ROYALE TROUBLESHOOTING MANUAL 3.Ø

P/N 11276

REV. B

JANUARY 1979

TABLE OF CONTENTS

SECTION		PAGE
1	INTRODUCTION	1
2	PROCEDURES INVOLVING THE FRONT PANEL	1
2.1	To read a File Register	1
2.2	To enter the monitor debugger	2
3	CHECKING I/O USING FIRMWARE	2
3.1	List of address stops	2
3.2	To check order codes to I/O devices	3
3.3	To check concurrent I/O address	3
3.4	To check disc status	4
3.5	To find which channel a terminal is on	4
3.6	Another way to find a terminal's channel	4
4	PROBLEMS AT LOGON	5
4.1	Assembler debugger at logon	5
4.2	Restoring base of the system dictionary	5
5	PROGRAM ERRORS	6
5.1	List of error numbers	7
5.2	Finding the program FID with an error	8
5.2.1	Example 1	8
5.2.2	Example 2	9
5.3	Determining the cause of an abort	9
6	PROCESS IDENTIFICATION BLOCK (PIB)	9
6.1	PIB table	10
6.2	Typical PIB status values	11
6.3	To look at a PIB using the debugger	12
7	COLDSTART	13
7.1	Core map after coldstart	13
7.2	What is loaded by the boot load process	14
8	DISC PROBLEMS	15
8.1	Disc status	16
8.2	To fix certain permanent disc errors	17
8.3	Key Locations	19
8.4	Frame Fault	19
8.5	Disc Interrupts	20
8.6	Finding Lost Disc Interrupts	20
8.7	Disc Controller Mod For Extended Core	20
9	TO CHECK SYSTEM CORESIZE	21
10	LINE PRINTER I/O	21
10.1	To watch data going to the line printer	21
10.2	Data in I/O buffer for line printer	22
11	SYSTEM HALTS AND LOOPS	22
11.1	System halts	22
11.2	Firmware loops	23
12	ADDITIONAL TROUBLESHOOTING TECHNIQUES	23

12.1 MPCB Errors 23
12.2 Tape Problems 23
12.3 Core Dump To Printer 24

13 PROPER MEMORY HARDWARE MODIFICATION CHECK 24

14 CONFIGURATION CONTROL 24

15 CONCURRENT INTERRUPTS FOR DEVICE ZERO 25

16 SYSTEM PERFORMANCE AND OPERATION TOOLS 25
16.1 DISCIO verb 25
16.2 BUFFERS verb 26
16.3 WHERE verb 27

17 APPENDIX A. ASCII CODE CHART 30

18 APPENDIX B. MICROCOMMAND REFERENCE 35

19 APPENDIX C. MONITOR DEBUGGER 37
19.1 Introduction 37
19.2 Memory command (M) 38
19.3 Go command (G) 38
19.4 Trace modes 38
19.5 Hexadecimal parameters 39

20 APPENDIX D. FIRMWARE LOCATIONS 40
20.1 Prom Chip Numbering System 40
20.2 Firmware Map 40

VERSION VIII

1 INTRODUCTION

This manual contains a variety of methods that can be used to determine problems in the ROYALE system. The ROYALE system contains many diagnostics already built into it that were not in 2.X systems. In no case should any system test or customer engineering personnel attempt to troubleshoot ROYALE hardware without exhausting all diagnostic aids (see Microdata P/N 11233 for a description of the diagnostic programs). Since the ROYALE system is not designed to aid troubleshooting it can be very hard to pinpoint any given problem using it. It will be up to the customer engineer or system test technician to make a logical guess as to what might be happening and then use some of the techniques in this manual to isolate the problem. Some hints will be given later in the manual on how to diagnose some of the harder problems. The order of troubleshooting a system is based on the problem and consequently this manual is not organized in any particular useful order. Therefore the reader must determine the sections most useful in solving his or her particular problem.

When a problem shows up it is important to record all the symptoms before attempting to solve it. For example, if a process aborts, write down what was printed on the terminal, the WHERE return stack (see section 16.3,) and what the user was doing. Did one terminal or several terminals abort? Were there any other unusual circumstances? If the system halts, write down all the front panel registers before attempting to restart it. This information may mean nothing to you, but if you need help in solving the problem the people you call will want to know it.

2 PROCEDURES INVOLVING THE FRONT PANEL

2.1 To read a File Register

1. Put the PANEL switch down.
2. Select the D display.
3. Put CXØØ in the console switches. X is the File Register number.
4. Read the value from the eight rightmost lights.
5. Place all front panel switches back up.

2.2 To enter the monitor debugger

1. Put sense switch 2 (only) down.
2. Press STEP. The system should halt. If it does not halt, press CLOCK, RESET, STEP.
3. Press RUN.
4. You will be in the monitor debugger on port zero.

3 CHECKING I/O USING FIRMWARE

There are various address stops that can be used to aid in the troubleshooting of a ROYALE problem(see Microdata Handbook on 1600 Computers for address stop description). This section includes first a list of address stops and some typical uses of them. The reader should be able to find other uses for address stops.

3.1 List of address stops

ADDRESS	MEANING	DATA DISPLAY
0047	External interrupt for other than 2614/2615 boards	Twice the device address of the interrupting device
0057	Data and status input from devices on a byte I/O basis	Byte being input
0062	Data and functions ouput to Devices on a byte I/O basis	Byte being output
0329	Single-step ROYALE machine instruction	Primary op code of the next instruction to be executed
1005	Program error trap	Twice the error code
0E4B	Transfer of control from monitor to virtual process	Low-order byte of PCB fid of process being activated
0E52	Same as 0E4B	High-order byte of PCB fid of process being activated
061A	External interrupt for 2614	Device address of device interrupting minus X'10'
0623	Same as 61A	Status byte of device interrupting
06A1	Concurrent I/O request	Twice the device address of the device requesting concurrent I/O
0633	Terminal input only	Byte to be put in Termio buffer from keyboard input
1056	First block of coldstart read into core	
102F	Powerfail restart executing	

3.2 To check order codes to I/O devices

The address stop at ØØ62 can be used to verify that the proper order codes are being sent to the proper devices. When this address stop is used, the data display contains the byte being output and the T register contains the order code and device address. To display the T register when the computer halts put the panel switch down and set console command switches to BØ2Ø. To continue put the panel switch up and the console switches to their original setting.

A typical sequence to the disc might be as follows: (Note that in order to see this sequence, it might be necessary to 'push through' about 5Ø address stops at ØØ62 to get past the terminal I/O that is done before the disc I/O.)

T	D	MEANING
14	Ø4	Queue drive zero on device X'14'
94	38	Beginning memory address (upper byte)
D4	39	Ending memory address (upper byte)
54	Ø2	Disc address (upper byte)
74	CØ	Disc address (lower byte)
F4	FF	Ending memory address (lower byte)*
B4	ØØ	Beginning memory address (lower byte)
34	ØØ	Action
14	9Ø	Start queued seeks and arm interrupt

* Lower bit of ending memory address is used to select upper or lower core bank (1=lower bank, Ø=upper bank). All ending address's are forced by hardware to be odd.

3.3 To check concurrent I/O address

Put an address stop at Ø6A1. The CPU will halt if a concurrent I/O request occurs. Twice the device address of the requesting controller will appear in the eight low-order display digits if the 'D' button is depressed. File register 13 contains twice the device address at this point. The high-order bit of file register 13 indicates the direction of the I/O(Ø=input, 1=output).

For example

D	FILE 13	MEANING
12	12	Reading mag tape (standard device address=9)
12	92	Writing mag tape (standard device address=9)
ØA	8A	Printing (standard device address=5)
Ø8	Ø8	Reading cards (standard device address=4)

Except in the case of the line printer, the address stop will cause block transfer operations to malfunction. Specifically

with the mag tape, the tape motion will automatically stop and no data will be transferred. Indication to the software of this failure is unpredictable.

3.4 To check disc status

1. Ensure no other I/O is in progress (i.e., mag tape, line printer, communication boards, etc.)
2. Depress data display button on front panel
3. Set an address stop at Ø047 (CPU will halt if disc I/O is in progress)
4. Mentally shift the contents of the 8 low-order display lamps right one bit to get the device address of the disc controller.
5. Set an address stop at Ø057
6. Depress RUN once
7. Read the major status of the disc controller from the 8 lower-order display lamps
8. Depress RUN once
9. If the error bit was on in step 7, the minor status is now displayed
10. Reset all command switches
11. Depress RUN to resume operations

3.5 To find which channel a terminal is on

1. Set a address stop at 61A
2. Depress the data display button
3. Depress a key on the terminal (the CPU should halt)
4. Add X'1Ø' to the eight-bit display to get the device address of the 2614 and 2615 board
5. Set the address stop at Ø623
6. Depress RUN once (the CPU should halt again).
7. The eight bit display is the status from the 2614/15 board;
8. Reset all command switches
9. Depress RUN once to resume processing

3.6 Another way to find a terminal's channel

1. Set address stop at 635
2. Depress the M display button
3. Depress a key on the terminal (the CPU should halt).
4. The upper byte is the device address and the 3 high-order bits of the lower byte are the channel number
5. Reset all command switches
6. Depress RUN once to resume processing

4 PROBLEMS AT LOGON

4.1 Assembler debugger at logon

The following procedure is for when you are at logon and want to get into the assembler debugger because you are unable to logon. Note that this procedure should only be used when the system is not being used by anyone else.

