when the system was first shipped from the factory, 3/6/75. The following description of the data listed in the rows for SLOT NO. 9 illustrates how the cards are to be updated.

SLOT NO. 9

a) When the system was shipped on 3/6/75 slot 9 contained an ARITH I, 144106-100, serial number 19 with ECO level A3.

b) On 4/9/75 the 144106-100, serial number 19 had ECO A4 installed.

c) On 5/29/75 the 144106-100, serial number 19 was replaced with serial number 31. This replacement is indicated by the RP, for replaced, in column 3 and the addition of 31 in the serial number column, S/N.

d) On 9/2/75 the 144106-100, serial number 31 had ECO A5 installed.

CE-2.3

EVANS & SUTHERLAND
COMPUTER CORPORATION
SALT LAKE CITY, UTAH 84112

ECO FIELD CONFIGURATION CONTROL

SUBSYSTEM PICTURE SYSTEM PROC. 101000-085 S/N 52
NAME NUMBER

SLOT NO.	CARD/ASS'Y NO. ASSEMBLY NAME	ECO DATE															S/N
6	144105 - 100	A2															22,
	SUPER CONTRL.	3/6/75															
9	144106 - 100	A3	A4	RP A4	A5												19,31,
	ARITH I	3/6/75	4/7/75	5 29 75	9 2 75												
Power	801369 - 001	NC															106,
Supply		3/6/75															
CABLE	145120 - 502	NC															3,
ASMB		3/6/75															
BACK	149102 - 100	NC	A1	A2	A3	A4											52,
PANEL		3/6/75	3/7/75	4/2/75	5/6/75	5/6/75											
	-																
	-																
	-																
	-																
	-																

SAMPLE

1. INTRODUCTION TO THE PDP-11

1.1 PDP-11 Architecture

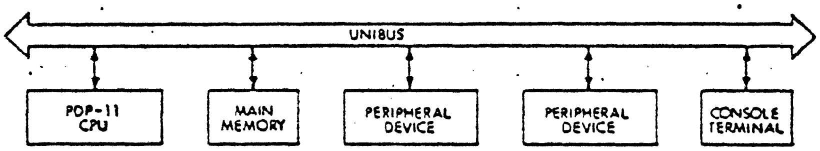


FIG 1.1 Simplified PDP-11 Block Diagram

1.1.1. UNIBUS: high-speed buss used for communication between the other major system components.

1.1.2. CPU, Central Processor: executes intructions and allocates UNIBUS usage.

1.1.3. Main Memory: storage area for programs and data, byte or word addressing. Stacks At 000 To 500 8

1.1.4. Peripherals: all I/O devices are interfaced to the UNIBUS.

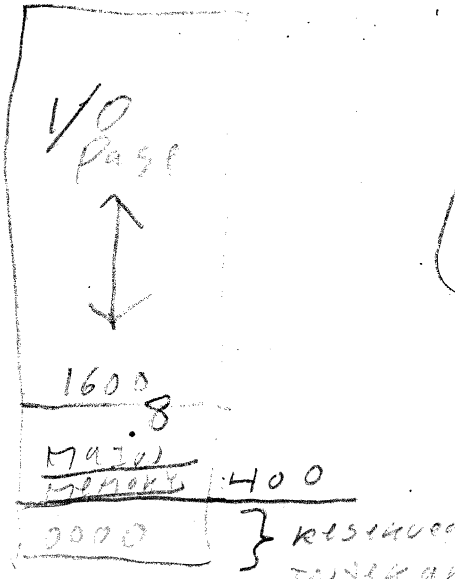
Words Always have even Addresses (Inc. 2)

LSB = Right MSB = Left

16 Bit word
8 bits
8 bits
0001
0002

(even only words)

(Bytes may be even or odd)



1.2. PROGRAMMING ELEMENTS

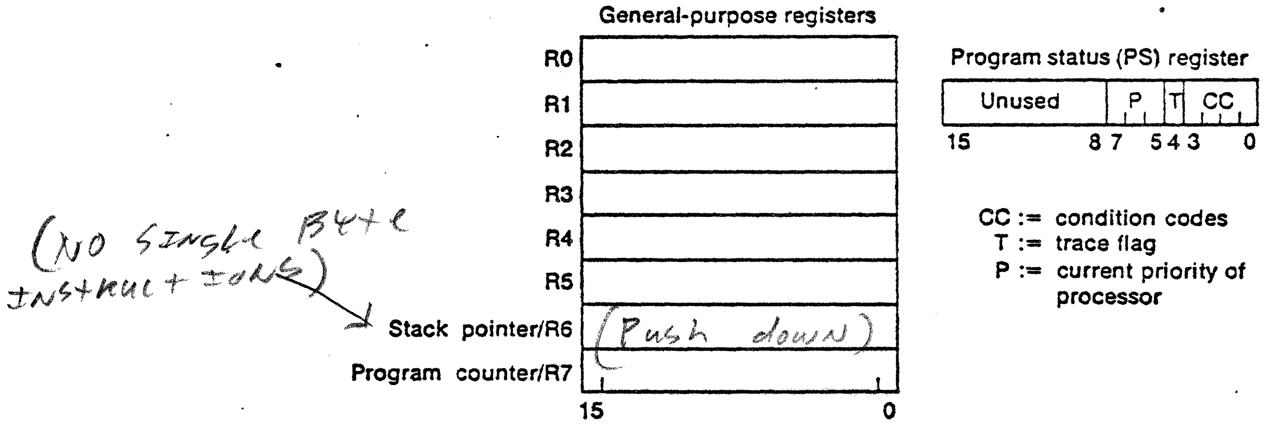


FIG 2.1 The Processor State of the PDP-11

1.2.1. General Registers: 8, of which R0 through R5 are for general use as accumulators, index registers, stack pointers, etc.

1.2.2. R6 or SP: System stack pointer. The stack is used to save registers and for subroutine linkage, etc.

1.2.3. R7 or PC: The Program Counter increments through the program, always pointing to the next instruction, operand or index, etc.

1.2.4. PSW, Processor Status Word: Contains processor priority, addressing information, and conditions pertaining to the last operation performed.

1.2.5. DMA, Direct Memory Access: Block data transfers between memory and peripherals may be initiated and then run to completion without software supervision.

1.2.6. NPR, Non-processor Request: A request for access to the UNIBUS which is issued by a peripheral rather than the CPU. Used in DMA transfers.

1.2.7. Priority Interrupts: Asynchronous events (e.g. DMA completion) may cause an interrupt to the currently executing program and activation of a Service Routine. This frees the CPU from the duty of polling peripherals. Only an interrupt of priority higher than the current processor priority (found in the PSW) is allowed to take effect.

1.2.8. PDP-11 Instructions: An instruction specifies the operation to be performed (e.g. move, add), method of addressing the source operand, and/or method of addressing the destination operand.

1.2.9. General Addressing Modes:

DIRECT MODES

Mode	Name	Assembler Syntax	Function
0	Register	Rn	Register contains operand.
2	Autoincrement	(Rn) +	Register contains address of operand. Register contents incremented after reference.
4	Autodecrement	—(Rn)	Register contents decremented before reference register contains address of operand.
6	Index	X(Rn)	Value X (stored in a word following the instruction) is added to (Rn) to produce address of operand. Neither X nor (Rn) are modified.

DEFERRED MODES

Mode	Name	Assembler Syntax	Function
1	Register Deferred	@Rn or (Rn)	Register contains the address of the operand
3	Autoincrement Deferred	@(Rn) +	Register is first used as a pointer to A word containing the address of the operand, then incremented (always by 2; even for byte instructions)
5	Autodecrement Deferred	@—(Rn)	Register is decremented (always by two; even for byte instructions) and then used as a pointer to a word containing the address of the operand
7	Index Deferred	@X(Rn)	Value X (stored in a word following the instruction) and (Rn) are added and the sum is used as a pointer to a word containing the address of the operand. Neither X nor (Rn) are modified

1.2.10. PC Addressing Modes (Register=7):

PC ADDRESSING

2	Immediate	#n	Operand follows instruction
3	Absolute	@#A	Absolute address follows instruction
6	Relative	A	Address of A, relative to the instruction, follows the instruction.
7	Relative Deferred	@A	Address of location containing address of A, relative to the instruction follows the instruction.

1.2.11. Programmer's Console: used for manual examination and modification of memory contents, for starting programs, etc. May not be available on PDP-11/04, 34, or 60.

1.2.12. Sample Program: Bootstrap the RK05 Disk Pack

```
12700   MOV     #177406,R0
177406
12710   MOV     #177400,(R0)
177400
12740   MOV     #5,-(R0)
5
105710  TSTB    (R0)
100376  BPL     .-2
5007    CLR     PC
```

2. OPERATING SYSTEMS

2.1. Some OPERATING SYSTEM Definitions

"A set of programs and routines which guide a computer in the performance of its tasks and assist the programs (and programmers) with certain supporting functions."--SAYERS

"Software which controls the execution of computer programs and which may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and related services."
--AMERICAN NATIONAL STANDARD DEFINITION

"That programming which is provided by the vendor of a computing system as an integral part of the product he markets"--KURZBAN, HEINES, and SAYERS

"Operating systems have two basic functions: they provide services for application program development and act as an environment in which application programs run. The character that an operating system has, that is, the services and environment it supplies, is appropriate only for a certain range of development and application requirements in order to serve selected needs efficiently.... AN OPERATING SYSTEM IS A COLLECTION OF PROGRAMS THAT ORGANIZES A SET OF HARDWARE DEVICES INTO A WORKING UNIT THAT PEOPLE CAN USE"--DEC

"Operating systems are intended to facilitate efficient use of computers. They provide a convenient interface to hide from programmers the complexity of the bare computing systems. They manage the resources of computing systems so that the resources are optimally utilized"
--KURZBAN, HEINES, and SAYERS

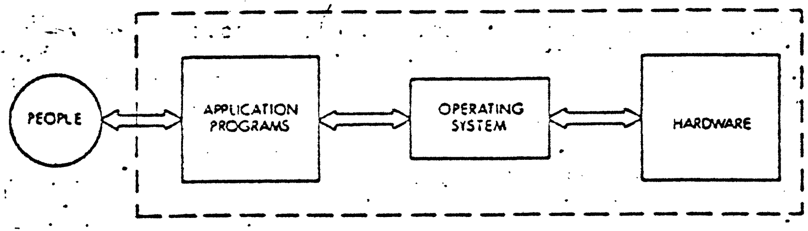


FIG 2.1 Computer System Components

IN SUMMARY:

A Collection of programs; a software system

A "soft" interface to a specific computer

Tailored to a specific group of applications

Purpose: to make the computer system usable

2.2. Typical Operating System Components

2.2.1. Management Program: "Executive" or "Monitor"

2.2.2. Utilities

Program Development Support: Editor, Assembler, Compiler, Linker, Librarian, Debugging Tools.

System Maintenance: Device Verification, Accounting

2.2.3. I/O Support and Drivers

2.2.4. File Management and Support: Open, Close, Read, Write, etc.

2.3. Attributes of Operating Systems

Single-User

Multi-User

Real-Time: Must perform to real-time constraints, for example, refresh rate, update rate, or sampling rate.

Disk-Based: All of the more powerful operating systems use a fast mass storage device as the system device, usually a disk-pack. They must do so if they are to be reasonably fast.

2.4. RT-11 Single-Job System: Single-user, fast, suitable for real-time applications. All files are contiguous, which is one major reason for the system's speed. It is relatively simple to understand and use. Comparable to DOS in range of applications, but much faster.