1. Depress carriage return on terminal Ø until LOGON PLEASE: message is repeated.
2. Depress sense switch 2 and press STEP and then RUN to enter monitor-debugger.
3. Read core location 8Ø2. Mentally add one to it. If the result of the addition is hex '2Ø', change the result of the addition to hex 'ØØ'.
4. Add the result of step 3 to hex '18ØØ' and read the location indicated.
5. Put a breakpoint on change of byte at the calculated location. (M14=ØØ, M15=18, M16=result of step 3, M17=byte read in step 4).
6. Press linefeed.
7. Press carriage return. The system should break to the monitor debugger.
8. Look at location 8ØC. The first seven bits indicate upper byte location of PCB and lowest bit indicates which bank. Reconstruct byte to indicate start of PCB and add 8Ø. For example if 8ØC contains a 7F then the start of the PCB is 1ØØØØ +7EØØ = 17EØØ and adding 8Ø results in an address of 17E8Ø.
9. Look at the above calculated address with the monitor-debugger and it should contain a one. Change the one to a zero.
10. Put a breakpoint at change of that byte from X'ØØ'. (M14=Ø8 or 88 if upper core, M15=contents of 8ØC with low bit zeroed, M16=8Ø, M17=ØØ).
11. Press linefeed.
12. When it breaks change that location from 'Ø1' back to 'ØØ' and change location 14 back to 'ØØ'.
13. Press linefeed, put sense switch 2 up, and hit the break key and you should be in the assembler debugger.
14. If you want to single step you have to change location .8E;2 to a .ØØØ5 (location of user in PCB)

4.2 Restoring base of the system dictionary

When you are unable to log on, the base of the system dictionary may have been lost. If you can get into the assembly language debugger and know where the system dictionary should be, you can often restore it. The correct base, modulo, and separation for the system dictionary may be found by looking on the 'SYSTEM' page of a file-stat report. CR, below, means carriage return.

1. Get into the assembly language debugger.
2. Type I127.5Ø;4CR. The current value of the base will be displayed, followed by an equal sign (=). Replace it with the correct value by typing the correct value and CR.
3. Type I127.54;2CR. The current value of the modulo will be printed, followed by an equal sign. Replace it with the correct value and type linefeed (not CR). The current value of the separation will be printed on the same line. Correct it and type CR.
4. Type END and CR. You should now be able to log on.

IMPORTANT

When the base of the system dictionary is lost, it is very likely that the system overflow table was clobbered, too. You should check it out with the POVF verb and do a file-save and restore immediately, if necessary.

5 PROGRAM ERRORS

The following method is to determine if a program error is occurring. Errors automatically trap to the assembler debugger through address 1ØØ5. Some so called errors are normal for the system such as break key (Error A) and certain foward link zeros (Error 5). Knowledge of the ROYALE assembly code might be necessary to determine the cause of an error.

1. Set an address stop at 1ØØ5 (CPU will halt on program error)
2. Record the contents of file registers 1,2,3,4 and 7. They will contain the following information:

FILE	CONTENTS
1	Upper byte of PCB's core memory address*
2	Upper byte of program's core memory address(PCU)*
3	Lower byte of program's core memory address(PCL)
4	High order byte of current instruction
7	Error code (see below)

*Note that the memory bank is unknown at this point.

3. Display L register and select panel mode.
4. Place B1ØA on switches and press CLOCK switch and note most significant bit of L register for which memory bank PCB is located (Ø=lower,1=upper)
5. Place B2ØA on switches and press CLOCK switch and note most significant bit of L register for which memory bank PCU is located (Ø=lower,1=upper)

5.1 List of error numbers

ERROR NO.	MESSAGE	DESCRIPTION
Ø	Illegal opcode	An illegal (undefined) operation has been found
1	Rtn stack empty	A Rtn (return) instruction was executed when the return-stack was empty (current pointer was at X'Ø184')
2	Rtn stack full	A BSL or BSLI (subroutine calls) instruction was executed when the return-stack was full (current pointer was at X'Ø1BØ'); the return-stack has been reset to an "empty" condition before the trap.
3	Referencing frame zero	An address register has an FID of zero
4	Crossing frame limit	An address register in the "unlinked" format has been incremented or decremented off the boundary of a frame, or has been used in a relative address computation that causes the generated relative address to cross a frame boundary
5	Forward link zero	An address register in the "linked" format has been incremented past the last frame in the linked frame set
6	Backward link zero	An address register in the "linked" format has been decremented prior to the first frame in the linked frame set
7	Privileged opcode	A privileged operation code (one executable only in the monitor mode of operation), has been found while executing in the virtual mode
8	Referencing illegal frame	An address register has an fid that exceeds the maximum value allowable in the current disc configuration
9	&	A disc error has occurred. The operation will be retrieved later.
A	Unusual comm status	With the 2614/15 communication controller this is equivalent to a break key. (Framing error)
B	Rtn stack format err	The return-stack pointers are in an illegal format.
C	Terminal overrun	This is a parity or terminal overrun and is presently ignored by the software
D	Mess processor	Some process is sending a

	request	message to another. This is not usually an error.
E	End process	Terminate the primary process and execute a go. Usually not an error.
F	Debug trace mode	When a trace mode is set, the firmware causes traps to the debugger using this error number.
1Ø	Divide overflow	A divide by zero has occurred. The debugger sets the overflow bit in users PCB.
11	Referencing illegal device	An attempt to access a device not authorized by the configuration control chip.
12	Unnorm. s/r	Storage register not normalized in branch register instruction.

5.2 Finding the program FID with an error

To determine the FID of the program currently executing use the following method:

1. Mark down contents of PCU(file register 2) and which core bank it is in. (During 1ØØ5 address stop)
2. Enter monitor-debugger by putting sense switch 2 down and pressing step then run.
3. Mentally shift PCU (file register 2) or the starting address of any core buffer whose FID you want to know, to the right one bit.
4. Mentally add 28Ø to the result of 3.
5. Mentally add 1ØØØØ to the result of 4 if upper bank of core.
6. Read the contents of that memory address using the monitor-debugger.
7. Mentally OR 7ØØ(low core) or 1Ø7ØØ (high core) to the PCU or other core buffer starting address.
8. Read contents of that address and the one after it.
9. Using the results of 8 as the 2 high order bytes and 6 and the low order byte, determine the FID of the buffer.
10. If you are looking up the FID of the program currently executing (file register 2), then look the program up in the table of programs in your Reality Reference Manual.

The above method may be used to find the FID of any core buffer. You simply use the first byte of the core address instead of the PCU.

5.2.1 Example 1

If file register 2 has a 1D and it is in upper core.

A) Shift 1D to the right one bit = ØE.
B) Add 1Ø28Ø = 1Ø28E.
C) Read contents of 1Ø28E.

D) OR 1Ø7ØØ to 1D =1Ø71D.
E) Read contents of 1Ø71D and 1Ø71E.
F) If the contents of 1Ø71D = Ø and
   1Ø28E = 22, then the FID
   would be hex 22 (decimal 34) and the program
   executing would be OF1.

5.2.2 Example 2

If you want the FID of the core buffer starting at FEØØ.

A) Shift FE to the right one bit = 7F.
B) Since it is lower core add 28Ø =2FF.
C) Read contents of 2FF.
D) Since it is lower core or 7ØØ=7FE.
E) Read contents of 7FE and 7FF.
F) If the contents of 7FE=1 7FF=69 and
   2FF=5Ø, then the FID
   associated with that buffer
   would be 1695Ø in hex.

5.3 Determining the cause of an abort

You can find the cause of an abort to the assembly debugger even if you are on another terminal than the one which aborted. Using the 'WHERE' verb you can see which lines are in the debugger by looking at the first location in the return stack and see if it is in frame 21(typically 21.Ø34). If the process is in the debugger, look at the number right after the line or port number which will be the PCB of that line in hexadecimal. The first byte (byte zero) of the PCB will give you the error number which corresponds to the table of errors listed above. For example if the process was line 2 and the PCB was indicated to be Ø24Ø you would type in .24Ø.Ø;1. This would give you the first byte of the PCB. If the first byte was .ØA then the reason for entering the debugger would be a break key or framing error on that line's comm board.

6 PROCESS IDENTIFICATION BLOCK (PIB)

The ROYALE CPU is designed as an interactive system capable of communicating with several users simultaneously. A user communicates with the system via a communication terminal such as a Teletype or a CRT terminal. Associated with each terminal is a process. A process is not an element of the system but rather a continuing operation on a set of functional elements. Refer to ROYALE Reference Manual for peripheral I/O details. For each process attached to the system, there is a Process Identification Block (PIB). Each PIB is 32 bytes long. The PIB for terminal zero is in main storage locations X'8ØØ'-X'81F'; locations X'82Ø' through X'83F' contain the PIB for terminal one, and so forth. Also associated with each PIB is a termio buffer of 32

bytes. The termio buffer is located 1000 hex bytes above the PIB so that the termio buffer for terminal zero is from X'1800' to X'181F'. The PIB contains information about the status of the process with which it is associated. The following is a description of the PIB contents.

6.1 PIB table

BYTE	BIT	NAME	MEANING
0	0	Active	Set when process may be activated.
	1	Sleep	Zeroed to sleep till time in PIBFID.
	2	Dioblk/	Zeroed by firmware on a frame fault.
	3	Not used/	Used to be PIBEND.
	4	Not used/	Used to be delay bit.
	5	Obyteblk/	Zero during terminal output.
	6	Ibyteblk/	Zero during terminal input.
	7	Not used/	Used to be cioblk.
1	0		Set to echo input.
	1	Indebug	Set when executing from dcb.
	2		Set by firmware on program error trap.
	3		Bank number.
	4-7		Error number.
2	0-7		Byte address of last character in Terminal I/O buffer
3	0-7		Number of bytes in terminal I/O buffer minus one.
4	0-7		Link to next PIB.
5	0-7		Link to previous PIB.
6	0-7		Disk error byte (major status).
7	0-7	PIB--dct	Pointer to device control table entry.
	0-3		Always zero.
	4-5		Last two bits of device address.
	6-7		Unit number.
8-9	0-15	Cylinder	Disc cylinder address.
8	0	Platter	Same value as head-bit(1)
	1-7		First seven bits of cylinder.
9	0-1		Last two bits of cylinder.
	2		Head bit 0.
	3-7		All five bits of the sector address
A	0-7	Head-bits(4-1)	

	Ø-3		Actual head bits 4-1
	4-5		Always zero.
	6-7		Action code. (Dscwrt and dscvfy)
B			Disk error byte (minor status)
C	Ø-7	Pibbuf	Buffer number.
C-F	Ø-31	Pibfid	Fid which caused frame fault. (After zeroing high byte)
1Ø	Ø-7		Terminal I/O bubble up counter
	Ø-3		Initial count
	4-7		Counter
11	Ø-7		Frame fault bubble down counter
	Ø-3		Initial count
	4-7		Counter
12	Ø-7		Real time clock bubble down counter
	Ø-3		Initial count
	4-7		Counter
13	Ø-7		Termio input activate count
14-17	Ø-31		Deactivation counter
18	Ø-7		Lock number
19			Time slice
1A-1B	Ø-15		Disc unit queue link (not used at present)
1C-1F	Ø-31	Readctr	Disc read counter

6.2 Typical PIB status values

The following table is a PIB status list. Some clues to why a system is failing can be found from this list. If a process continually disc roadblocked there might be a problem with the disc. If a process is continually terminal output roadblocked there might be a problem with the 2614 or 2615. More information than the PIB status will be needed to prove these assumptions.

BYTE Ø	BYTE 1	MEANING
5F	ØØ	Frame fault, waiting for frame to be brought in core from disc.
3F	ØØ	Asleep until time in PIBFID or break key.
7D	ØØ	Terminal I/O input roadblocked
7B	ØØ	Terminal I/O output roadblocked
FB	ØØ	Can be activated or is active for purpose of transferring more characters to the terminal I/O buffer.

7F	00	Can be activated or is active.
FD	00	Can be activated or is active for the purpose of removing characters from the terminal I/O buffer.
7D	40	Terminal I/O input roadblocked in debugger
7B	40	Terminal I/O output roadblocked in debugger
7F	40	Can be activated or is active in debugger.

6.3 To look at a PIB using the debugger

When in the assembler debugger the following commands may be used to look at any PIB.

KEYIN	LINE NO.	CHANNEL	DEVICE ADDRESS
X.FFFFFC.00;32	0	0	18
X.FFFFFC.20;32	1	1	18
X.FFFFFC.40;32	2	2	18
X.FFFFFC.60;32	3	3	18
X.FFFFFC.80;32	4	4	18
X.FFFFFC.A0;32	5	5	18
X.FFFFFC.C0;32	6	6	18
X.FFFFFC.E0;32	7	7	18
X.FFFFFC.100;32	8	0	19
X.FFFFFC.120;32	9	1	19
X.FFFFFC.140;32	10	2	19
X.FFFFFC.160;32	11	3	19
X.FFFFFC.180;32	12	4	19
X.FFFFFC.1A0;32	13	5	19
X.FFFFFC.1C0;32	14	6	19
X.FFFFFC.1E0;32	15	7	19
X.FFFFFB.00;32	16	0	1A
X.FFFFFB.20;32	17	1	1A
X.FFFFFB.40;32	18	2	1A
X.FFFFFB.60;32	19	3	1A
X.FFFFFB.80;32	20	4	1A
X.FFFFFB.A0;32	21	5	1A
X.FFFFFB.C0;32	22	6	1A
X.FFFFFB.E0;32	23	7	1A
X.FFFFFB.100;32	24	0	1B
X.FFFFFB.120;32	25	1	1B
X.FFFFFB.140;32	26	2	1B
X.FFFFFB.160;32	27	3	1B
X.FFFFFB.180;32	28	4	1B
X.FFFFFB.1A0;32	29	5	1B
X.FFFFFB.1C0;32	30	6	1B
X.FFFFFB.1E0;32	31	7	1B

The leading X and trailing ;32 are not needed after they have been keyed in once. To look up Termio Buffers change the leading F to an E. For example, line 5's Termio Buffer can be displayed by typing, "X.EFFFFC.A0;32CR".

7 COLDSTART

7.1 Core map after coldstart

This table describes the core map of the system as it is initialized by the cold-start process. A minimum of 16K of core is required. If there is extended core, certain buffers above 64K will be initialized as indicated. Status=80 means that the program is core locked at cold start. Status=FF indicates the program will be read in and out of core as needed.

CORE ADDRESS	FID	STATUS	PROGRAM NAME	COMMENTS
0000-01FF	NONE	80	MPCB	MONITORS PCB
0200-03FF	NONE	80	MSCB	STATUS AND FID TABLES
0400-05FF	NONE	80	MONITOR	
0600-07FF	NONE	80	MTCB	FID TABLES
0800-09FF	FFFFFC	80	PIB0	PROCESS'S 0-15
0A00-0BFF	FFFFFB	VARIES	PIB1	PROCESS'S 16-31 *
0C00-0DFF	NA	NA	NOT ASSIGNED	
0E00-0FFF	NA	NA	NOT ASSIGNED	
1000-11FF	NONE	80	MQCB	HASH TABLES
1200-13FF	VARIABLE	FF	PCB0	LINE 0 PCB
1400-15FF	VARIABLE	FF	DCB0	LINE 0 SCB
1600-17FF	47	FF	ABSL1	ABS LOADER
1800 19FF	EFFFFC	80	TERMIO 1	TERMINAL I/O BUFFER LINES 0-15
1A00-1BFF	EFFFFB	VARIES	TERMIO 2	TERMINAL I/O BUFFER LINES 16-31 *
1C00-1DFF	NONE	80	MONITORY	DISC I/O ROUTINES
1E00-1FFF	NONE	80	MONITORX	MONITOR ROUTINES
2000-21FF	NONE	80	MONITORZ	MONITOR ROUTINES
2200-23FF	1	FF	DB1	DEBUGGER CODE
2400-25FF	17	FF	DB2	DEBUGGER CODE
2600-27FF	18	FF	DB3	DEBUGGER CODE
2800-29FF	19	FF	DB4	DEBUGGER CODE
2A00-2BFF	20	FF	DB5	DEBUGGER CODE
2C00-2DFF	21	FF	DB6	DEBUGGER CODE
2E00-2FFF	161	FF	DB7	DEBUGGER CODE
3000-31FF	210	FF	ABSL2	ABS LOADER
3200-33FF	6	FF	TERMIO	TERMINAL INPUT OUTPUT
3400-35FF	4	FF	TCL-INIT	PROCESS INITIALIZATION
3600-37FF	7	FF	DISKFIO-I	FILE ROUTINES
3800-39FF	35	FF	TAPEIO-II	TAPE ROUTINES
3A00-3BFF	36	FF	TAPEIO-II	TAPE ROUTINES
3C00-3DFF	285	FF	TAPEIO-IV	TAPE ROUTINES
3E00-3FFF	8	FF	SYSTEM-SUBS-I	SYSTEM ROUTINES
10200-103FF	NONE	80	MSCB1	UPPER CORE MSCB
11000-111FF	NONE	80	MQCB	UPPER CORE MQCB

Note * Assigned only if needed.

7 COLDSTART

7.1 Core map after coldstart

This table describes the core map of the system as it is initialized by the cold-start process. A minimum of 16K of core is required. If there is extended core, certain buffers above 64K will be initialized as indicated. Status=80 means that the program is core locked at cold start. Status=FF indicates the program will be read in and out of core as needed.

CORE ADDRESS	FID	STATUS	PROGRAM NAME	COMMENTS
0000-01FF	NONE	80	MPCB	MONITORS PCB
0200-03FF	NONE	80	MSCB	STATUS AND FID TABLES
0400-05FF	NONE	80	MONITOR	
0600-07FF	NONE	80	MTCB	FID TABLES
0800-09FF	FFFFFC	80	PIB0	PROCESS'S 0-15
0A00-0BFF	FFFFFB	VARIES	PIB1	PROCESS'S 16-31 *
0C00-0DFF	NA	NA	NOT ASSIGNED	
0E00-0FFF	NA	NA	NOT ASSIGNED	
1000-11FF	NONE	80	MQCB	HASH TABLES
1200-13FF	VARIABLE	FF	PCB0	LINE 0 PCB
1400-15FF	VARIABLE	FF	DCB0	LINE 0 SCB
1600-17FF	47	FF	ABSL1	ABS LOADER
1800 19FF	EFFFFC	80	TERMIO 1	TERMINAL I/O BUFFER LINES 0-15
1A00-1BFF	EFFFFB	VARIES	TERMIO 2	TERMINAL I/O BUFFER LINES 16-31 *
1C00-1DFF	NONE	80	MONITORY	DISC I/O ROUTINES
1E00-1FFF	NONE	80	MONITORX	MONITOR ROUTINES
2000-21FF	NONE	80	MONITORZ	MONITOR ROUTINES
2200-23FF	1	FF	DB1	DEBUGGER CODE
2400-25FF	17	FF	DB2	DEBUGGER CODE
2600-27FF	18	FF	DB3	DEBUGGER CODE
2800-29FF	19	FF	DB4	DEBUGGER CODE
2A00-2BFF	20	FF	DB5	DEBUGGER CODE
2C00-2DFF	21	FF	DB6	DEBUGGER CODE
2E00-2FFF	161	FF	DB7	DEBUGGER CODE
3000-31FF	210	FF	ABSL2	ABS LOADER
3200-33FF	6	FF	TERMIO	TERMINAL INPUT OUTPUT
3400-35FF	4	FF	TCL-INIT	PROCESS INITIALIZATION
3600-37FF	7	FF	DISKFIO-I	FILE ROUTINES
3800-39FF	35	FF	TAPEIO-II	TAPE ROUTINES
3A00-3BFF	36	FF	TAPEIO-II	TAPE ROUTINES
3C00-3DFF	285	FF	TAPEIO-IV	TAPE ROUTINES
3E00-3FFF	8	FF	SYSTEM-SUBS-I	SYSTEM ROUTINES
10200-103FF	NONE	80	MSCB1	UPPER CORE MSCB
11000-111FF	NONE	80	MQCB	UPPER CORE MQCB

Note * Assigned only if needed.