2.4.1. Utilities: EDIT, MACRO Assembler, FORTRAN Compiler, PIP File Management, LINK, LIBR, and ODT Debugger.

2.4.2. Sample RT-11 Memory Layout:

```
        ***************************************
0-36    *       SYSTEM TRAP VECTORS           *
        ***************************************
40-56   *       USER COMMUNICATION AREA       *
60-376  *       OPTIONAL INTERRUPT VECTORS    *
400-476 *       SYSTEM OPERATIONS             *
        ***************************************
500-776 *       STACK                         *
        ***************************************
1000+   *       USER PROGRAM                  *
        *-------------------------------------*
        *       DEVICE HANDLERS, REQUESTED BY *
        *       USER PROGRAM                  *
        *-------------------------------------*
        *       USER BUFFER AREA              *
        ***************************************
        *       (UNUSED)                      *
        ***************************************
        *       KMON- KEYBOARD MONITOR        *
        *       MAY BE OVERLAID BY USER PROGRAM *
        ***************************************
        *       USR-"USER SERVICE ROUTINE"    *
        *       MAY BE OVERLAID BY USER PROGRAM,*
        *       THEN SWAPPED IN DUE TO PROGRAM *
        *       REQUEST.  INTERPRETS FILE SPEC- *
        *       IFICATION STRING, OPENS & CLOSES*
        *       FILES.                        *
        ***************************************
        *       RMON- RESIDENT MONITOR, MUST  *
        *       REMAIN IN MEMORY AT ALL TIMES *
        ***************************************
I/O page |      upper 4k                      |
```

2.5. Evans & Sutherland Diagnostic Monitor:

Similar to RT-11, with compatible file structure and device support. Unlike RT-11, ESD does not include program-development support. Utilities: PATCH, to modify loadable programs, and UPDATE, comparable to PIP. Like RT-11, it can use Dectape as the system device.

PS2 - IO/status = 167660

2.6. RSX-11M Operating System: Multi-user system which supports system devices which are not supported by RT-11 or ESD, e.g. RP04 Disk Pack. A very powerful system which is difficult to use and still more difficult to understand as compared to RT-11. The system is available in a "Mapped" version which requires the KT-11 Memory Management unit and provides memory management, protection, and dynamic relocation, and an "Unmapped" version which does not require the KT-11, is simpler and is less powerful.

2.6.1. Utilities: Most utilities in DOS and RT-11 are available as well as others such as FLX, DSC, and PRESRV.

2.6.2. Partition: the memory block into which a task (program) is loaded.

2.6.3. Multiprogramming: sharing of the CPU among two or more tasks which reside simultaneously in memory (in different partitions).

2.6.4. Checkpointing: the process of saving an active task image on the system device so that another higher priority task may occupy the partition.

2.6.5. Scheduling: one of the RSX Executive's most difficult jobs; the job of deciding who gets which resource (CPU, Memory Partition, Peripheral) next.

2.6.6. Unmapped RSX-11M as a PS2 Diagnostics System: used at sites which do not have RK05 Disk Drives but some other drive which is supported by RSX-11M such as RP02 through RP06, RK06 and RX06.
In this context RSX should be regarded as a single-user system.

2.6.7. Mapped RSX-11M and PS2 Maintenance:
The most important difference between a PS2 Graphics task under RT-11 and the "equivalent" task under RSX-11M is generally a difference of timing.
Timing changes may expose errors in the hardware or in the Graphics Software Package.
Also, there is a Picture System Driver for Mapped RSX-11M which has no counterpart in unmapped RSX or in RT-11.

3. Picture System II Diagnostic Programs

3.1. "Machine Independence"

3.2. Specified by the Design Engineer

3.3. Documentation: PS2 Hardware Diagnostics Manual (PS2 HDM)

3.4. Standard Operator Interface:

H	Help
P	Pass Count
D	Do Phases
X	Execute
L	Loop on Error
C	Loop on Error, and Continue
M	Modify Device Addresses

3.5. Error Messages:

```
1: DOIT ADDR ERR; ADDR=XXXXXX EXPT=XXX (cont)
        RECD=XXX INDEX=23
```

3.6. MIXIT Programming Language

3.6.1. General Instructions: MOV, ADD, ADD2, SUB, SUB2, INC, DEC, CLR, COM, AND, OR, SLS, SRS, SLD, SRD.

3.6.2. Test and Branch Instructions: CMPL, CMPA, TST, JMP, BRZ, BRN, BRP.

3.6.3. Data Storage Instructions: BLOCK, DATA, CDATA, DIFF.

3.6.4. Subroutine Declaration:

```
SUBR    A,N
  A = Subroutine Name
  N = Number of arguments
```

3.6.5. Subroutine Call:

```
CALL    A,<X,Y...>
```

3.6.6. Subroutine Return: RTRN

3.6.7. Global Declarations: HERE, THERE

3.6.8. Subroutine Arguments: .1, .2 etc.

3.6.9. Indexing: first argument (index) may be number, name, or subroutine argument; second argument (base address) may be name or subroutine argument. Examples:

```
CALL    <T1,TABL>
MOV     <1,ZERO>,<.1,TAB2>
```

3.6.10. Indirect Addressing: <arg,>

3.6.11. Utility Subroutines:

RDPS	Read PS
WRPS	Write PS
READ	Read Interface Register
WRIT	Write Interface Register
CINT	Connect Interrupt
DINT	Disconnect Interrupt
DMA	Initiate DMA Transfer

3.6.12. Example: QSD002.MIX, QSD002.LST

3.7. MNEMONIC: Command language for table-driven MIXIT diagnostics.

3.7.1. Interpreter Call:

```
CALL    CODE,<TABL>
```

3.7.2. I/O Commands: lxyz

3.7.3. Mask Command: 0001

3.7.4. Mark Command: 0010

3.7.5. Loop Control: 0011, 0110

3.7.6. Address Incrementation Switch: 0100

3.7.7. Random Number Initialization: 0101

3.7.8. Special Commands: 0111

0	Enable Random Number Gen.
1[,n]	Disable Random, Enable Frozen Data
2,...	Extension of the "Mark" Command
3[,..]	Toggle Table Data Switch
4,..	Load the PS Address Bucket
5,SUBR	Call Subroutine SUBR
6,DATA	Load DOIT
7,DATA	Load DOIT, Then Clock the MAP

3.7.9. Example: QSD004.MIX

3.8. Troubleshooting with QSDDT: see Appendix F, and examples.

3.9. E & S Diagnostic Monitor (ESD) (RT11-V1) UTILITIES removed [illegible]

3.9.1. UPDATE

3.9.2. PATCH

3.10. Diagnostic Operation Under the RSX-11M Unmapped Operating System: see Appendix D

3.10.1. Task Termination

June 25, 1976

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
```

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
.DIFF     a,b              ;define a word of data +b-a (offset
                           ;in words, a and b must be names)
```

Subroutine 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,...>           ;defines global entry points
.THERE    <a,b,...>           ;defines external globals
```

Miscelleneous 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
```

Program Test Word

When a .CMPA, .CMPL or .TST instruction is specified, the resulting zero/nonzero, positive/negative value is placed in the Progra 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
```

```
        JMP     STRT            ;FOR MANUAL STARTUP
        HEAD    <SAMPLE MAIN PROGRAM>
MSGS:   DATA    2               ;MESSAGE BLOCK FOR "INIT"
        DATA    <MS1,10.>       ;ANNUNCIATION MESSAGES
        DATA    <MS2,15.>
        DATA    3               ;"H" RESPONSE MESSAGES
        DATA    <MS3,15.>
        DATA    <MS4,16.>
        DATA    <MS5,13.>
          .
          .
MS1:    CDATA   <QSD000.S01>    ;THE ACTUAL MESSAGES
MS2:    CDATA   <DEMO DIAGNOSTIC>
MS3:    CDATA   <THESE LINES ARE>
MS4:    CDATA   <OUTPUT ONLY WHEN>
MS5:    CDATA   <"H" IS TYPED.>
          .
          .
          .
        THERE   <SMES,SOCT,GETS,GETN,WRPS,RDPS,DOPH,PHAZ,DPCH,ERRL,INIT>
PTBL:   DATA    <0,PH1,PH23,PH23,PH4,...>       ;KEEP THESE TWO--
PMAX:   DIFF    PTBL,PMAX                       ;TOGETHER. (SIZE OF PHASE TABL )
ERMX:   DATA    12.                     ;HIGHEST ERR.MSG.NO. (.LE.16.)
          .
MS9:    DATA    18.                     ;KEEP THESE TWO--
        CDATA   <ALL TESTS COMPLETE>    ;--TOGETHER.
          .
MS10:   DATA    47.
        CDATA   <3: ERSATZ ERR; ATTERCOP DOES NOT EQUAL TOBNODDY>
          .
X:      DATA    <0,1,2,3,4,5,6,...>
GOOD:   BLOCK   1               ;TEMPORARY STORAGE
TEST:   BLOCK   1               ;TEMPORARY STORAGE
        HEAD    <SETUP CODE>
STRT:   MOV     PMAX,DOPH       ;START HERE
        DEC     DOPH            ;COMPUTE MAXIMUM NO. OF PHASES (GLOBAL)
        MOV     ERMX,PHAZ       ;GET HIGHEST MSG. NO. (GLOBAL)
        CALL    INIT,<MSGS>     ;GO DO OPERATOR DIALOGUE
          .
          .                     ;DO TEST GROUP INITIALIZATION
          .
        CALL    DPCH,<PTBL>     ;CALL THE DISPATCHER
        CALL    SMES,<MS9,<1,MS9>>      ;REPORT COMPLETION OF ALL TESTS
          .
          .
        STOP                    ;RETURN TO MONITOR
          .
          .
        HEAD    <PHASE 1 CODE>
        SUBR    PH1,1           ;PROBABLY WON'T USE THE ARGUMENT
          .                     ;(ARG.=PHASE #)
          .
        CMPL    GOOD,TEST       ;FAILURE?
        BRZ     PH1A
        CALL    SMES,<MS10,<1,MS10>>    ;YES, REPORT IT
        CALL    ERRL,<<3,X>,REPT>       :ALLOW FOR LOOP-ON-ERROR (TTTT MODIFIED)
PH1A:     .
          .
        RTRN                    ;RETURN FROM PHASE 1 (THIS PASS)
        HEAD    <PHASE 1 ERROR REPEAT SUBROUTINE>
        SUBR    REPT,1          ;REPEATS THE ERROR REPORTED AS MS10
          .                     ;DO THE REPEAT TEST
          .
        CMPL    GOOD,TEST       ;MAKE THE COMPARISON
        MOV     TTTT,.1         ;RETURN RESULT WITHOUT LOOKING
        RTRN                    ;(0=GOOD;NON-ZERO=BAD)
        HEAD    <PHASE 2 & 3 CODE>
        SUBR    PH23,1          ;ENTRY POINT FOR PHASES 2 & 3
          .                     ;(PARAM WILL = 2 OR 3,
          .                     ;RESPECTIVELY)
          .
        FIN     STRT            ;DONE
        JMP     STRT            ;FOR MANUAL STARTUPS
;
;
; PROGRAM:  QSD002.MIX
;
; AUTHOR:  STEPHEN N. MCALLISTER
;
; DATE WRITTEN:  5/14/76
;
; 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    11.
        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.>
MSG1:   CDATA   <QSD002.S01>
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>
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   <REMOVE JUMPER -- CARR. RTRN.>
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
X74:    DATA    74
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,TIME,RNDM>
        THERE   <DPCH,ERRL,PHAZ,DOPH>
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:    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,<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,<MSGJ,<1,MSGJ>>    ;NO, 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
        BRP     P2E
        INC     ADDR            ;NO, BUMP N
        JMP     P2C             ;AND LOOP
P2E:    TST     PRTB            ;PRTB SET YET?
        BNZ     P2L             ;YES
        CALL    SMES,<MSGJ,<1,MSGJ>>    ;NO, ASK FOR JUMPER
        CALL    GETS,<X1,TEMP>  ;WAIT FOR ANSWER
        CLR     ADDR            ;CLEAR N
        INC     PRTB            ;SET PRTB
        JMP     P2A
P2L:    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,<MSGE,<1,MSGE>>    ;YES, SAY SO
        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    TIME,X74         ;60-SECOND DELAY
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
```