7.2 What is loaded by the boot load process

To understand what the bootstrap process does it is necessary to know a little about the bootstrap section of FILE-SAVE and SYS-GEN tapes. When the FILE-SAVE proc is run it executes the FORM-LIST verb with the cold-list item in SYSTEM-OBJECT. This selects 31 modes (items in SYSTEM-OBJECT) to be dumped to tape. The COLDDUMP verb reads them from SYSTEM-OBJECT and writes them, in order, to tape in 512 byte blocks, forming the bootstrap section. (This implies that, in order to do a boot load from a FILE-SAVE, cold-list and the 31 items it refers to must be present in SYSTEM-OBJECT.) Then the SAVE verb creates the ABS and files sections.

When you start a boot load from the front panel, one block is read from tape into the first 512 bytes of core. Control is then transfered to the instruction at core location 2Ø. (On DMA systems this doesn't happen until you hit INTERRUPT or RUN on the front panel.) That instruction starts a transfer of the second block on tape into core. Next the OPTIONS message is printed. Then several more blocks are read into core. After MSETUP3 is loaded (see next page) the configurator or disc formatter is run, depending on the option selected. Finally the remainder of the 31 blocks of the bootstrap are loaded and control passes to ABS.

The configurator goes through the following steps when it is run:

1. Read the maximum system configuration from the configuration prom.
2. Print the 'options' message.
3. Print the 'spooler on phantom port' message.
4. Print the 'n is the spooler's line' message.
5. Print number of ABS frames.
6. Sense and print amount of core.
7. Setup the buffer tables and if an upper bank is present move MSCB1 to upper core. This step is not done when warmstarting.
8. Initialize the PIBs if not a warmstart.
9. Sense the disc configuration and print a message for each drive found. A drive is not considered found if its address is not in the configuration prom.
1Ø. Print the 'configuration correct?' message. If you answer 'N' to this question the system will halt.

The table on the next page shows which modes are loaded for each one of the bootstrap options. A 'Y' entry indicates that the mode is loaded when that particular option is selected. The table is in order by location in the bootstrap tape. Note that when a mode name begins with the letters 'SM' it is a ABS frame. The ABS frames in this list may be loaded into core by a boot load. They are not loaded onto disc by the boot load.

MODE	-----OPTION SELECTED------						
	W	X	A	AF	F	D	
MPCB	Y	Y	Y	Y	Y	Y	
MBOOT	Y	Y	Y	Y	Y	Y	
*** OPTIONS MESSAGE DISPLAYED HERE ***							
MSCB	N	Y	Y	Y	Y	Y	
PCBØ	N	Y	Y	Y	Y	Y	
DCBØ	N	Y	Y	Y	Y	Y	
SM-47	N	Y	Y	Y	Y	Y	
MSCB1	N	Y	Y	Y	Y	Y	
FORMAT1	N	Y	Y	Y	Y	Y	
FORMAT2	N	Y	Y	Y	Y	Y	
MSETUP1	Y	Y	Y	Y	Y	Y	
MSETUP2	Y	Y	Y	Y	Y	Y	
MSETUP3	Y	Y	Y	Y	Y	Y	
*** CONFIGURATOR RUN HERE ***							
MONITORY	Y	Y	Y	Y	Y		NO MORE TAPE IS READ
MONITORX	Y	Y	Y	Y	Y		WHEN D IS SELECTED
MONITORZ	Y	Y	Y	Y	Y		
SM-1	N	Y	Y	Y	Y		
SM-17	N	Y	Y	Y	Y		
SM-18	N	Y	Y	Y	Y		
SM-19	N	Y	Y	Y	Y		
SM-2Ø	N	Y	Y	Y	Y		
SM-21	N	Y	Y	Y	Y		
SM-161	N	Y	Y	Y	Y		
SM-21Ø	N	Y	Y	Y	Y		
SM-6	N	Y	Y	Y	Y		
SM-4	N	Y	Y	Y	Y		
SM-7	N	Y	Y	Y	Y		
SM-35	N	Y	Y	Y	Y		
SM-36	N	Y	Y	Y	Y		
SM-285	N	Y	Y	Y	Y		
SM-8	N	Y	Y	Y	Y		
MONITOR	Y	Y	Y	Y	Y		

Note that the X, A, AF, and F restores load in the entire bootstrap section while the W and D options do not.

8 DISC PROBLEMS

To get a history of disc errors see Royale Programmer's Reference Manual. Be sure you have the updated version which explains the DE-RESET, DE-START, DE-STOP, and DE-COPY verbs. Note that the SYSTEM-SETUP proc automatically performs a DE-RESET and DE-START.

8.1 Disc status

The following procedure is how to trap on a disc error and get minor and major status using monitor-debugger.

1. Set up monitor-debugger by pressing STEP then RUN on the front panel with sense switch 2 down.
2. Set a break point at location 1EØ5 (14=Ø2 15=1E 16=Ø5)
3. Press line feed on terminal Ø and system will continue until a disc error occurs.
4. When system breaks into monitor debugger check location 1CØØ for minor status and location ØØØØ for major status.

Major status table (bits numbered from right-to-left)

Bits	Status
Ø-1	Service drive (unit) number Ø,1,2,3
2	Error consult minor status
3	Controller ready
4	Seek error
5	Ten platter disc
6	Not used
7	Returned

Minor status table (bits numbered from right-to-left)

Bit	Status
Ø	Disc not ready
1	Sector not found
2	Platter format protected
3	Dma channel overrun
4	Address (header) check code error
5	Data check code error
6	Sector write protected
7	Disc address non-compare

See disc controller spec for further information on meaning of disc status.

Next you can find out the controller address of the disc with the error.

1. Read memory locations 12Ø and 121. They give you the address of the PIB of the process using the disc.
2. Mentally change the last character of the address to 7.
3. Read that byte of the PIB. The fifth and sixth bits from the left are the last two bits of the device address.

FOR EXAMPLE:

Step 1: M120 is 08.
M121 is 00.
The core address of the PIB is 0800.

Step 2: 0807.

Step 3: M0807 is 04.
The bit pattern of 04 is 0000 0100.
The fifth and sixth bits from the left are 01, so the disc is on device 15 (disc controllers are numbered from hex '14' or 0001 0100 to hex '17' or 0001 0111.)

8.2 To fix certain permanent disc errors

When a series of ampersands, '&', appear on a terminal and continue to appear without stopping, this indicates a permanent disc error. By using the monitor-debugger this problem can often be remedied within a few minutes. The following procedure allows you to fix data check code disc errors.

On releases 3.1 and 3.1B, once ten ampersands have been printed, a percent sign (%) will be printed and the disc read retries will be stopped. At that point the terminal operator has the option of letting the disc read try ten more times by hitting linefeed or stopping the operation by hitting break and typing end.

1. Enter the monitor-debugger on line 0 by putting sense switch 2 down and pressing STEP then RUN. Put a breakpoint at location 1E05 (14=2 15=1E 16=05). Press linefeed and the system should break at that address on the next detected disc error. On 3.1 systems you probably will have to press linefeed one more time on the terminal that reported the error so that it will retry the read.
2. When the debugger breaks at 1E05 look at locations 120 and 121 for the location of the PIB. 'OR' in a X'0A' to that location and look up the contents of that location. If the lower nibble (bits 4-7) is a zero you may proceed, if not, the error cannot be corrected by this means. Next look up locations 000 and 1C00 and if the values in them indicate a bad check code you may proceed. The byte at location zero (major status) should have the error bit turned on. Possible values include X'04' and X'2C'. The byte at location 1C00 (minor status) should be X'20', indicating a data check code error. If these bytes do not have the proper bits on, the problem may be dirty heads or a dirty or scratched surface and cannot be fixed by this procedure.
3. If everything was OK in step 2, then 'OR' in a X'0C' to locations 120 and 121 and read the contents of the next four locations indicated by the address formed. The first location is the address that the disc just read data into with the core address always even and the odd bit indicating

which core bank. The second, third, and fourth bytes form the FID address. Mark both these down for future reference.

4. Put a breakpoint at 1D3C (14=2 15=1D 16=3C) and go to 1D22 (G1D22). On 3.ØF systems, the breakpoint should be at 1D32 and you should go to 1D18.
5. The debugger should break at 1D3C. Shift the contents of the first of the four locations found in step 3 (core buffer address) right one bit and add X'2ØØ' to the result. If that location contained an odd number add an additional 1ØØØØ to the result. Look up the contents of that address and there should be an X'FY' where Y is anything. Change the X'FY' to X'EY', change location 14 to a zero, and exit with X return.
6. The ampersands should go away. The FID that was found in step three might have a error in it since it had a bad data check code. It should be checked with the dump verb and corrected if necessary and possible. Next flush core and try dumping the FID again. If the error comes back then the sector on disk is bad. If it is bad and cannot be fixed immediately and the sector must be used there is a temporary solution in step 7.
7. If the sector found in step 6 is bad you can core lock that frame so that the disc is not used for that FID. Then follow the same procedure as above but in step five instead of changing X'FY' to X'EY' change it to X'8Y'.

For example

Assume the following data after the break at 1EØ5:

LOCATION	DATA
12Ø	Ø8
121	2Ø
82A	ØØ
82C	69
82D	ØØ
82E	Ø6
82F	4Ø
1CØØ	2Ø
ØØØ	Ø4
1Ø234	F3

After the debugger breaks at location 1EØ5 we find that locations 12Ø and 121 have 82Ø in them. ORing in an A results in location 82A.(step 2) Since 82A contains a zero we can proceed to step 3. We then or in a C to locations 12Ø and 121, get 82C, and record the data in 82C, 82D, 82E, and 82F. We put a breakpoint at 1D3C and go to 1D22. When it breaks in step 5, calculate the core buffer address by shifting 69 to the right one bit which changes it to a 34. Then OR in 2ØØ to give 234. Since the 69 is odd, OR in 1ØØØØ also to give a final result of 1Ø234. Looking at that location we find an X'F3' which should be changed to X'E3'. Then type X carriage return and the ampersands hopefully will go away.