```
         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
>
```

Revised 12/21/76

MNEMONIC

(Mixit Nails Errors by Means Of Numerically-Interpretive Code)

This document describes a special, abbreviated coding language for use with MIXIT subroutines, in analyzing and reporting PICTURE SYSTEM hardware errors. Its interpreter consists of a MIXIT subroutine called CODE, which is called by the following calling sequence:

```
        CALL    CODE,<TABL>
```

where the argument passed is the beginning of a table of the form:

```
TABL:   DIFF    TABL,ENDT                   ;SIZE OF TABLE
        DATA    AAAABBBBCCCCDDDD            ;FIRST FOUR 4-BIT COMMANDS
        DATA    <I,J,K,L,M,...>             ;ARGUMENTS FOR ABOVE 4 CMDS.
        DATA    EEEEFFFFGGGGHHHH            ;NEXT FOUR COMMANDS, ETC.
          .
          .
          .
E   :   DATA    TABL                        ;END-OF-TABLE (POINTS TO TABLE HEAD)
```

Note that in the above example, as in the explicit examples below, the commands are shown encoded as individual bits. However, in an actual program, the four commands would be condensed into a single octal value.)

There are several self-checking features in a MNEMONIC program, such as the word at the front of the table indexing to the last word, which s a pointer to the front again. These features must be very carefully observed, as must the required number of arguments per command, and the caution in the "0000" command below. These features help lessen the high possibility of program runaway inherent in pure numeric coding.

The command repertoire of MNEMONIC is discussed below, by command. Labels are usually optional (and unnecessary), but where indicated, they must be used exactly as specified. Lower-case letters indicate numeric r[illegible]ments, while upper-case names signify actual labels or mnemonic MIXIT commands such as DATA or DIFF.

In addressing PICTURE SYSTEM device registers and memory, a le in INIT globally named PSTB is used to allow a diagnostic test a device in an address independant manner. The table contains nters to SCB device addresses or SCB base addresses in the case a device has more than one register. To allow for devices ch use partial fields of a word, for example in the system errupt control block, some addresses have shift counts associated with em which indicate how many places the rightmost bit of the partial ld is shifted left from bit 0.

To specify a PS address, one must provide a reference to an ry in PSTB which specifies the base address of the device and a mber (positive or negative) which indicates the offset from that se address to the desired address. All test data which is to read or written to registers which are divided into device peculiar ields must be considered to be right justified. In this way, data be appropriately shifted depending on the partial field shift nt at execution time.

```
1xyz
DATA     <pstbref,offset,value> or <pstbref,offset> or <value>
```

This is the PICTURE SYSTEM I/O command for MNEMONIC. The three lower-case letters in the command are bits to be set as follows:

x=0: Data value is supplied from another source, such as random number generation or a data table (Sec. 10 & 12), or previously-supplied data is used. For this case "value" is omitted.

x=1: The previous value is saved away, and the "value" argument is used for the I/O data. The saved value is then restored.

y=0: The "pstbref,offset" arguments are omitted, and the PS address previously supplied as described in Sec. 13 is used again, either incremented (see Sec. 6), or not.

y=1: The previous address is saved away, and the "pstbref,offset" arguments are used for the PS address of the I/O.
The saved address is then restored.

z=0: The PSBUS is read at the specified address, and the contents of "value" are exclusive-ORed with it. The result is then ANDed with the mask (see Sec. 3), and a non-zero result signifies that an error has occurred. If so, it is duly reported (see Sec. 11), and the appropriate error loop procedure is executed as defined in the ERRL subroutine of INIT.

z=1: The contents of "value" are written onto the PSBUS at the address specified by "psaddr".

If an error is detected, the current location and command word contents are saved, and if error looping is required, it commences immediately after the most recent Mark command (Sec. 4 & 11), and proceeds back down to the location saved, tallying errors on the way. Thus, if several reads are specified between two Marks, and if the original error occurred at other than the first of these reads, looping will continue if ANY of the reads produces an error. In this case there is no way short of scoping to determine which read is producing the error in subsequent trips through the loop, as the error message is only reported once. Therefore, it is advisable to specify the fewest reads possible between Marks.

```
•         0000
```

This command implies that the remainder of the current Command word is null. If it isn't, a system error is generated, and the program terminates. This command is useful for patching a program, since all command words are required to have exactly four commands.

```
•         0001
          DATA    mask
```

This provides for the loading of a mask for controlling which bits of the word read from the PSBUS are to be matched against the test value. Wherever a zero-bit occurs in the mask, the corresponding bits tested are assumed to agree without matching. A default value of 177777 is provided for this word.

```
•         0010
```

This command is used to "Mark" the current location for later error looping. The command automatically saves the location index, the current Command word, the current random number key, table-load index, "psaddr" and "value". If an error occurs which requires looping, these values are restored, and execution begins immediately following this command. Note that the above are the ONLY values stored, and that the user should beware to place any special commands, such as to enable random data, immediately after the Mark, if such commands are to be reversed or modified before the error test is made. Thus, the proper default conditions will be guaranteed upon returning to the Mark.

```
•         0011
LAB:      DATA    <count,0,0>
```

This command, together with the "Subtract One and Branch (SOB)" command described in Sec. 8, provide loop capabilities to MNEMONIC. The loop is entered at the "0011" command, when the loop count is initialized to the value "count". Thereafter, whenever the "0110" command is encountered, the count is decremented by one, and if it is not zero, control is transferred to the first command following the "0011" command. Since the loop count is maintained at the "0011" command, as many "0110" commands as desired may refer to it. An unconditional branch may be defined by making "count" a very large number, and by replacing the "0" following it with the same large positive number. When used in this manner, however, the "0011" command must be the LAST command in its particular command word (see Sec. 2). Note that no values are saved or restored in looping. It is therefore the user's responsibility to make sure that all modes and values are set as desired upon entering the loop.

```
•         0100
```

Each time this command is executed, an internal switch in the MNEMONIC interpreter is toggled, which switch controls whether "psaddr" (see Sec. 1) is incremented before use. The switch is initially set to non-increment mode. Note that incrementation never applies to addresses that are supplied with the command (i.e. the "y" bit equals 1). Note also that incrementation takes place BEFORE the I/O access. The user should therefore see that, when incrementation is used, the default first address be ONE LESS than the first address desired. (See also Sec. 13.)

```
        0101
```

This command causes another switch internal to MNEMONIC to be toggled, which switch initiates the saving or restoring of the current random number key. Thus, if a string of random numbers were to be written out to the PSBUS, and then the identical string were to be generated again and checked with the read-in data, the user would proceed as follows:

1. Execute the "0101" command.
2. Enable the Random Number Generator (see Sec.9).
3. Do a series of "10y1" commands to write out the data.
4. Execute another "0101" command in order to restore the random number key.
5. Do a series of "10y0" commands to read and compare the incoming data with the new string.

Note that MNEMONIC is initialized such that the first use of the "0101" command initiates a key save, rather than a key restore.

```
        0110
        DIFF    TABL,LAB
```

This command is the second of the pair including the "0011" command. Its use is detailed in Sec. 5 above. In the DIFF statement above, LAB is the label associated with the "0011" command, and TABL is the beginning of the command table (i.e. the argument of the CODE subroutine call).

```
        0111
        DATA    0
```

This is the first of a series of commands whose value is "0111". The first argument of the data statement is the sub-command code, and must always appear exactly as specified. This particular command causes the Random Number Generator to be enabled, such that if a command of the form "10yz" is given, data is supplied as a random number computed from the current key. The key is then updated so that the next random number is unique. Setting this switch also disables the "frozen data" feature (see Sec. 10).

```
        0111
        DATA    <1[,number]>
```

This command disables the Random Number Generator, and also enables the Frozen Data feature. Under this feature (as noted here and in Sec. 12 below), whenever a command of the form "10yz" is encountered, the data used is the data which was last used to write or test the PSBUS. If the Table Load feature (Sec. 12) is enabled, the "number" parameter must be omitted. This parameter supplies the "frozen data" to be used by the "10yz" command. That data can be overridden at any time by using instead the "11yz" form of the command, which contains explicit data. MNEMONIC is initialized with the Frozen Data switch set.

11.

```
        0111
        DATA     <2,MSG,ernum[,MSGSUB]>
```

This command is an extension of the Mark command ("0010") and, when its task is done, transfers control to it. This should ALWAYS be the first command in any program. It is used to set up pointers to the message information associated with this Mark (or series or Marks). The "ernum" parameter is a number which will serve as the first argument of the ERRL call in case of an error, and is the Error Number of the current error (for details, see the description of the "L" and "C" options in the INIT Operator's Guide). MSG is the label of a brief (LT 20 bytes) error type description, of the form:

```
        MSG:     DATA     -n.
                 CDATA    <text>
```

where "n" is the number of bytes in the text. Messages will be of the form:

```
        ernum:   text; ADDR=xxx DATA EXPT=yyy DATA RECD=zzz INDEX=iii
```

If "ernum" is negative, its absolute value will be used, and the optional entry MSGSUB will be accepted. This is the entry point of an optional user-defined MIXIT subroutine which is called immediately after the error message is typed, and in which the user may generate additional error messages before returning to MNEMONIC. The calling sequence will be the same as that described in Sec. 14. The "INDEX" referred to is the number of words down the table where the command is located FOLLOWING the read command which encountered the error. It may be used in debugging programs, or in correlating the error to the part of the program which detected it in cases where the same "ernum" and "text" occur in several places. INDEX is an octal value.

12

```
        0111
        DATA     <3[,LTAB,dex]>
```

This command toggles a MNEMONIC mode switch which, when set, provides a special modification to the Frozen Data feature (sec. 10). If this mode is set, each Frozen Data Request ("0111sub1") causes the next successive element of a user-supplied table of data to be made available as the current Frozen Data element. LTAB is the label associated with the first word of the table, and "dex" is a number which serves as the index to the table for the first fetch. That index is incremented automatically, AFTER each fetch, and a zero value refers to the first table entry. If the previous index is to be retained, "dex" should be made a "-1". Both LTAB and "dex" must be omitted when toggling the switch TO THE OFF position. To the unwary user, this can be a source of programming error.

13.

```
        0111
        DATA     <4,pstbref,offset>
```

This command provides for loading the PSaddress bucket (see Sec. 1). Note that when Address Incrementation (Sec. 6) is enabled, "offset" should be ONE LESS than the first address used.

14.

```
        0111
        DATA     <5,SUBR>
```

This is a "cop-out" command which calls a MIXIT subroutine (SUBR) supplied by the user, in which he may do anything not specifically provided by MNEMONIC. The calling sequence will be:

```
        CALL     SUBR,AREA .
```

where AREA is the constants-and-data-storage area of Subroutine CODE, the MNEMONIC interpreter program. Its contents are:

```
EA:     0.        ;RANDOM KEY
        1.        ;CURRENT INDEX
        2.        ;INDEX AT WHICH ERR OCCURRED (0 IF OK)
        3.        ;GOOD DATA BUCKET
        4.        ;TEST DATA BUCKET
        5.        ;PSADDR BUCKET
        6.        ;TEST MASK
        7.        ;FROZEN DATA TABLE PTR (0=NOT TABLE LOAD)
        8.        ;FROZEN DATA TABLE INDEX
        9.        ;RANDOM KEY AT MARK
        10.       ;ERROR NUMBER THIS MARK
        11.       ;ERROR MSG. THIS MARK
        12.       ;LOAD TABLE INDEX THIS MARK
        13.       ;PSADDR THIS MARK
        14.       ;DATA THIS MARK
        15.       ;INDEX THIS MARK
        16.       ;COMMAND WORD THIS MARK
        17.&18.   ;CURRENT CONTROL WORD
        19.       ;USE RANDOM/FROZEN DATA (0=FROZEN)
        20.&21.   ;SAVE/RESTORE FF & BUCKET (0=SAVE)
        22.       ;PSA INCR. FF (0=NO INCR.)
        23.       ;OR-WORD FOR ERRORS
        24.       ;CW AT WHICH ERR OCCURRED
        25.       ;COMMAND TABLE BASE (=A(.1))
        26.       ;ADDRESS OF USER ERROR SUBROUTINE
        27.&28.   ;READS SINCE LAST MARK
        29.       ;CURRENT BITSHIFT FOR PARTIAL MEMORY FIELDS
        30.       ;CURRENT BITSHIFT AT LAST MARK
        31.       ;PSA BITSHIFT
        32.       ;PSA BITSHIFT AT LAST MARK
```

```
5.                0111
                  DATA      <6,DOIT>
```

This command causes the MAP to be placed in Maintenance Mode (if it is not already), and the Doit Register to be loaded with the contents of the six-word user-supplied table DOIT. The value "2" is then placed in the Maintenance Status Register (Address=177754), turning off Maintenance Mode but leaving the Map Halt bit on. The PROM/RAM address will have been incremented by one.

```
6.                0111
                  DATA      <7,DOIT>
```

This command is identical to the one above, except that a "3" instead of a "2" is loaded into the MSR, thus also clocking the MAP.

GENERAL NOTES:

A useful modification of the "0111sub3" command involves putting labels on specified points of the LTAB Load Table, and then initializing the index to that point by the arrangement:

```
                  0111
                  DATA      <3,LTAB>
                  DIFF      LTAB,LABn
```

```
        JMP     STRT
;
;
; PROGRAM:  QSD004.MIX
;
; AUTHOR:  STEPHEN N. MCALLISTER
;
; DATE WRITTEN:  6/4/76
;
; DESCRIPTION:  THIS MIXIT PROGRAM, WHICH MAKES USE OF THE MNEMONIC
;       INTERPRETER, TESTS THE DOIT AND DOIT ADDR REGISTERS, AS WELL
;       AS THE CONTROL STORE.
;
;
        HEAD    <MESSAGES & CONSTANTS>
MSGS:   DATA    2
        DATA    <MS0,10.>
        DATA    <MS1,29.>
        DATA    4
        DATA    <MS2,18.>
        DATA    <MS3,28.>
        DATA    <MS4,15.>
        DATA    <MS5,22.>
MS0:    CDATA   <QSD004.S01>
MS1:    CDATA   <PROM/RAM & DOIT REGISTER TEST>
MS2:    CDATA   <PH.1--DOIT ADDRESS>
MS3:    CDATA   <PH.2--ADDRESS INCREMENTATION>
MS4:    CDATA   <PH.3--DOIT BITS>
MS5:    CDATA   <PH.4--PROM/RAM CONTENT>
        THERE   <SMES,SOCT,DOPH,PHAZ,DPCH,INIT,CODE,CROM,LPSA>
DTBL:   DATA    <0,LST1,LST2,LST3,LST4> ;MNEMONIC LIST TABLE
PTBL:   DATA    <0,PH,PH,PH,PH> ;ONE FOR EACH PHASE LIST
PMAX:   DIFF    PTBL,PMAX
ERMX:   DATA    4
MS9:    DATA    8.
        CDATA   <ALL DONE>
STRT:   MOV     PMAX,DOPH
        DEC     DOPH
        MOV     ERMX,PHAZ
        CALL    INIT,<MSGS>
        CALL    DPCH,<PTBL>
        CALL    SMES,<MS9,<1,MS9>>
        STOP
        SUBR    PH,1
        MOV     <.1,DTBL>,DTBL
        CALL    CODE,<<DTBL,>>
        RTRN
        HEAD    <PHASE 1 LIST--MAP ADDRESS REGISTER TEST>
LST1:   DIFF    LST1,LND1       ;TABLE SIZE
        DATA    77427   ;0111 1111 0001 0111
        DATA    <2,MS1A,1>      ;MARK FOR DOIT ADDR TEST
        DATA    <177753,1>      ;MASTER RESET MAP
        DATA    377             ;GET MAP P/R ADDR MASK
        DATA    <3,TB1A,0>      ;SET UP LOAD TABLE
        DATA    71562   ;0111 0011 0111 0010
        DATA    <4,177756>      ;POINT MNEM. PSA AT P/R ADDR
L1A:    DATA    <5,0,0>         ;SET LOOP AT 5 PASSES
        DATA    1               ;GET NEXT BIT PATTERN
                                ;MARK--P/R ADDR.
        DATA    114140  ;1001 1000 0110 0000
                                ;WRITE IT (TB1A)
                                ;READ & TEST IT--23***
        DIFF    LST1,L1A        ;END OF LOOP
```

```
LND1:   DATA    LST1            ;END OF TABLE
MS1A:   DATA    -13.
        CDATA   <DOIT ADDR ERR>
TB1A:   DATA    <0,-1,125252,52525,123456>
        HEAD    <PHASE 2 LIST--PROM/RAM ADDRESS INCREMENTATION>
LST2:   DIFF    LST2,LND2       ;TABLE SIZE
        DATA    77577   ;0111 1111 0111 1111
        DATA    <3,TBL2,0>      ;MCR TEST TABLE SET
        DATA    <177753,1>      ;RESET THE SYSTEM
        DATA    <2,MS2A,2>      ;MARK FOR MCR & P/R ADDR TEST
        DATA    <177754,3406>   ;MCR ADDR = -1
        DATA    76467   ;0111 1101 0011 0111
        DATA    <4,177756>      ;SET PSA FOR P/R ADDR
        DATA    -1              ;P/R ADDR = -1
L2A:    DATA    <256.,0,0>      ;SET OUTER LOOP AT 256 PASSES
        DATA    <5,SUB2>        ;GO SET INITIAL TEST VALUES
        DATA    33427   ;0011 0111 0001 0111
L2B:    DATA    <8.,0,0>        ;SET INNER LOOP AT 8 PASSES
        DATA    <5,SB2B>        ;POKE THE B-BUS
        DATA    3400            ;SET MASK FOR MCR
        DATA    1               ;GET NEXT TABLE ENTRY
        DATA    120706  ;1010 0001 1100 0110
        DATA    177754          ;READ & TEST MCR--37***
        DATA    377             ;SET MASK FOR P/R ADDR
L2C:    DATA    0               ;READ & TEST P/R ADDR--41***
        DIFF    LST2,L2B        ;END OF INNER LOOP
        DATA    60000   ;0110 0000 0000 0000
        DIFF    LST2,L2A        ;END OF OUTER LOOP
LND2:   DATA    LST2            ;END OF TABLE
MS2A:   DATA    -17.
        CDATA   <PROM/RAM ADDR ERR>
TBL2:   DATA    <0,400,1000,1400,2000,2400,3000,3400>
BBUS:   DATA    177757
;
;       SUBROUTINE TO UPDATE R/P ADDRESS
;
        SUBR    SUB2,1          ;ENTRY POINT
        MOV     X255,L2C        ;MAKE THE TEST ADDR. --
        SUB     <1,L2A>,L2C     ; -- A FUNCTION OF LOOP COUNT
        CLR     <8.,,.1>        ;RESET THE TABLE INDEX
        RTRN                    ;RETURN TO MNEMONIC
;
;       SUBROUTINE TO BUMP THE POINTERS
;
        SUBR    SB2B,1          ;ENTRY POINT
        CALL    LPSA,BBUS       ;POKE THE B-BUS
        RTRN                    ;RETURN TO MNEMONIC
        HEAD    <PHASE 3 LIST--DOIT REGISTER TEST>
LST3:   DIFF    LST3,LND3       ;TABLE SIZE
        DATA    13767   ;0001 0111 1111 0111
        DATA    -1              ;MASK FOR FULL WORD
        DATA    <2,MS3A,-3,SUB3>        ;MARK FOR DOIT BIT TEST
        DATA    <177753,1>      ;RESET THE MAP
        DATA    <3,TB1A,0>      ;SET TO LOAD FROM START
        DATA    31367   ;0011 0010 1111 0111
L3A:    DATA    <5,0,0>         ;SET OUTER LOOP FOR 5 PASSES
                                ;MARK--DOIT BITS
        DATA    <177754,3406>   ;SET THE MDSEL TO -1
        DATA    <4,177757>      ;SET MNEM. PSA AT B-BUS
        DATA    71626   ;0111 0011 1001 0110
        DATA    1               ;GET NEXT VALUE
L3B:    DATA    <6,0,0> ;SET INNER LOOP AT 6 PASSES
                                ;WRITE THE VALUE (TB1A)
        DIFF    LST3,L3B        ;END OF INNER LOOP
```

```
L3C:    DATA    <6,0,0>         ;SET INNER LOOP AT 6 PASSES
                                ;READ & TEST IT--40***
        DIFF    LST3,L3C        ;END OF INNER LOOP
        DATA    60000           ;0110 0000 0000 0000
        DIFF    LST3,L3A        ;END OF OUTER LOOP
LND3:   DATA    LST3            ;END OF TABLE
MS3A:   DATA    -13.
        CDATA   <DOIT LOAD ERR>
;
;       SUBROUTINE TO OUTPUT MCR
;
        SUBR    SUB3,1          ;ENTRY POINT
        MOV     X5,TTTT         ;RECOMPUTE THE MCR
        SUB     <1,L3C>,TTTT
        CALL    SMES,<MS4C,<1,MS4C>>    ;OUTPUT IT
        CALL    SOCT,<X5,TTTT>
        RTRN                    ;RETURN FOR ERROR PROCESSING
        HEAD    <PHASE 4 LIST--PROM/RAM CONTENTS CHECK>
LST4:   DIFF    LST4,LND4       ;TABLE SIZE
        DATA    77567   ;0111 1111 0111 0111
        DATA    <3,CROM,0>      ;DATA TABLE LOAD
        DATA    <177753,1>      ;RESET THE MAP
        DATA    <2,MS4A,-4,SB4B>        ;MARK FOR P/R CONTENT CHECK
        DATA    <4,177757>      ;POINT MNEM. PSA AT B-BUS
        DATA    31177   ;0011 0010 0111 1111
L4A:    DATA    <256.,0,0>      ;SET OUTER LOOP AT 256 PASSES
                                ;MARK
        DATA    <5,SUB4>        ;GO SET THE P/R ADDR.
L4B:    DATA    <177756,0>      ;SET P/R ADDRESS (ADDR MODIFIED)
        DATA    170610  ;1111 0001 1000 1000
        DATA    <177754,3006>   ;SET MCR = -2
        DATA    0               ;MASK TO READ W/O CHECKING
                                ;DO IT
                                ;AND AGAIN
        DATA    11570   ;0001 0011 0111 1000
        DATA    -1              ;MASK TO TEST WHOLE WORD
L4C:    DATA    <6,0,0>         ;SET INNER LOOP AT 6 PASSES
        DATA    1               ;GET ROM DATA TABLE ENTRY
                                ;READ & TEST IT--37***
        DATA    63000   ;0110 0110 0000 0000
        DIFF    LST4,L4C        ;END OF INNER LOOP
        DIFF    LST4,L4A        ;END OF OUTER LOOP
LND4:   DATA    LST4            ;END OF TABLE
MS4A:   DATA    -17.
        CDATA   <PROM/RAM DATA ERR>
MS4B:   DATA    -13.
        CDATA   <    P/R ADDR=>
MS4C:   DATA    -8.
        CDATA   <    MCR=>
X255:   DATA    255.
X5:     DATA    5
;
;       SUBROUTINE TO SET UP ADDRESS POINTERS
;
        SUBR    SUB4,1          ;ENTRY POINT
        MOV     X255,<1,L4B>    ;SET UP THE POINTER
        SUB     <1,L4A>,<1,L4B>
        RTRN                    ;RETURN TO MNEMONIC
;
;       SUBROUTINE TO REPORT ERROR ADDRESS INFO
;
        SUBR    SB4B,1          ;ENTRY POINT
        CALL    SMES,<MS4B,<1,MS4B>>    ;OUTPUT THE P/R ADDR
        CALL    SOCT,<MS4B,<1,L4B>>
        MOV     X5,TTTT         ;COMPUTE THE MCR
        CALL    SOCT,<X5,TTTT>
        RTRN                        ;RETURN FOR ERROR PROCESSING
        FIN     STRT
>
```