8.3 Key Locations

ADDRESS	USE
120-121	PIB address of process that is active when in virtual mode. This location must either contain the start of a PIB (800, 820, 840, etc.) or the monitor's PIB (6F0).
621	Disc controller 14*
622	Disc controller 15*
623	Disc controller 16*
624	Disc controller 17*
625	Device 14 unit 0**
626	Device 14 unit 1**
627	Device 14 unit 2**
628	Device 14 unit 3**
629	Device 15 unit 0**
62A	Device 15 unit 1**
62B	Device 15 unit 2**
62C	Device 15 unit 3**
62D	Device 16 unit 0**
62E	Device 16 unit 1**
62F	Device 16 unit 2**
630	Device 16 unit 3**
631	Device 17 unit 0**
632	Device 17 unit 1**
633	Device 17 unit 2**
634	Device 17 unit 3**

*These locations will contain numbers in the range of 0-4 depending on how many disc controllers are active. (0=No controllers active, 1=1 controller active, etc.)

**These locations have the process associated with the disk unit if the unit is active. The byte will contain the PIB address divided by 16 (shifted right four). For example process 2 PIB =840, the byte will contain a 84. If the disc is not active it will contain a zero. If the disc is active the frame to be read in will be in locations D,E,and F of the PIB and the upper byte of the core location to be read in will be byte C. For example in line 2 the FID will be in locations 84D, 84E, and 84F and the upper byte of the core address will be in location 84C.

8.4 Frame Fault

To watch frame faults in the monitor put a breakpoint at 401 (M14=02, M15=04, M16=01) using the monitor debugger. The FID of the frame that caused the frame fault can be found as follows:

1. Look at core locations 120 and 121. They contain the address of the process frame faulting.
2. Add D to the contents of that location
3. The next three bytes will be the FID

For example if 12Ø and 121 contained a 84Ø, then look at locations 84D,84E,84F. These locations contain the FID and the process frame faulting would be line 2.

8.5 Disc Interrupts

To monitor disc interrupts put a breakpoint at location 4Ø3 using the monitor-debugger(M14=Ø2,M15=Ø4,M16=Ø3). When the system breaks look at location F and that will be twice the device # of the disc controller interrupting.

8.6 Finding Lost Disc Interrupts

When a system hangs with all terminals beeping there are several possible causes. You can determine if a lost disc interrupt is the cause by using the following procedure which utilizes the information in the above sections:

1. While the system is hung break into the monitor debugger.
2. Look in core locations 621-624 for active disc seeks. (The use of these core locations was described in the section Key Locations, above). If they are all zero, the problem is not a lost disc interrupt.
3. If one or more is not zero, look in core locations 625-634 to determine which units are being used. Active units will have non-zero values.
4. Place a breakpoint at the disc interrupt entry to the monitor (M14=Ø1, M15=Ø4, M16=Ø3).
5. Press linefeed. The system should break immediately at location 4Ø3. If it does not, you have "lost a disc interrupt" for one of the devices indicated in step 3. (There will probably only be one non-zero number from step 3).
6. You may now do an INTERRUPT RESET INTERRUPT to "find the interrupt" and resume normal operation.

8.7 Disc Controller Mod For Extended Core

The following method can be used to verify the disc controller modification for extended core. The system must have more than 64K to use this procedure. Cold start the system and when the option message prints out press STEP then RUN to enter the monitor-debugger. Put a break point at 4Ø1 (M14=Ø2, M15=Ø4, M16=Ø1) and press linefeed to continue. Press return and the option message will appear again. Answer the messages until the break occurs into the monitor-debugger. Determine the amount of core in the system and using the table below look at the following two buffers. Mark down a few locations in each. Put a break point at 4Ø3(change M16 to Ø3) and press linefeed to continue. When the system breaks check the same locations as before to determine whether the lower or upper bank of core

changed. If the upper bank changed the modification is probably ok.

Size of Core	Lower Bank Buffer	Upper Bank Buffer
80K	3E00	13E00
96K	7E00	17E00
112K	BE00	1BE00
128K	FE00	1FE00

9 TO CHECK SYSTEM CORESIZE

Using the monitor debugger, look at locations E4 and E5. E4 should always be a zero. E5 should have the core size indicated in the following table. If either of these are not true than something is wrong with the system.

LOCATION E5	MEMORY SIZE
10	16K
18	24K
20	32K
28	40K
30	48K
38	56K
40	64K
50	80K
60	96K
70	112K
80	128K

10 LINE PRINTER I/O

10.1 To watch data going to the line printer

When strange characters appear on the listings or strange formattings occurs such as top of form when one was not expected, it is advantageous to find out what the data bytes look like as they are sent to the controller.

1. Make sure no one else is using the system.
2. Do an SP-ASSIGN N so the spooler will not be used.
3. Output the data to the line printer. i.e. COPY file-name item-name (P) or run the PROC or the BASIC or the RPG program which generates the output.
4. When the printer output gets close to the print line in question, hit break on the terminal to interrupt the output.
5. Set an address stop at 0FFC.
6. Key in a linefeed on the terminal to start printing again.
7. Depress run everytime the CPU halts until the line prior to the one in question has been printed.
8. Change the address stop to 06CE. One data byte will now be

transferred everytime RUN is depressed.
9. Read the data byte from the eight low order data display lamps with the data display button depressed.
10. Press RUN to continue

10.2 Data in I/O buffer for line printer

Another interesting piece of information is the data as it sits in the work buffer. If the data bytes being sent to the controller are correct, software can be eliminated as the problem. If they are not correct, you should look in the OB work area, using the assembler debugger at the time the line in question is in the OB work area. This can be done by hitting the break key instead of doing step 10 above. Put the address stop switch up and when RUN is depressed the debugger will be entered and you can look at the OB work area as follows: (CR means carriage return)

1. Key in X.1D4;2CR. This will display the displacement in hex, .0hhh=
2. Note the displacement and key in CR.
3. Key in I.1D7;3CR. This will display the frame number in decimal, yyyy=
4. Note the frame number and key in CR.
5. Key in Cyyyy.hhh;140CR where yyyy and hhh are the frame number and displacement from steps 3 and 1, respectively. This will print the actual data in character format. Type CR. If you wish to look at the same data in hex, type Xyyyy.hhh;140CR, followed by another CR when it prints the equal sign.
6. Type linefeed.

11 SYSTEM HALTS AND LOOPS

11.1 System halts

L REGISTER	M/N REGISTER	D REGISTER	MEANING
1019	address of error	error#	monitor error
006F	01EC	NA	parity error on coldstart (X,W,A,AF,D)
006F	0437	NA	configuration prom missing on coldstart
006F	04BF	NA	memory error 1st 16k block X,A,AF)

Note: Press RUN after error and the system will go into the monitor-debugger. Further troubleshooting can now be performed.

11.2 Firmware loops

L register	Meaning	Probable cause
103A-103B	Concurrent tape error on reading 1st block of coldstart. M and N registers contain how many bytes were read in. Bytes should have equaled 200 hex.	Bad tape
0E12-0E46	Select next user error PIB links messed up. Interrupt reset interrupt will fix links.	1st 16k memory bad
04FC-0510	Hash links messed up Must coldstart	1st 16k memory in either bank!

12 ADDITIONAL TROUBLESHOOTING TECHNIQUES

12.1 MPCB Errors

The following key locations are loaded by tape into the first 512 (Hex 200) locations in core. They are never changed by the ROYALE system. The monitor-debugger should be used to examine these locations and if any of them have changed the most probably cause of failure would be a bad memory board (1st 16K board)

ADDRESS	CONTENTS
020	90
100	00
101	00
104	00
108	04
109	00
110	06
111	00
118	1C
119	00

12.2 Tape Problems

If there is a DMA or concurrent tape problem, the location of the tape pointers are as follows:

1E4,1E5	starting core address of tape I/O buffer
1E6,1E7	ending core address of tape I/O buffer

At the end of a normal tape transfer the starting address will be one greater than the ending address. Remember that in concurrent tape the firmware updates the starting address and in DMA the hardware updates the address.

12.3 Core Dump To Printer

A core dump will be very useful if either you want to look at a large number of locations, or for analysis at a higher level when in the field. If you decide to do a core dump, do not do any other operations before hand, to avoid destroying useful data.

The core dump will not damage any part of the system. The procedure for coredumps is as follows:

1. Press INTERRUPT or CLOCK to halt system
2. Press RESET
3. Put sense switch 3 down
4. Ready printer
5. Press RUN

For the most meaningful dumps allow all existing core to be dumped. The core dump program is in firmware and will dump 128K of core if not stopped. The core dump can be stopped by putting sense switch 3 up. Note that the address will cycle through 64K for both the upper and lower core banks.

13 PROPER MEMORY HARDWARE MODIFICATION CHECK

The first 16K of the lower core bank must be modified to at least rev 'N' electronics. All other 16K boards must be modified only if using extended core. To check proper modification, use the monitor debugger to zero the lower core location of the module modified and place all ones in the corresponding upper bank location. Then read the lower core location and check for zero. Read the upper core location and check for all ones. If both banks read the same there is something wrong with the modification.

For example set location Ø to a Ø and location 1ØØØØ to X'FF' and read both back.

14 CONFIGURATION CONTROL

Configuration control prevents a system from using devices or disk frames not assigned to that system. A prom on the 8K ROM board defines the configuration of the system. The verb "LIMITS" will print out the configuration of the system. The following table shows what happens when the configuration control is violated.