```
.MAIN.  RT-11 MACRO VM02-10   16-JUN-77 00:02:27
TABLE OF CONTENTS
```

```
.MAIN.  RT-11 MACRO VM02-10   16-JUN-77 00:02:27 PAGE 1

                       ;---------------------------------------------------
                       ;        JMP      STRT                 ;FOR MANUAL STARTUPS
                       ;---------------------------------------------------
```

```
.MAIN.  RT-11 MACRO VM02-10   16-JUN-77 00:02:27 PAGE 2
MIXIT ASSEMBLY

                                 .SBTTL   MIXIT ASSEMBLY
 000000 000401                   BR       TTTT+2               ;FOR MAIN PROGRAMS
                       ;-------------------------------------------------
                       ;TTTT:    .DATA    0                    ;PROGRAM TEST WORD
                       ;-------------------------------------------------
 000002                TTTT:
 000002 000000                   .WORD    0
 000004 012700                   MOV      #STRT,%0
        002210'
 000010 000110                   JMP      (%0)
                       ;
                       ;
                       ; PROGRAM:  QSD002.MIX
                       ;
                       ; AUTHOR:  STEPHEN N. MCALLISTER
                       ;
                       ; DATE WRITTEN:  5/14/76
                       ;
                       ; 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>
                       ;--------------------------------
```

```
                                   .SBTTL  MESSAGE SECTION
                        ;---------------
                        ;MSGS:     DATA      2
                        ;---------------
   000012               MSGS:
   000012  000002                  .WORD     2
                        ;-------------------------
                        ;          DATA      <MSG1,10.>
                        ;-------------------------
   00014   000102'                 .WORD     MSG1
   00016   000012                  .WORD     10.
                        ;-------------------------
                        ;          DATA      <MSG2,33.>
                        ;-------------------------
   00020   000114'                 .WORD     MSG2
   00022   000041                  .WORD     33.
                        ;---------------
                        ;          DATA      11.
                        ;---------------
   00024   000013                  .WORD     11.
                        ;-------------------------
                        ;          DATA      <MS10,44;>
                        ;-------------------------
   00026   000156'                 .WORD     MS10
   00030   000054                  .WORD     44.
                        ;-------------------------
                        ;          DATA      <MS11,34.>
                        ;-------------------------
   00032   000232'                 .WORD     MS11
   00034   000042                  .WORD     34.
                        ;-------------------------
                        ;          DATA      <MS12,26.>
                        ;-------------------------
   00036   000274'                 .WORD     MS12
   00040   000032                  .WORD     26.
                        ;-------------------------
                        ;          DATA      <MS13,29.>
                        ;-------------------------
   00042   000326'                 .WORD     MS13
   00044   000035                  .WORD     29.
                        ;-------------------------
                        ;          DATA      <MS14,28.>
                        ;-------------------------
   00046   000364'                 .WORD     MS14
   00050   000034                  .WORD     28.
                        ;-------------------------
                        ;          DATA      <MS15,29.>
                        ;-------------------------
   00052   000420'                 .WORD     MS15
   00054   000035                  .WORD     29.
                        ;-------------------------
                        ;          DATA      <MS16,12.>
                        ;-------------------------
   00056   000456'                 .WORD     MS16
   00060   000014                  .WORD     12.
                        ;-------------------------
                        ;          DATA      <MS17,17.>
```

```
58                                ;-------------------------
59 00062 000472'                          .WORD    MS17
60 00064 000021                           .WORD    17.
61                                ;-------------------------
62                                ;         DATA     <MS18,11.>
63                                ;-------------------------
64 00066 000514'                          .WORD    MS18
65 00070 000013                           .WORD    11.
66                                ;-------------------------
67                                ;         DATA     <MS19,20.>
68                                ;-------------------------
69 00072 000530'                          .WORD    MS19
70 00074 000024                           .WORD    20.
71                                ;-------------------------
72                                ;         DATA     <MS20,22.>
73                                ;-------------------------
74 00076 000554'                          .WORD    MS20
75 00100 000026                           .WORD    22.
76                                ;----------------------------
77                                ;MSG1:    CDATA    <OSD002.S01>
78                                ;----------------------------
79 00102                          MSG1:
80 00102       121                        .ASCII   'OSD002.S01'
   00103       123
   00104       104
   00105       060
   00106       060
   00107       062
   00110       056
   00111       123
   00112       060
   00113       061
81 00114       000                        .BYTE    0
82          000114'                       .=.-1
83                                        .EVEN
84                                ;-------------------------------------------------------
85                                ;MSG2:    CDATA    <PICTURE SYSTEM MEMORY DIAGNOSTICS>
86                                ;-------------------------------------------------------
87 00114                          MSG2:
88 00114       120                        .ASCII   'PICTURE SYSTEM MEMORY DIAGNOSTICS'
   00115       111
   00116       103
   00117       124
   00120       125
   00121       122
   00122       105
   00123       040
   00124       123
   00125       131
   00126       123
   00127       124
   00130       105
   00131       115
   00132       040
   00133       115
   00134       105
   00135       115
```

```
                              .SBTTL  DISPATCHER
                        ;---------------------------------------------------------
                        ;     THERE   <INIT,SMES,SOCT,GETS,GETN,WRPS,RDPS,TIME,RNDM>
                        ;---------------------------------------------------------
                              .GLOBL  INIT
                              .GLOBL  SMES
                              .GLOBL  SOCT
                              .GLOBL  GETS
                              .GLOBL  GETN
                              .GLOBL  WRPS
                              .GLOBL  RDPS
                              .GLOBL  TIME
                              .GLOBL  RNDM
                        ;--------------------------------
                        ;     THERE   <DPCH,ERRL,PHAZ,DOPH>
                        ;--------------------------------
                              .GLOBL  DPCH
                              .GLOBL  ERRL
                              .GLOBL  PHAZ
                              .GLOBL  DOPH
                        ;---------------------
                        ;STRT: MOV    X7,PHAZ
                        ;---------------------
02210                   STRT:
02210  012700                 MOV     #X7,%0
       001254'
02214  012701                 MOV     #PHAZ,%1
       000000G
02220  011011                 MOV     (%0),(%1)
                        ;---------------------
                        ;     MOV     X7,DOPH
                        ;---------------------
02222  012700                 MOV     #X7,%0
       001254'
02226  012701                 MOV     #DOPH,%1
       000000G
02232  011011                 MOV     (%0),(%1)
                        ;-------------------------------------
                        ;     CALL    INIT,<MSGS>    ;INITIALIZE
                        ;-------------------------------------
02234  012702                 MOV     #INIT,%2
       000000G
02240  010546                 MOV     %5,-(%6)
02242  012701                 MOV     #..8190+2,%1
       002264'
02246  012700                 MOV     #MSGS,%0
       000012'
02252  010021                 MOV     %0,(%1)+
02254  012705                 MOV     #..8190,%5
       002262'
02260  004712                 JSR     %7,(%2)
02262                   ..8190:
02262  000401                 BR      ..8189
                              .BLKW   1
02266                   ..8189:
02266  012605                 MOV     (%6)+,%5
                        ;------------------------------------------------
```

```
                      ;ST1:   TST     MSIZ                 ;GET MEMORY SIZE?
                      ;-----------------------------------------------------
02270                 ST1:
02270 012700                  MOV     #MSIZ,%0
      002154'
02274 011067                  MOV     (%0),TTTT
      175502
                      ;-------------------------------------------------
                      ;       BNZ     ST3                  ;ALREADY GOT
                      ;-------------------------------------------------
02300 005767                  TST     TTTT
      175476
02304 001403                  BEQ     ..8188
02306 012700                  MOV     #ST3,%0
      002612'
02312 000110                  JMP     (%0)
02314                 ..8188:
                      ;-----------------------------------------------------------
                      ;       CALL    SMES,<MSGM,<1,MSGM>>      ;GET MEMORY SIZE
                      ;-----------------------------------------------------------
02314 012702                  MOV     #SMES,%2
      000000G
02320 010546                  MOV     %5,-(%6)
02322 012701                  MOV     #..8187+2,%1
      002360'
02326 012700                  MOV     #MSGM,%0
      001070'
02332 010021                  MOV     %0,(%1)+
02334 012700                  MOV     #1,%0
      000001
02340 006300                  ASL     %0
02342 062700                  ADD     #MSGM,%0
      001070'
02346 010021                  MOV     %0,(%1)+
02350 012705                  MOV     #..8187,%5
      002356'
02354 004712                  JSR     %7,(%2)
02356                 ..8187:
02356 000402                  BR      ..8186
                              .BLKW   2
02364                 ..8186:
02364 012605                  MOV     (%6)+,%5
                      ;-------------------------------------
                      ;       CALL    GETN,<X4,TTTT>
                      ;-------------------------------------
02366 012702                  MOV     #GETN,%2
      000000G
02372 010546                  MOV     %5,-(%6)
02374 012701                  MOV     #..8185+2,%1
      002424'
02400 012700                  MOV     #X4,%0
      001246'
02404 010021                  MOV     %0,(%1)+
02406 012700                  MOV     #TTTT,%0
      000002'
02412 010021                  MOV     %0,(%1)+
02414 012705                  MOV     #..8185,%5
```

```
       002422'
02420  004712               JSR     %7,(%2)
02422                ..8185:
02422  000402               BR      ..8184
                            .BLKW   2
02430                ..8184:
02430  012605               MOV     (%6)+,%5
                     ;---------------------------------------------------------
                     ;      BRZ     ST1                     ;MAKE SURE IT'S LEGAL
                     ;---------------------------------------------------------
 2432  005767               TST     TTTT
       175344
 2436  001003               BNE     ..8183
 2440  012700               MOV     #ST1,%0
       002270'
 2444  000110               JMP     (%0)
 2446                ..8183:
                     ;------------------
                     ;      BRN     ST1
                     ;------------------
 2446  005767               TST     TTTT
       175330
 2452  100003               BPL     ..8182
 2454  012700               MOV     #ST1,%0
       002270'
 2460  000110               JMP     (%0)
 2462                ..8182:
                     ;-------------------
                     ;      CLR     MSIZ
                     ;-------------------
 2462  012700               MOV     #MSIZ,%0
       002154'
 2466  005010               CLR     (%0)
                     ;-------------------
                     ;      DEC     MSIZ
                     ;-------------------
 2470  012700               MOV     #MSIZ,%0
       002154'
 2474  005310               DEC     (%0)
                     ;------------------------
                     ;ST2:  ADD     X16K,MSIZ
                     ;------------------------
 2476                ST2:
 2476  012700               MOV     #X16K,%0
       001272'
 2502  012701               MOV     #MSIZ,%1
       002154'
 2506  061011               ADD     (%0),(%1)
                     ;-------------------
                     ;      DEC     TTTT
                     ;-------------------
 2510  012700               MOV     #TTTT,%0
       000002'
 2514  005310               DEC     (%0)
                     ;------------------
                     ;      BNZ     ST2
                     ;------------------
```

```
2516 005767              TST     TTTT
     175260
2522 001403              BEQ     ..8181
2524 012700              MOV     #ST2,%0
     002476'
2530 000110              JMP     (%0)
2532              ..8181:
                  ;----------------------------------------
                  ;      CMPL    MSIZ,XHI           ;TOO HIGH?
                  ;----------------------------------------
2532 012700              MOV     #MSIZ,%0
     002154'
2536 012701              MOV     #XHI,%1
     001274'
2542 005067              CLR     TTTT
     175234
2546 011000              MOV     (%0),%0
2550 161100              SUB     (%1),%0
2552 006000              ROR     %0
2554 005567              ADC     TTTT
     175222
2560 050067              BIS     %0,TTTT
     175216
                  ;-----------------------------------
                  ;.      BRN     ST3                ;NO
                  ;-----------------------------------
2564 005767              TST     TTTT
     175212
2570 100003              BPL     ..8180
2572 012700              MOV     #ST3,%0
     002612'
2576 000110              JMP     (%0)
2600              ..8180:
                  ;--------------------------------------------
                  ;      MOV     XHI,MSIZ           ;YES, FIX IT UP
                  ;--------------------------------------------
2600 012700              MOV     #XHI,%0
     001274'
2604 012701              MOV     #MSIZ,%1
     002154'
2610 011011              MOV     (%0),(%1)
                  ;-------------------------------------------------
                  ;ST3:  CALL    DPCH,<JTBL>        ;CALL THE DISPATCHER
                  ;-------------------------------------------------
2612              ST3:
2612 012702              MOV     #DPCH,%2
     000000G
2616 010546              MOV     %5,-(%6)
2620 012701              MOV     #..8179+2,%1
     002642'
2624 012700              MOV     #JTBL,%0
     002156'
2630 010021              MOV     %0,(%1)+
2632 012705              MOV     #..8179,%5
     002640'
2636 004712              JSR     %7,(%2)
2640              ..8179:
```

```
2640 000401                   BR      ..8178
                              .BLKW   1
2644                  ..8178:
2644 012605                   MOV     (%6)+,%5
                      ;---------------------------------------------------
                      ;       CALL    SMES,<MS99,<1,MS99>>     ;SAY DONE
                      ;---------------------------------------------------
2646 012702                   MOV     #SMES,%2
     000000G
2652 010546                   MOV     %5,-(%6)
2654 012701                   MOV     #..8177+2,%1
     002712'
2660 012700                   MOV     #MS99,%0
     001206'
2664 010021                   MOV     %0,(%1)+
2666 012700                   MOV     #1,%0
     000001
2672 006300                   ASL     %0
2674 062700                   ADD     #MS99,%0
     001206'
2700 010021                   MOV     %0,(%1)+
2702 012705                   MOV     #..8177,%5
     002710'
2706 004712                   JSR     %7,(%2)
2710                  ..8177:
2710 000402                   BR      ..8176
                              .BLKW   2
2716                  ..8176:
2716 012605                   MOV     (%6)+,%5
                      ;------------------------------------------
                      ;       STOP                      ;QUIT...
                      ;------------------------------------------
                              .MCALL  .EXIT
2720                          .EXIT
                      ;----------------------------------------------
                      ;       HEAD    <ERROR PROCEDURE PROCESSOR>
                      ;----------------------------------------------
```

QSDDT OPERATION UPDATE

June 7, 1978

Evans & Sutherland Computer Corporation
580 Arapeen Drive
Salt Lake City, Utah 84108

TABLE OF CONTENTS

CHAPTER 6 QSDDT OPERATION

CHAPTER SIX

6.1 INTRODUCTION TO QSDDT

QSDDT is a general purpose PS-2 diagnostic tool written in the MIXIT programming language. QSDDT provides commands to examine and modify registers or memory locations in 12_{10} addressing modes as follows:

M: PS-2 Memory/SCB (Ø to 177777)
B: Host Computer Memory Buffer (Ø to 177)
I: Host Computer Interface Registers (Ø to 4)
D: Picture Processor DOIT Register (Subfields Ø to 5)
C: Picture Processor Control Store (Ø to 377)
P: Picture System Device Table (PSTB)
H: Host Computer Table (HSTB)
A: Character Memory (Ø-1777,2000-3777)
E: Character Generator Coefficient Memory (Ø-77, 200-377)
L: Line Generator/Character Generator State (Ø-37)
Q: Inter-character and Inter-line Spacing Registers (10-13)
F: Autorefresh Parameters (Ø-2)

In addition to commands which examine and modify locations, QSDDT supports other commands including the following: (a) reset the Picture System, (b) wait for DMA ready, (c) wait for Real-Time Clock, (d) save, unsave or verify the MAP DOIT Register, (e) specify octal or decimal input, (f) transfer data from one location to another, (g) execute a command line repeatedly, and (h) compare a received value with an expected value.

All command strings are terminated with a carriage return. Processing of a command string consists of an interpretation phase followed by an execution phase. Interpretation of a repeated command string is performed once only, allowing relatively high speed execution.

6.1.1 Introductory Examples

It is not necessary to know all commands in order to use QSDDT. A few of the simplest commands are also the most useful commands. Other commands may be learned as need arises. For reference, section 6.17 contains a summary of all QSDDT commands, with section numbers for more detailed information. The following examples are enough to make QSDDT useful in many cases:

R	Reset Picture System
MØ	Display contents of PS Memory location Ø
MØ=200,176000	Deposit 200 (octal) in PSMEM(Ø) and 176000 in PSMEM(1)
FØ=0,100,17;G	Set refresh start address =0(FØ) refresh´limit=100 (F1) clock rate=120(F2) and start autorefresh (G)

```
K                               Kill autorefresh


MØ:17                           Display PSMEM(Ø) through
                                PSMEM(17)


MØ
M=125252,52525,$1000            Load PSMEM(Ø) through
                                PSMEM(1777) with checker-
                                board pattern


MØ
?125252,?52525,$1000            Verify the checkerboard
                                pattern, and report the
                                first error