Violation	In monitor mode	In virtual mode
Core	Monitor error halt with X'13' in D register	Same as monitor mode
Devices	I/O instruction nop'ed	Abort into assembler debugger

		with illegal device address
Frame id's	Monitor error halt with X'Ø8' in D register	Abort into assembler debugger with referencing illegal frame message

15 CONCURRENT INTERRUPTS FOR DEVICE ZERO

The Royale system only uses device zero for bisync. If bisync is not present in the system the address pointers for device zero should never change. Some problems have been experienced using the GPIO board as a printer output device. Concurrent interrupts for device zero have appeared due to a malfunction in the printer-GPIO interface. This problem can easily be verified by looking at the pointers for device zero which are at locations 1CØ and 1C1 in core. With the monitor-debugger look at the two addresses and record them. After using the printer extensively or when the system hangs up look at the addresses again. If they have changed you most likely have a "printer-device zero" problem. The problem is usually noise on the printer ready line from the GPIO.

16 SYSTEM PERFORMANCE AND OPERATION TOOLS

16.1 DISCIO verb

The DISCIO verb displays the amount of system disc I/O activity and provides a measurement of system disc performance. This display aides in locating causes of system performance problems. DISCIO generates two reports depending on the specified options. The first report (Report 1) consists of disc unit I/O's per second, disc I/O's per second, disc reads per second, disc writes and verifies per second, cummulative disc unit I/O's, cummulative disc I/O's, cummulative disc reads, and cummulative disc writes and verifies. The cummulative values are zero at the DISCIO operation's start. The second report (Report 2) consists of disc unit I/O totals. The general verb form is :

DISCIO (options)

To ensure accurate results during Report 1 displays, no ports may be logged off.

Three options exist for the DISCIO verb. The "T" option indicates the report type. When this option is specified, Report 2 is generated. Without this option, Report 1 is generated. The "n" option indicates the number of display iterations with "n" being any decimal number. This option only effects Report 1 displays. If this option isn't specified, one display iteration is performed. Report 2 always performs one display iteration. The "P" option indicates the output device. When this option is specified, the report is output to the printer. Without this option, the report is output to the terminal.

DISCIO reveals system performance problems. A low number of disc I/O's per second and a high number of active processes may indicate a system disc I/O bottleneck. This problem may be caused by an inadequate number of disc units or controllers, a slow disc seeking mechanism, or inefficient MONITOR disc handling code. A high number of reads per second and a high proportion of I/O's on one disc unit may imply a permanent disc error. To correct this error, see DISC ERRORS section of REALITY TROUBLESHOOTING MANUAL.

16.2 BUFFERS verb

The BUFFERS verb displays the core memory contents and provides a measurement of core buffer I/O activity. This verb displays two reports depending on the specified options. The first report (Report 1) contains core buffer memory locations, and FIDs and status of frames in core buffer memory locations. The core buffer status consists of I/O busy, temporary CORELOCK, permanent CORELOCK, and write required indicators. The second report (Report 2) contains the number of core buffers occupied by ABS frames, work space frames, and user program and data frames. This report includes the number of read-active core buffers. The general verb form is :

BUFFERS (options)

Four options exist for the BUFFERS verb. The "T" option indicates the report type. If this option is specified, Report 2 is generated. Without this option, Report 1 is generated. The "S" option indicates a sorted report. This option only effects Report 1 displays and creates a sorted listing by core buffer frame FID. Without this option, Report 1's display is in core buffer memory location order. The "P" option indicates the output device. If this option is specified, the report is output to the printer. Without this option, the report is output to the terminal. The "N" option inhibits automatic paging of terminal output. Without this option, the terminal display pauses at the page end until some terminal input is entered. With this option, no pausing occurs.

The BUFFERS verb aides in system operation. Before coldstart operations, no write required frames may be core resident to reduce the possibility of GROUP FORMAT ERRORS. The MONITOR automatically flushes core when no active processes exist. Because Report 1 displays all core resident, write required frames, BUFFERS may be used to determine if a coldstart operation may be safely performed.

16.3 WHERE verb

WHERE prints one line for every process logged on (including the spooler). It is more accurate for telling which ports are logged on than LISTU because LISTU depends on the ACC (accounting) file being correct.

The output of WHERE shows you port number, PCB for that port (in hex), PIB status byte, and the software return stack.

EXAMPLE:

```
:WHERE          CR>
  PORT     PS    RTN STACK..
02 0320    7F     121.000 121.07D
04 0360    7D       6.08A   6.040   5.064
10 0420    7D      21.034   6.08A   6.040   5.064
31 06C0    3F     165.080 164.052
```

The return stack is particularly valuable. It tells you precisely what a process is doing at any time by showing the locations in ABS of the instructions being executed. Each return stack entry is part of a subroutine that was called by the next return stack entry.

Consider the return stack for port 4 in the previous example:

6.08A 6.040 5.064

This says that the port is executing an instruction in frame 6 at a displacement of '8A' bytes into the frame. That instruction is part of a subroutine which, when it is finished executing, will return to another instruction in frame 6 at displacement '40' bytes. That, in turn, is part of a subroutine which will return to an instruction in frame 5 at displacement '64' bytes into the frame. This port, by the way, is at TCL.

If you know what parts of the operating system reside in which frames, you can get a good idea of what the system is doing at any time by looking at the WHERE output. On 2.4 and 2.5 level systems this was easy. Appendix D of the Programmer's Reference Manual told you what each frame does. In the above example, frame 6 is called "TERMIO" and frame 5, "TCL-I". The instructions in frame 6 are waiting for terminal input and have been called by the TCL processor.

Port 2 is executing in frame 121, which is called "WHERESUBS". That is the port that is actually doing this WHERE command.

Port 31, the spooler, is executing frames 165 and 164, which are called "SPOOLADD" and "SPOOLOUT".

Whenever the first entry in the return stack is in frame 21, that process is in the assembly language debugger, either because someone hit the break key or the process aborted. Port 1Ø is in the debugger and was waiting for terminal input at TCL immediately before the debugger was entered. In this case someone simply hit break while the process was at TCL.

On 3.Ø and 3.1 systems there is no appendix telling what each frame does. The following list shows the function of some of the more commonly used frames.

FRAMES	FUNCTION
5	TCL
6	TERMINAL I/O
13-16	EDITOR
21	DEBUGGER
34	OVERFLOW TABLE
35,36	TAPE I/O
122	SLEEP
127	OVERFLOW TABLE
154,155	BASIC
163-171	SPOOLER
175,18Ø-188	BASIC
2ØØ-2Ø6	FILE-SAVE
21Ø-219	FILE RESTORE
38Ø-399	BISYNC

Here are a few sample return stacks from a 3.1B system and descriptions of what the processes were doing:

122.1DB 122.181
  Asleep. You often see this in ATP.

6.Ø8A 6.Ø4Ø 156.ØDC 13.Ø5Ø
  Waiting for terminal input in the editor.

6.Ø8A 6.Ø4Ø 181.1A7
  Waiting for terminal input in BASIC.

394.13Ø 393.1Ø3 387.Ø4B
  Bisync.

165.Ø8Ø 164.Ø52
  The spooler.

2Ø3.Ø51 2Ø3.1DE 2Ø2.18D 2Ø1.158
  Doing a file save.

One other note... You get a return stack when you run the MONITOR proc in ATP. The return stack addresses will be one byte less than the ones WHERE gives you. MONITOR shows the address of the last byte of the last instruction executed (the subroutine call) while WHERE shows the address of the first byte of the next instruction. That is, MONITIOR prints where an instruction was called from and WHERE prints where it will return to.

EXAMPLES:

The MONITOR return stack for TCL is
```
     6.Ø89   6.Ø3F   5.Ø63
```

The MONITOR return stack for the spooler is
```
   165.Ø7F 164.Ø51
```

17 APPENDIX A. ASCII CODE CHART

DECIMAL	HEX	EBCIDIC EQUIVALENT	ASCII	PRISM CHARACTER	SPECIAL USE IN ROYALE DISPLAY
Ø	ØØ	ØØ	NUL	NONE P cs	DELAY CHAR. SORT KEY DELIMITER
1	Ø1	Ø1	SOH	NONE A c	PRISM HOME CHARACTER
2	Ø2	Ø2	STX	NONE B c	
3	Ø3	Ø3	ETX	NONE C c	END OF TEXT
4	Ø4	37	EOT	NONE D c	
5	Ø5	2D	ENQ	NONE E c	
6	Ø6	2E	ACK	NONE F c	CURSOR FOWARD ON PRISM
7	Ø7	2F	BEL	NONE G c	BELL ON PRISM
8	Ø8	16	BS	NONE H c	BACKSPACE ON PRISM
9	Ø9	Ø5	HT	NONE I c	TAB
1Ø	ØA	25	LF	NONE J c	CURSOR DOWN ON PRISM
11	OB	OB	VT	NONE K c	VERTICAL ADDRESS ON PRISM
12	OC	OC	FF	NONE L c	SCREEN ERASE ON PRISM
13	ØD	ØD	CR	NONE M c	CARRIAGE RETURN
14	ØE	ØE	SO	NONE N c	
15	ØF	ØF	SI	NONE O c	
16	1Ø	1Ø	DLE	NONE P c	HORIZONTAL ADDRESS ON PRISM, BLANK COMPRESSION CHARACTER
17	11	11	DC1	NONE Q c	
18	12	12	DC2	NONE R c	RETYPE ENTIRE LINE. ENABLE SLAVE PRINTER
19	13	3A	DC3	NONE S c	DUMP PRISM SCREEN TO SLAVE PRINTER
2Ø	14	3C	DC4	NONE T c	DISABLE SLAVE PRINTER
21	15	3D	NAK	NONE U c	CURSOR BACK ON PRISM
22	16	32	SYN	NONE V c	
23	17	26	ETB	NONE W c	
24	18	18	CAN	NONE X c	CANCEL LINE
25	19	19	EM	NONE Y c	
26	1A	3F	SUB	NONE Z c	CURSOR UP ON PRISM
27	1B	27	ESC	NONE	
28	1C	1C	FS	NONE	
29	1D	1D	GS	NONE	
3Ø	1E	1E	RS		
31	1F	1F	US	NONE	
32	2Ø	4Ø	SPACE	b SPACE	
33	21	5A	!	! A cs	
34	22	7F	"	" B cs	STRING DELIMITER IN ENGLISH AND BASIC
35	23	7B	#	# C cs	
36	24	5B	$	$ D cs	
37	25	6C	%	% E cs	
38	26	5Ø	&	& F cs	