X                               Exit QSDDT
```

6.2 MEMORY COMMANDS

Following are the commands which open, examine, and modify registers (including memory locations) in the various addressing modes. Opening a location does not necessarily entail reading the location.

In the following discussion, "X" is an address mode symbol (M,B,I,D,C, etc), and "n" is a numeric value. "[n]" indicates a numeric specification which may be omitted.

The Memory Commands are as follows:

Xn Set addressing mode = X, and open location n. Example: MØ

=n Deposit n in the currently open location. Example: MØ=200

,[n] Open the location following the current location. If n is specified, deposit n in the newly opened location. Example: MØ=Ø,,1 modifies MØ and M2.

'[n] Open the location preceding the currently open location. If n is specified, deposit n. Example: M100=Ø'1 modifies M100, then M77.

. Display the contents of the currently open location. Example: M177760=1. modifies and then reads the same location back.

<CR> If the last opened location has not been modified, then display its contents. Example: MØ<CR>.

/ Read but do not display the currently open location. Example: CØ/ loads MAP Control Store Ø into the MAP DOIT register.

:n Display the contents of the block of locations beginning with the currently open location and ending with location n. Example: MØ:17.

>   Transfer the last read data (as modified by * or N command) to the next opened location. Example: MØ>1 copies MØ into M1.

n   According to the context established by the other commands in this section, n may be interpreted as the address of a location to be opened, or a value to be deposited. Either context may be forced by preceding n with = to deposit, or one of the address mode commands (M,B,I,D,C, etc.) to open. At the beginning of a line or following a semi-colon, n is interpreted as an address, in the current addressing mode. Example: MØ=2. Ø is an address, 2 is a value.

6.3 ADDRESSING MODES

The last issued address mode command defines the current domain of the memory commands which read, display, and modify registers or memory locations. Unless otherwise specified the addressable registers contain 16 bits. MAP Control Store memory locations consist of 96 bits which correspond to the six accessible 16 bit subfields of the DOIT register. The addressing modes are as follows:

6.3.1 (MØ:177777) PS-2 MEMORY/SYSTEM CONTROL BLOCK: In this mode all Picture System Memory locations, and registers of the PS-2 SCB, are accessible. The range of legal addresses is Ø through 177777, although many systems include less than the full memory configuration.

6.3.2 (BØ:177) HOST COMPUTER MEMORY BUFFER: The primary purposes of this mode are to facilitate DMA diagnosis, and to provide capability for refresh from a host computer memory buffer. When starting up, QSDDT reports the absolute address of the host buffer (DBUF = XXXXXX). Typically, the operator would deposit data in this buffer, then deposit the reported DBUF address in I3 (DMABA-PDP-11 only) in preparation for a DMA transfer. The reported DBUF address will be incorrect on a system using memory management.

Locations in the host buffer are addressed for purposes of examination and modification with a word index ranging from Ø to 177. To use a DMA start address other than at the beginning of the host buffer, the user must compute the absolute address within the buffer. In the case of the PDP-11, the absolute address is computed as DBUF + (2 x index).

6.3.3 (IØ:77) HOST COMPUTER INTERFACE REGISTERS: This mode addresses the registers on the host computer side of the PS-2 interface (PSBUS) as follows (PDP-11 only):

0 = PSDATA: Direct I/O Data Register,
default address = 767660
(195131 card)

1 = DIOPSA: Direct I/O Picture System Address,
default address = 767662
(195105 card)

2 = DMAWC: DMA word count (two's complement),
default address = 767664
(195131 card)

3 = DMABA: DMA Host Buffer Address,
default address = 767666
(195131 card)

4 = IOST: I/O Status Register,
default address = 767670
(195131 card)

For a non-standard UNIBUS configuration, the base address of this block of registers (default 767660) is modifiable as location H1 (see addressing mode "H" below).

Uses for "I" mode include diagnosis of the interface and general access to the I/O Status Register.

The range of legal addresses in I mode is Ø through 77, but the significance of each address may vary with different computers, and no address above 20 has been used on any computer at the present time. Documentation for "READ" and "WRITE" subroutines for each computer will detail the significance of each address in mode "I".

6.3.4 (DØ:5) PICTURE PROCESSOR DOIT REGISTER: The hardware DOIT register consists of eight virtual segments Ø through 7. The first 6 of these segments contain Picture Processor (MAP) control signals and represent the current state of the MAP. Segments 6 and 7 of the DOIT do not represent control data, but are accessed to write the contents of the DOIT into the Control Store, and load the Control Store output into the DOIT respectively. The DOIT register is implemented on the 195115-100 and -101 cards. Systems containing the Writable Control Store have 195124 cards in lieu of 195115's.

QSDDT recognizes only segments Ø through 5 as legal DOIT addresses. Segments 6 and 7 are implicitly utilized in Control Store Addressing Mode.

6.3.5 (CØ:377) PICTURE PROCESSOR CONTROL STORE: This addressing mode is somewhat unique due to the fact that addressable locations contain 96 bits. Opening and displaying a location causes the contents of that location to be loaded into the DOIT register, which is thereafter displayed in six segments. Attempts to deposit data into Control Store locations will have no effect, of course, if the system does not contain the Writable Control Store option. Otherwise, the current contents of the DOIT will be written into the open Control Store location.

Modifying the MAP Writable Control Store:

Commands to modify a location (= , ') are effective only if followed by a numeric specification. Therefore, in order to write the contents of the DOIT into Writable Control Store, a dummy numeric argument must be supplied with these commands. Example: C100=Ø causes the current contents of the DOIT (rather than the dummy value Ø) to be written into MAP Control Store location 100, provided the Picture System contains Writable Control Store.

6.3.6 (P2:64) PICTURE SYSTEM DEVICE TABLE (PSTB): This mode is used to establish non-standard addresses in the PS-2 System Control Block. The addresses reside in a software table called PSTB, the same table which is accessible by means of the "Modify" command in standard PS-2 Diagnostics (cf. Chapter 5).

6.3.7 (H1:13) HOST COMPUTER TABLE (HSTB): This mode provides access to HSTB, a table which defines the UNIBUS base address of the PS-2 device register (IOST, etc.) and the PS-2 Interrupt Vector Base. This is also a table which is accessible through the "Modify" command in standard PS-2 Diagnostics (cf. Chapter 5). Ordinarily, no modification of values in this table is necessary, unless multiple picture systems are interfaced to one processor.

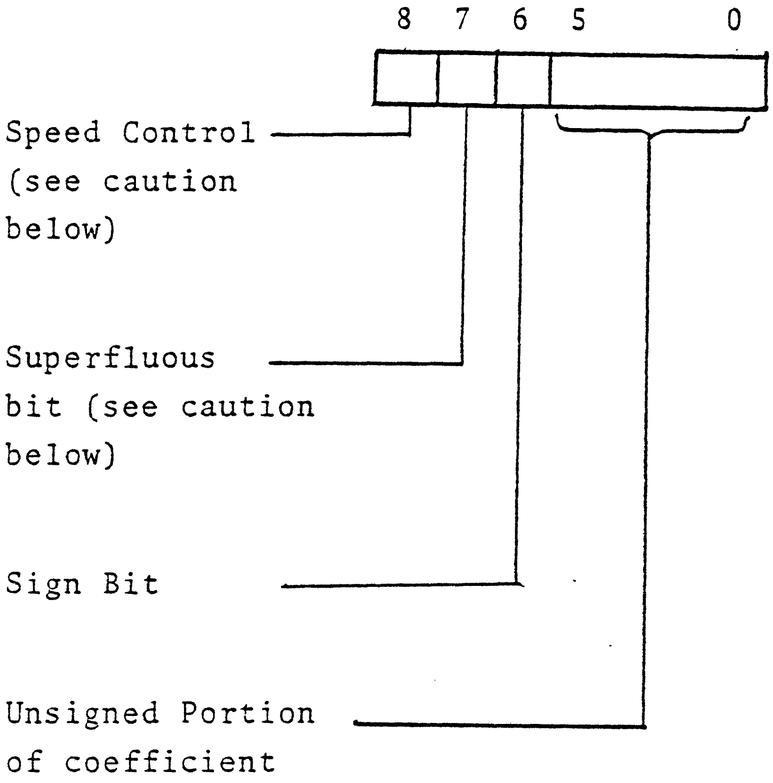
6.3.8 (AØ:3777) CHARACTER MEMORY: This mode provides read-only access to the Character ROM (Ø-1777) and read/write access to the Character RAM (2000-3777) located on the 195222 card. Data is treated as 12_{10} bits right-justified within a 16_{10} bit software word. For the formats of Character Generator commands, see Section 2.4.4.3 of the PS-2 Reference Manual.

Caution: It is possible to cause scope burns by modifying Character Generator commands.

6.3.9 (EØ:377) CHARACTER GENERATOR COEFFICIENT MEMORY: (Located on the 195221 card) according to PS-2 Reference Manual Sections 2.4.4.4, 2.4.4.5.b, and 2.4.5 "Reading the Coefficient Memories", the address of a coefficient memory element consists of a value in the range Ø-17 (read only) or 40-77 (read/write), followed by a suffix A,B,C, or D. QSDDT treats coefficient memory addresses as having a range of Ø-77 (read only) and 200-377 (read/write). In the QSDDT addressing scheme, the two least significant bits of the address designate A,B,C, or D. Correspondence to the Reference Manual addressing scheme is as follows:

Reference Manual	QSDDT	
ØA	0	First read-only location
ØB	1	
ØC	2	
ØD	3	
⋮	⋮	
17C	76	
17D	77	Last read-only location
40A	200	First read/write location
⋮	⋮	
77D	377	Last read/write location

The format for coefficient Memory data is as follows: (see Reference Manual 2.4.4.4).


8
7
6
5
0
Speed Control (see caution below)
Superfluous bit (see caution below)
Sign Bit
Unsigned Portion of coefficient

<u>Caution--Speed Control:</u> (a) Speed Control bits always read back as Ø, even though 1 way be written in. (b) The Speed Control bits of corresponding A/C or B/D elements should always be equal.

<u>Caution--Superfluous Sign Bit:</u> Bit 7 should always be equal to bit 6.

<u>Caution--Scope Burns:</u> It is possible to cause scope burns by modifying the Coefficient RAM.

6.3.10 (LØ:37) LINE GENERATOR/CHARACTER GENERATOR STATES: This data is read only. Addresses Ø-17 correspond to PGXBUS modes Ø-17. Addresses 20-37 correspond to PGYBUS modes Ø-17. See Section 2.4.5 of the PS-2 Reference Manual.

6.3.11 (Q10:13) INTER-CHARACTER AND INTER-LINE SPACING REGISTERS: This mode provides write only access in the range 10-13 as follows (see Reference Manual 2.4.4.5.c):

10: X 12-bit integer part
11: Y 12-bit integer part
12: X 12-bit fractional part
13: Y 12-bit fractional part

6.3.12 (FØ:2) AUTOREFRESH PARAMETERS: This address mode facilitates autorefresh control independent of the type of refresh controller (single-user or multi-user) available in the Picture System.

FØ: Refresh Start Address
F1: Refresh Limit
F2: Clock Rate (17=120 HZ, 16=60 HZ, etc.)

For further details, see Section 6.5, Autorefresh Control. Note that Autorefresh Parameters do not take effect until a "G" (Start Autorefresh) command is issued, even if autorefresh is already running. Also, in the case of a Multi-user Refresh Controller, autorefresh will not operate unless an "R" command

(PS Reset) is issued before the first "G" command.

NOTE: FØ and F1 should always be assigned even values. F1 should be set one greater than the last Picture System Memory location to be included in the refresh.

6.4 REPEATED EXECUTION ($ OR $N)

Both definite and indefinite repetition will terminate in the event of one of the following errors: (See section 6.15)

ADDRESS ERROR
READ ERROR
WRITE ERROR
COMPARISON ERROR

In the event of a SYNTAX ERROR or EXECUTION BUFFER FULL, no attempt will be made to execute the command string even once.

6.4.1 INDEFINITE REPETITION ($): The dollar-sign followed by no number or a negative number causes "non-terminating" execution of a command string (see exception above). At the end of a command string (e.g. M,=$) the string up to the $ is interpreted and executed. If a command string consists of "$" only, then the last preceding input command string will be re-executed, indefinitely.

6.4.2 DEFINITE REPETITION ($n): The dollar-sign followed by a positive number causes the command string up to the "$" to be re-executed the specified number of times. If a command string consists of $n (e.g. $20) only, then the last preceding input command string (up to but not including "$" character, if any) will be re-executed n times.

6.5 AUTOREFRESH CONTROL (G,K)

The "G" command causes autorefresh to be initiated according to autorefresh parameters FØ through F2 (see 6.3.12), and "K" causes autorefresh to halt. The "R" command (PS Reset) will also halt autorefresh.

Autorefresh parameters for a display buffer already in PS Memory may be determined as follows, prior to issuing a "G" command:

Refresh Start Address: With a Single-user Refresh Controller, examine M177735. With a Multi-user Refresh Controller, and a display buffer generated by a diagnostic, examine M37654. For such display buffers, the Refresh Start Address will normally be zero.

Refresh Limit: With a Single-user Refresh Controller, examine location M177736. With a Multi-user Refresh Controller, and a display buffer generated by a diagnostic, examine M37655. The contents of this location is two greater than the F1 parameter.

Clock Rate: The clock rate takes effect only upon occurrence of a "G" command. The same rate is used for both autorefresh and real-time clock, as follows:

F2 = 17	120 HZ
16	60 HZ
15	40 HZ
14	30 HZ
⋮	
F2 = Ø	previous rate is used

Special Considerations Concerning the Multi-user Refresh Controller: An "R" command (PS Reset) must be issued before the first "G" command in order for autorefresh to operate.

The "G" command causes the contents of the PS Memory location addressed by F1 to be saved, and a Refresh Controller halt command (40202) to be stored in that location. When F1 is changed, the next "G" command restores the previous limit location, saves the contents of the location now addressed by F1, and writes the halt command into the new limit location. FØ and F1 should always contain even values.

6.6 TEST FOR EXPECTED VALUE (?)

This command makes it possible to write simple memory tests in QSDDT (see section 6.1.1). A numeric value must always follow the question-mark, and represents the expected contents of the last opened location. If the expected value does not equal the received value, execution of a command-line repetition ($) is terminated, and a message is printed as follows:

```
COMPARISON ERROR; EXPECTED=NNNNNN RECEIVED XN=NNNNNN
```

6.7 INPUT RADIX CONTROL (O,Z)

By default, all QSDDT input is octal. Execution of the commands "Z" and "O" set the input radix, however, to decimal and octal respectively. The effect of these commands is not limited to one command string, but persists until the complementary command is executed.

NOTE: All QSDDT output is octal, irrespective of the input radix. Thus, QSDDT facilitates decimal-to-octal conversion, but not the converse.

6.8 NUMERIC SPECIFICATION (-,")

A number may be specified with 1 to 6 characters as follows:

The first character may be a digit, a minus sign, or a double-quote character. A minus sign specifies a two's complement value (e.g. -1 = 177777). A double-quote character specifies a sign - extended value (e.g. "37 = 177737).

The input radix is octal by default, or depends upon the last executed radix command (O for octal or Z for decimal).

6.9 SYNCHRONIZATION COMMANDS (T,W)

The "T" command causes QSDDT to wait for a real-time clock request. The "W" command causes QSDDT to wait for DMA READY. These commands may be useful during repeated exectution of a command string, or to confirm operation of the real-time clock.

6.10 DOIT COMMANDS (S,U,V)

The "S" command causes the current contents of the MAP DOIT register to be saved in a software block of 6 words. The "U" command restores the contents of the software block to the DOIT register. These commands may be useful when single-stepping the MAP or modifying the MAP Writable Control Store. The "V" command causes the current contents of the DOIT to be displayed as six values on a single line. It is equivalent to "BØ:5" except for the display format.

6.11 QSDDT TERMINATION (X)

This command causes QSDDT to terminate and return control to the operating system.

6.12 PICTURE SYSTEM RESET (R)

The "R" command causes a Picture System Reset to be issued.

6.13 NEGATE AND COMPLEMENT (N,*)

These commands cause the following actions: (a) The contents of the currently open location is read into a software location, and (b) the contents of the software location is converted to the complement in the case of "*" or the two's complement in the case of "N". Note that these commands do not modify the contents of the currently open location, although the currently open location can be modified by a command such as "MØ*>MØ".

6.14 END OF STATEMENT (;)

The semi-colon is used to separate command statements within one line of input.

6.15 QSDDT ERROR MESSAGES

SYNTAX ERROR: Following this message, the input command string will be retyped up to the point at which the syntax error was detected. Note: Any spaces in a QSDDT command string will result in a syntax error.

ADDRESS ERROR: Following this message, the illegal address will be printed. This message results from violation of the range of legal addresses in the current mode.

WRITE ERROR: Following this message, the offending address will be printed. This error results from the attempt to write into a read-only location.

READ ERROR: Following this message, the offending address will be printed. This error results from the attempt to read a write-only location (address mode "Q"). If the command string entails displaying the contents of the location, invalid display will occur following the error message.

EXECUTION BUFFER FULL: The execution buffer is loaded with command indices and numbers during interpretation of the input command string. It is possible, though abnormal, for the execution buffer to overflow, in which case a shorter command string must be typed.

6.16 EXAMPLES

6.16.1 General Examples: Following are some general examples of QSDDT commands. The underscored lines are the input command strings. The other lines are printed by QSDDT.