39	27	7D	'	'	G cs	STRING DELIMITER IN ENGLISH AND BASIC
40	28	4D	(	(	H cs	
41	29	5D	)	)	I cs	
42	2A	5C	*	*	J cs	
43	2B	4E	+	+	K cs	
44	2C	6B	,	,	L cs	
45	2D	60	-	-	M cs	
46	2E	4B	.	.	N cs	
47	2F	61	/	,	O cs	
48	30	F0	0	0	P cs	
49	31	F1	1	1	Q cs	
50	32	F2	2	2	R cs	
51	33	F3	3	3	S cs	
52	34	F4	4	4	T cs	
53	35	F5	5	5	U cs	
54	36	F6	6	6	V cs	
55	37	F7	7	7	W cs	
56	38	F8	8	8	X cs	
57	39	F9	9	9	Y cs	
58	3A	7A	:	:	Z cs	
59	3B	5E	;	;		
60	3C	4C	<	<		
61	3D	7E	=	=		
62	3E	6E	>	>		
63	3F	6F	?	?		
64	40	7C	@	@		
65	41	C1	A	A		
66	42	C2	B	B		
67	43	C3	C	C		
68	44	C4	D	D		
69	45	C5	E	E		
70	46	C6	F	F		
71	47	C7	G	G		
72	48	C8	H	H		
73	49	C9	I	I		
74	4A	D1	J	J		
75	4B	D2	K	K		
76	4C	D3	L	L		
77	4D	D4	M	M		
78	4E	D5	N	N		
79	4F	D6	O	O		
80	50	D7	P	P		
81	51	D8	Q	Q		
82	52	D9	R	R		
83	53	E2	S	S		
84	54	E3	T	T		
85	55	E4	U	U		
86	56	E5	V	V		
87	57	E6	W	W		
88	58	E7	X	X		
89	59	E8	Y	Y		
90	5A	E9	Z	Z		
91	5B	80	[	[		STRING SEARCH DELIMITER
92	5C	E0	/	/		

93	5D	9Ø	]	[	ENGLISH STRING SEARCH DELIMITER
94	5E	5F	^	^	ENGLISH STRING SEARCH DELIMITER
95	5F	6D	_	_	
96	6Ø	79		NONE	Ø cs
97	61	81		NONE	1 cs
98	62	82		NONE	2 cs
99	63	83		NONE	3 cs
1ØØ	64	84		NONE	4 cs
1Ø1	65	85		NONE	5 cs
1Ø2	66	86		NONE	6 cs
1Ø3	67	87		NONE	7 cs
1Ø4	68	88		NONE	8 cs
1Ø5	69	89		NONE	9 cs
1Ø6	6A	91		NONE	
1Ø7	6B	92		NONE	
1Ø8	6C	93		NONE	
1Ø9	6D	94		NONE	
11Ø	6E	95		NONE	
111	6F	96		NONE	
112	7Ø	97		NONE	Ø c
113	71	98		NONE	1 c
114	72	99		NONE	2 c
115	73	A2		NONE	3 c
116	74	A3		NONE	4 c
117	75	A4		NONE	5 c
118	76	A5		NONE	6 c
119	77	A6		NONE	7 c
12Ø	78	A7		NONE	8 c
121	79	A8		NONE	9 c
122	7A	A9		NONE	
123	7B	CØ		NONE	
124	7C	6A		NONE	
125	7D	DØ		NONE	
126	7E	A1		NONE	SORT KEY DELIMITER
127	7F	Ø7	DEL	NONE	
128	8Ø	Ø4		NONE	
129	81	Ø6		NONE	
13Ø	82	Ø8		NONE	
131	83	Ø9		NONE	
132	84	ØA		NONE	
133	85	13		NONE	
134	86	14		NONE	
135	87	15		NONE	
136	88	17		NONE	
137	89	1A		NONE	
138	8A	1B		NONE	
139	8B	2Ø		NONE	
14Ø	8C	21		NONE	
141	8D	22		NONE	
142	8E	23		NONE	
143	8F	24		NONE	
144	9Ø	28		NONE	
145	91	29		NONE	

146	92	2A		NONE
147	93	2B		NONE
148	94	2C		NONE
149	95	30		NONE
150	96	31		NONE
151	97	33		NONE
152	98	34		NONE
153	99	35		NONE
154	9A	36		NONE
155	9B	38		NONE
156	9C	39		NONE
157	9D	3B		NONE
158	9E	3E		NONE
159	9F	41		NONE
160	A0	42		NONE
161	A1	43		NONE
162	A2	44		NONE
163	A3	45		NONE
164	A4	46		NONE
165	A5	47		NONE
166	A6	48		NONE
167	A7	49		NONE
168	A8	4A		NONE
169	A9	4F		NONE
170	AA	51		NONE
171	AB	52		NONE
172	AC	53		NONE
173	AD	54		NONE
174	AE	55		NONE
175	AF	56		NONE
176	B0	57		NONE
177	B1	58		NONE
178	B2	59		NONE
179	B3	62		NONE
180	B4	63		NONE
181	B5	64		NONE
182	B6	65		NONE
183	B7	66		NONE
184	B8	67		NONE
185	B9	68		NONE
186	BA	69		NONE
187	BB	70		NONE
188	BC	71		NONE
189	BD	72		NONE
190	BE	73		NONE
191	BF	74	b	NONE
192	C0	75	@	@
193	C1	76	A	A
194	C2	77	B	B
195	C3	78	C	C
196	C4	8A	D	D
197	C5	8B	E	E
198	C6	8C	F	F
199	C7	8C	G	G
200	C8	8E	H	H

201	C9	8F	I	I		
202	CA	9A	J	J		
203	CB	9B	K	K		
204	CC	9C	L	L		
205	CD	9D	M	M		
206	CE	9E	N	N		
207	CF	9F	O	O		
208	DO	AØ	P	P		
209	D1	AA	Q	Q		
210	D2	AB	R	R		
211	D3	AC	S	S		
212	D4	AD	T	T		
213	D5	AE	U	U		
214	D6	AF	V	V		
215	D7	BØ	W	W		
216	D8	B1	X	X		
217	D9	B2	Y	Y		
218	DA	B3	Z	Z		
219	DB	B4	[	[		
220	DC	B5	/	/		
221	DD	B6	]	]		
222	DE	B7	^	^		
223	DF	B8				
224	EØ	B9	@	@		
225	E1	BA	a	A		
226	E2	BB	b	B		
227	E3	BC	c	C		
228	E4	BD	d	D		
229	E5	BE	e	E		
230	E6	CA	f	F		
231	E7	CA	g	G		
232	E8	CB	h	H		
233	E9	CC	i	I		
234	EA	CD	j	J		
235	EB	CE	k	K		
236	EC	CF	l	L		
237	ED	DA	m	M		
238	EE	DB	n	N		
239	EF	DC	o	O		
240	FØ	DD	p	P		
241	F1	DE	q	Q		
242	F2	DF	r	R		
243	F3	E1	s	S		
244	F4	EA	t	T		
245	F5	EB	u	U		
246	F6	EC	v	V		
247	F7	ED	w	W		
248	F8	EE	x	X		
249	F9	EF	y	Y		
250	FA	FA	z	Z		
251	FB	FB	SB	[	K cs	START BUFFER
252	FC	FC	SVM	/	L cs	SUBVALUE MARK
253	FD	FD	VM	]	M cs	VALUE MARK
254	FE	FE	AM	^	N cs	ATTRIBUTE MARK
255	FF	FF	SM	_	O cs	SEGMENT MARK

OBJECT BASE	COMMAND	MNENOMIC	CLASS
0000	EXECUTE, LITERAL TYPE	ELT	2
0000	EXECUTE, OPERATE TYPE	EOT	1
0000	JUMP EXTENDED	JE	5
1000	LOAD ZERO CONTROL	LZ	3
1000	NO OPERATION	NOP	4
1001	SET FILE 0 BIT 6	LZ 1	4
1002	RESET FILE 0 BIT 6	LZ 2	4
1020	RETURN	RTN	4
1040	SELECT PRIMARY FILE	SPF	4
1060	RETURN,SELECT PRIMARY FILE	RSP	4
1080	SELECT SECONDARY FILE	SSF	4
10A0	RETURN,SELECT SECONDARY FILE	RSS	4
1100	LOAD T	LT	3
1200	LOAD M	LM	3
1300	LOAD N	LN	3
1400	JUMP IN 1K	JP	3
1500	JUMP IN 1K	JP	3
1600	LOADU	LU	3
1700	LOAD SEVEN CONTROL	LS	3
1704	DISABLE EXTERNAL INTERRUPTS	DEI	4
1708	ENABLE EXTERNAL INTERRUPTS	EEI	4
1710	DISABLE R.T. CLOCK	DRT	4
1720	ENABLE R.T. CLOCK	ERT	4
1780	HALT	HLT	4
1800	LOAD EIGHT CONTROL	LE	3
1900	RETURN, LOAD T	RLT	3
1A00	MODIFY LOWER COMMAND	MLC	4
1B00	INHIBIT L SAVE	ILS	4
1B08	BANK SELECT LOWER	BSL	3
1B09	BANK SELECT UPPER	BSL	3
1C00	JP IN 1K	JP	3
1D00	JP IN 1K	JP	3
2000	LOAD FILE WITH LITERAL	LF	2
3000	ADD FILE WITH LITERAL	AF	2
4000	TEST IF ZERO	TZ	2
5000	TEST NOT ZERO	TN	2
6000	COMPARE FILE	CP	2
7010	ENTER SENSE SWITCHES	ESS	1
7020	SHIFT FILE RIGHT 4	SRF	1
7040	ENTER INTERNAL STATUS	EIS	1
7070	ENTER CONSOLE SWITCHES	ECS	4
7080	CLEAR I/O MODE	CIO	1
7090	CONTROL OUTPUT	COX	1
70A0	DATA OUTPUT	DOX	1
70C0	CONCURRENT ACKNOWLEDGE	CAK	1
70D0	INTERRUPT ACKNOWLEDGE	IAK	1
70E0	DATA INPUT	DIX	1
8000	ADD FILE	ADD	1