```
>RUN QSDDT
DBUF = xxxxx                    address of BØ
```

```
%

M"37                            examine RF Status Register
                                (Single-user Refresh Con-
                                troller)

M177737 = 100000

%

ZBØ=100.                        set decimal input radix,
                                deposit decimal 100 in
                                BØ, display BØ

BØ = 144

%

O                               restore octal input radix

%

BØ*>B1.                         deposit BØ complement in
                                B1 and display

B1 = 177633

%

BØ:2                            display BØ through B2

BØ = 144 177633 4757

%

=Ø.                             modify and display B2

B2 = Ø

%

PØ                              attempt to open PSTB[Ø]

ADDRESS ERROR PØ                illegal address

%

,,                              advance to PSTB[2]

P2 = 177744

%
```

6.16.2 Line Generator Test Pattern: The following sequence of commands will cause a test pattern to be displayed by DMA to the line generator.

```
>RUN QSDDT
DBUF = xxxxx
%
R                              reset the Picture System
"47                            examine DMAPSA
M177747 = 177777               DMA was directed to MAP
                               passive input port
%
="5.                           redirect to Picture Gen-
                               erator passive input port
M177747 = 177775
%
BØ=300,176000,133776,3776      load host buffer with
                               data to draw a big "T"
,170000,3776,170000,4002
%
,134002,3776,170000,3776
%
,170000,4002.                  display last location to
                               check word count
B15 = 4002
%
WI2=-16,xxxxx,1$10000          (PDP-11 only) this line
                               waits for DMA ready, loads
                               DMAWC with negative word
```

count, DMABA with the buffer address which was announced when QSDDT started up, sets "GO" in IOST, then goes back to wait for DMA ready and repeat 10000 times. A big "T" should now be visible on the scope

6.16.3 MAP Control Store Example: The following example examines and then modifies the contents of MAP Control Store location 4:

```
%
C4                                          read C4 into DOIT
C4 = 2367 57777 177777 177673 1777375 177777
%
DØ = 1,2,3,4,5,6/;V                         load the DOIT with new
                                            data and verify
DOIT = 1 2 3 4 5 6
%
C4 = Ø                                      dummy write operation
                                            causes DOIT to be written
                                            into C4
%
```

6.16.4 Dump MAP Internal Registers into PSMEM

```
*RU QSDDT
DBUF=XXXXX
```

```
%
R                              reset
%
M27=Ø
%
M"53=1=40,Ø                    reset MAP, then set MAO,
                               then MMSR=Ø
%
M"50=377,Ø                     MAOL, MAOA
%
M-1=12001=Ø                    MPIP:  RSR Store(377,Ø)
%
M27
M27=157                        MAP stack ptr should equal
                               117,137 or 157 etc.
%
MØ:3Ø                          this will display the
                               first 30 locations, for
                               example
MØ = XXX XXXX XX XXX
M4 = XX XXXXX XXX XX
etc.
%
```

6.16.5 Dump MAP Internal Registers (by DMA) into Host Computer Buffer.

```
*RU QSDDT
%
```

```
R                              reset

%

B27=Ø

%

M"53=0,"Ø                      MAOL=Ø,MAOA=17777Ø; Point
                               MAP output at DMA Passive
                               Input Port (DMAPIP)

12=-200,XXXXX                  (PDP-11 only) DMAWC (200
                               is size of DBUF); DMABA=
                               DBUF

%

I4=14=1                        (PDP-11 only) IOST:  DMAIN,
                               PASSIVE, then set GO

%

I4
I4=100014                      (PDP-11 only) DMA not
                               ready

%

M-1=12200=0                    RSR Store (200,0)

%

I4
I4=140214                      (PDP-11 only) DMA is now
                               ready

%

B27
B27=137                        MAP stack ptr should equal
                               117,137 or 157,etc.

%
```

6.16.6 Modification of Character RAM (Mode A)

```
*RU RSD009

RSD009.S02

CHARACTER GENERATOR VISUAL TEST

%

D

DO WHICH PHASE(S)?

2

%

X

RUNNING

PH 2:  E&S STANDARD CHARACTER SET, RAM

PHASE 2 DONE

TEST COMPLETE

*RU QSDDT

DBUF = xxxxx

%

A3764:3770                     Display the stroke defini-
                               tions for fast lower-case
                               "u"

A3764 = 2004 2414 2500 2404

A3770 = 1140

%

3764=Ø                         Still in "A" mode, change
                               the first move to a no-op,
                               observe the fast "u" drop
                               below its line on the
                               screen
```

```
%
,Ø
```

Change the next stroke to a no-op, observe the effect. Similarly, 3766 and 3767 may be changed. NOTE: 3770 is a halt command and should not be changed to a no-op.

For moves and draws, bits 7-4 = delta x and bits 3-Ø = delta y. Consult reference manual Section 2.4.4.3 for more detailed information.

6.16.7 Modification of Coefficient RAM (Mode E)

Execute QSDØ27 Phase 15 in similar fashion to the preceding example.

```
*RU QSDDT
DBUF = xxxxx
%
E200:203
E200 = 42 0 0 42
%
E200=40
```

Note the effect of this example on a line in the top-left quadrant

```
,40
```

Text blows way out of proportion because a speed bit in location 201, which always reads

```
                                      as Ø, got changed from
                                      1 to Ø
=440.                                 Restore the speed bit
E201 = 40
%
E202=1Ø
%
,420                                  This changes 203.  Speed
                                      bits of 203 and 201 are
                                      equal.  200-203 = 40A-D,
                                      which is the first matrix
                                      in the coefficient RAM.
E203 = 20
%
```

Other RAM locations in the range 200-377 affect other characters on the screen.

6.16.8 Modification of Spacing Registers (Mode Q)

Execute QSDØ27 phase 12 in order to load the Character RAM (A2000 +) with the "box" definition which adds displacement after drawing the box character.

```
*RU QSDDT
DBUF = xxxxx
%
M"35.,                                (or M37654.,) Read Refresh
                                      Start Address and Refresh
                                      Limit
```

R	Reset Picture System
MØ=300,176000,130000,170000	LG Reset, Move
%	
,37003,1006,0,0	(PDP-11 only) Load font: Ø is the box character. This is the end of the refresh buffer. For machines with first byte on the left side of each word: 1476,3002,0,0.
M207=Ø	
FØ=Ø,1Ø,17;G	Refresh Start Address, Refresh Limit, Clock Rate, Start Autorefresh. Four boxes should now appear on the screen.
%	
Q10=10	X integer part of spacing registers; note the screen
%	
=20	Increase delta X
%	
=200	
%	
,100	Y integer part
%	
=200	
%	

```
,400                                X fractional part has
                                    negligible effect, as
                                    does Y fractional part.

,200
```

6.16.9 Searching a Refresh Buffer

Suppose that a "glitch" appears on the screen and it is desirable to determine its exact location in the refresh buffer. This can be determined by doing "binary search" with the refresh limit. Run the phase of the diagnostic which produces the glitch (an autorefresh phase) and then run QSDDT.

```
%

R                                   Reset-necessary only for
                                    Multi-user Refresh Con-
                                    troller.

%

M177736

M 177736 = 1000                     Determine the refresh
                                    limit.  For our example,
                                    assume 1000.  For a Multi-
                                    user Refresh Controller,
                                    examine M37655, and sub-
                                    tract 2.

F1=400G                             Default values are FØ=0
                                    and F2=17 (120 HZ).  Cut
```

the refresh limit in half, and restart autorefresh. If the glitch disappears, try F1=600G, or if it is still in the buffer, F1-200G. Continue in this way to narrow down the location of the glitch.

6.16.10 DMA to Line Generator (NORD-10)

Reset, then initialize M"47 and BØ through B15 as in example 6.16.2.

```
WI5=16;I3=XXXXX,4004$10000
```

This line waits for DMA ready, loads DMA word count, loads DMA start address, and loads DMA control, setting "write" and "activate" bits. These actions are repeated 10000 times.

6.16.11 DMA to Line Generator (Interdata 8-32)

Before running QSDDT, display the System Memory Partitions ("D<space>M<CR>"). Convert the hexadecimal base address of the ".BG" partition to octal (for example, hex 10400 = octal 202000). Now run QSDDT, reset the Picture System, and initialize M"47 and BØ through B15 as in example 6.16.2. The DMA start address must be computed by adding the .BG base address to the reported DBUF address (for example, 202000+57710=261710). The least significant 16 bits are loaded into I10 (I12 for DMA end address), and the most significant 4 bits into I11 (I13 for DMA end address).

I11=1,61742,1

Initialize extended part of DMA start address, DMA end address, and extended DMA end address.

WI1Ø=61710;I6=0=1$10000

This line waits for DMA ready, loads DMA start address, clears DMA control, then sets "go" in DMA control. These actions are repeated 10000 times.

6.17 SUMMARY OF QSDDT COMMANDS

CHARACTER	MEANING	SECTION
$	Repeat command string indefinitely	6.4.2
$n	Repeat command string n times	6.4.2
'[n]	Open preceding location and deposit n	6.2
*	Read and complement	6.12
,[n]	Open succeeding location and deposit n	6.2
.	Read and display contents of currently open location	6.2
/	Read the currently open location	6.2
:n	Read and display from current location to location n	6.2
;	End of command statement	6.13
=[n]	Deposit n in the current location	6.2
>Xn	Deposit the current value in location n	6.2
?	Compare received and expected values	6.6
A	Character Memory Address Mode	6.3.8
B	Host Buffer Address Mode	6.3.2
X	Control Store Address Mode	6.3.5
D	DOIT Address Mode	6.3.4
E	Coefficient Memory Address Mode	6.3.9
F	Autorefresh Parameters	6.3.12
G	Start Autorefresh (GO)	6.5
H	Host Table (HSTB) Address Mode	6.3.7
I	Interface Register Address Mode PSDATA, DIOPSA, DMAWC, DMABA, IOST (Ø through 4)	6.3.3
K	Halt Autorefresh (KILL)	6.5
L	Line Generator/Character Generator State Mode	6.3.10
M	PS Memory/SCB Address Mode	6.3.1
N	Read and negate	6.12
O	Octal Input	6.5
P	Picture System Device Table Address Mode (PSTB)	6.3.6
Q	Inter-Character/Inter-Line Spacing Address Mode	6.3.11
R	Reset Picture System	6.11
S	Save DOIT	6.8
T	Wait for Real-Time Clock	6.7
U	Unsave DOIT	6.8
V	Verify DOIT	6.8
W	Wait for DMA ready	6.7
X	Exit QSDDT	6.9
Z	Decimal Input	6.5