9000	SUB FILE,TWOS COMPLEMENT	SBT	1
9040	SUBTRACT FILE ONES COMPLEMENT	SBO	1
9040	DECREMENT FILE	DEC	1
A000	READ MEMORY,FULL CYCLE	RMF	1
A010	WRITE MEMORY,FULE CYCLE	WMF	1
A020	READ MEMORY,HALF CYCLE	RMH	1
A030	WRITE MEMORY,HALF CYCLE	WMH	1
B000	COPY	CPY	1
B040	+1 TO FILE/REG.	POF	1
C000	LOGICAL OR WITH FILE	LOR	1
C000	MOVE FILE	MOV	1
D000	EXCLUSIVE OR WITH FILE	XOR	1
E000	AND WITH FILE	AND	1
F000	SHIFT FILE LEFT	SFL	1
F020	SHIFT FILE RIGHT	SFR	1
F040	SHIFT FILE LEFT AND INSERT	SLI	1
F060	SHIFT FILE RIGHT AND INSERT	SRI	1

19 APPENDIX C. MONITOR DEBUGGER

19.1 Introduction

The Monitor Debugger is an extension to the Royale firmware set, permanently included in 3.Ø firmware. When not active it does not affect the execution time of the Royale instruction set. It is used in conjunction with the line Ø terminal and is activated with the front panel STEP switch. When activated, the Monitor Debugger can access and change any location in core and can be used with one of four trace modes for troubleshooting.

The Monitor Debugger is entered for the first time by placing sense switch 2 down, pressing STEP, which should halt the computer, and then pressing RUN, which should cause terminal zero to break into the Monitor Debugger with the address and primary opcode of the next instruction to be executed displayed. Since this is a monitor control debugger, the whole Royale system is stopped with the exception of concurrent processing and DMA. DMA and concurrent processing, if in progress, finish the current operation.

Starting with version IX firmware boards, any time the Monitor Debugger is entered the M and N registers will contain the following values:

M - Upper byte of current PCB's word address

N - PIB Link of current active PIB

For example, if M = 6C and N = 28, then the PCB of the process that was executing starts at core location 6CØØ and its PIB is at core location 82Ø. (To convert PIB links to core locations you reverse the digits and add a zero.) If M is zero, then the monitor was in control and N would be meaningless. Note that execution of any Monitor Debugger commands that affect memory locations will change the M and N registers.

The Monitor Debugger remains in control until the operator intervenes as follows:

1. Type X <CR> on the CRT. This forces control back to the ROYALE system and shuts off the debugger.
2. INTERRUPT-RESET-INTERRUPT on the CPU front panel likewise resets the debugger and performs its normal operation.

To exit with the Monitor Debugger still in control press either a linefeed or a G return. This will cause a return to the Royale operating system. Under these conditions the debugger may be entered again by pressing the break key as long as sense switch 2 remains down. If sense switch 2 is up then the break key will act normally and force entrance into the assembler debugger. The

Monitor Debugger will also be entered again if a break condition is met regardless of the position of sense switch 2.

19.2 Memory command (M)

The memory command is of the form Mxxxxx where the x's are a core memory location in hexadecimal. See section titled 'Hexadecimal parameters' for the conventions on entering hex values. Blank terminates the address and causes the hex value of the byte stored at that location to be printed followed by an = sign. The value at that location can be changed simply by keying in a new hex value followed by one of the allowable delimiters. If the byte is not to be changed, simply key in one of the delimiters without keying in a hex value. The allowable delimiters and their meanings are:

Blank	Display the value at the next consecutive core memory location on the current line.
Control N	Display on the next line the address of the next consecutive core memory location followed by the value at that location.
Control P	Display on the next line the address of the previous core memory location followed by the value at that location.

All other characters return control to the main debugger loop so a different command may be entered.

19.3 Go command (G)

The go command is of the form Gxxxxx where the x's represent an execution address in hexadecimal. See section titled hexadecimal parameters for the conventions of entering hex values. If no execution address is entered, the Royale firmware will begin execution where it left off. Otherwise execution will begin at the address specified.

19.4 Trace modes

There are four trace modes which are controlled by the bytes at core locations X'14' - X'17'. The byte at location X'14' controls which trace mode is currrently in effect and upper or lower core bank. The following table shows the values of this byte and the corresponding trace mode evoked.

ØØ	No trace
Ø1	Single instruction trace
Ø2	Break point trace, low core
Ø4	Break on change of a byte to a particular value, low core
Ø8	Break on change of a byte from a particular value, low core
14	Break on change of byte to a particular value, low core

and address match
18 Break on change of byte from a particular value, low core and address match
82 Break point trace, high core
84 Break on change of a byte to a particular value, high core
88 Break on change of a byte from a particular value, high core
A4 Break on change of a byte to a particular value, high core and address match
A8 Break on change of a byte from a particular value, high core and address match

The single instruction trace breaks to the Monitor Debugger after execution of every Royale instruction. Some Reality instructions, because of there internal nature, will break more than once for every RNI cycle. No other bytes besides the one at X'14' are used to control the trace mode.

The break point trace uses the bytes at locations X'15' and X'16'. These bytes contain the execution address which will cause a break to the monitor debugger when Royale tries to execute at that location. Remember that upper or lower core is selected by the highest bit of the byte at location X'14'.

The break on change uses all four bytes. The bytes at location X'15' and X'16' contain the core memory location of the byte which is being watched with the upper bit of location X'14' telling which bank. The byte at X'17' contains the value used for comparison when breaking on the change of a byte.

There are three bytes at address X'88' thru X'8A' which are used for an address match when checking for both byte change and address match. The reason for three bytes is to allow for 128k of core.

19.5 Hexadecimal parameters

Hex parameters include core memory addresses on memory commands, hex values to change core memory, and execution addresses on go commands. For values which are expected to be five hex digits, if fewer than five hex digits are keyed in, high order zeroes are assumed. If more than five hex digits are keyed in, only the last five are used. For values which are expected to be two hex digits, if only one is keyed in, a high order zero is assumed. If more than two hex digits are keyed in, only the last two are used. Non hex characters (except blank and control characters) are converted to some hex digit. Blank and control characters terminate a hex parameter.

20 APPENDIX D. FIRMWARE LOCATIONS

20.1 Prom Chip Numbering System

NEW SYSTEM LOWER	NEW SYSTEM UPPER BYTE	OLD SYSTEM SEGMENT BYTE	HEX ADDRESS 0=LOWER BYTE 1=UPPER BYTE
1	2	SEG 0/1	0000-01FF
3	4	SEG 2/3	0200-03FF
5	6	SEG 4/5	0400-05FF
7	8	SEG 6/7	0600-07FF
9	10	SEG 8/9	0800-09FF
11	12	SEG 10/11	0A00-0BFF
13	14	SEG 12/13	0C00-0DFF
15	16	SEG 14/15	0E00-0FFF
17	18	SEG 16/17	1000-11FF
19	20	SEG 18/19	1200-13FF
21	22	SEG 20/21	1400-15FF
23	24	SEG 22/23	1600-17FF
25	26	SEG 24/25	1800-19FF
27	28	SEG 26/27	1A00-1BFF
29	30	SEG 28/29	1C00-1DFF
31	32	SEG 30/31	1E00-1FFF

20.2 Firmware Map

LOCATION	USE
0000-01F1	ROYALE INSTRUCTIONS
01F2-01FF	FREE
0200-02EF	ROYALE INSTRUCTIONS
02F0-02FF	ROYALE INSTRUCTIONS
0300-03FB	ROYALE INSTRUCTIONS
03FC-03FF	ROYALE INSTRUCTIONS
0400-07A6	ROYALE INSTRUCTIONS
07A7-07A9	FREE
07AA-07DB	8K DIAGNOSTIC
07DC-07FF	ROYALE INSTRUCTIONS
0800-0CEF	ROYALE INSTRUCTIONS
0CF0-0CFF	ROYALE INSTRUCTIONS
0D00-0DFF	ROYALE INSTRUCTIONS
0E00-0EFF	ROYALE INSTRUCTIONS
0F00-0FFC	ROYALE INSTRUCTIONS
0FFD-0FFF	FREE
1000-11E1	ROYALE INSTRUCTIONS
11E2-11EC	FREE
11ED-11FF	RDF RETURN AND EXTENDED CORE
1200-1275	CORE DUMP

1276-127F	FREE
1280-12F2	MONITOR-DEBUGGER
12F3-12FF	FREE
1300-13FF	MONITOR-DEBUGGER
1400-1400	8K DIAGNOSTIC
1401-1472	MEMORY DIAGNOSTIC
1473-14DA	RD00
14DB-14FF	FREE
1500-150A	RD00
150B-150F	FREE
1510-1519	TAPE DIAGNOSTIC
151A-15AD	FREE
15AE-17FF	8K DIAGNOSTIC
1800-19FD	RDF
19FE-19FF	FREE
1A00-1B35	TAPE DIAGNOSTIC
1B36-1BFF	RD00
1C00-1FD3	RD00
1FD4-1FFF	FREE