Novell Boot ROM

Developer's Guide for

DOS Workstations

9 July 1992

Version 1.0

Part Number 107-000026-001

Disclaimer

Novell, Inc. makes no representations or warranties with respect to the contents or use of this manual, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Novell has made every effort to supply trademark information about company names, products, and services mentioned in this document. Trademarks indicated below were derived from various sources.

Borland is a trademark of Borland, Incorporated.

Ethernet is a trademark of Digital Equipment, Incorporated, Intel Incorporated, and Xerox, Incorporated.

IBM, Micro Channel, and Token-Ring are trademarks of International Business Machines, Incorporated.

Intel is a trademark of Intel, Incorporated.

Microsoft is a trademark of Microsoft, Incorporated.

NetWare and Novell are registered trademarks of Novell, Incorporated.

References

The following publications contain information that might be helpful when you read this document:

IBM Remote Program Load User's Guide, 83X7840

IBM Token-Ring Network Architecture Reference, SC30-3374-02 39F9354

Open Data-Link Interface Developer's Guide for NetWare DOS Workstation Drivers, Part Number 107-000010-001

Open Data-Link Interface Developer's Guide for DOS Workstation Protocol Stacks

Table of Contents

The Boot ROM Developer's Kit 1
Contents of the Boot ROM Developer's Kit 1
The Boot ROM Creation Process 3
Assumptions about the HSM 4
Theory of Operation 4
Booting a Diskless Workstation with RPL 5
Quick Reference 5
Description of Procedure 5
Locating An RPL File Server 7
The Find Frame 7
The Found Frame 9
The Send File Request Frame 10
The File Data Response Frame 11
The Boot ROM Interfaces 11
The Boot ROM LSL Interface 11
The Boot ROM LSL MLID Support Interface . . 12
The Boot ROM LSL Protocol Support Interface 13
The Boot Strap Program 15
Boot Strap Program Machine State 15
Enabling the Boot Strap Program to Use the HSM 16
The RPL Display Screen 16

List of Figures

Figure 1. Flow of Operations 6
Figure 2. Find/Found Frame RPL Protocol Flow Diagram 7
Figure 3. Format of the Find Frame 8
Figure 4. Hexadecimal Trace of a Find Frame from an Ethernet Board 8
Figure 5. Format of the Found Frame 9
Figure 6. Hexadecimal Trace of a Found Frame from an Ethernet Board 10
Figure 7. Format of the Send File Request Frame 10
Figure 8. Hexadecimal Trace of a Send File Request Frame from an Ethernet Board 11
Figure 9. Format of the File Data Response Frame 11
Figure 10. Hexadecimal Trace of a File Data Response Frame from an Ethernet Board 12
Figure 11. Block Diagram of Boot ROM Interfaces 12
Figure 12. An Example RPL Display Screen 17

List of Tables

Table 1. RPL Basic Events 5
Table 2. Exceptions to Standard LSL Functionality 13
Table 3. Limitations of the HSM IOCTL Interface 13
Table 4. Changes to the LSL Protocol Support Interface 14
Table 5. Functions Not Implemented in the Boot ROM LSL 14
Table 6. Registers in Effect During the Boot Strap Program 15
Table 7. Status of Registers when Calling a Protocol Stack 16
Table 8. Three-Character IDs 17

The Boot ROM Developer's Kit

The Novell Boot ROM Developer's Kit allows board manufacturers to create a Boot ROM by linking an HSM (Hardware Specific Module) used for DOS ODI workstations with object files in the kit provided by Novell. You can use most HSMs without modification, easing version control of the software and reducing the amount of code developed by the board manufacturer.

The Boot ROM you create uses the IBM Find/Found RPL Protocol to allow a hardware independent Boot Strap Program to boot a diskless workstation. The IBM RPL Protocol is used because it is generic and allows the board manufacturer to provide a single Boot ROM that will boot from any network supporting the protocol. Currently, this includes Novell NetWare, Microsoft LAN Manager, and IBM LAN Server.

Contents of the Boot ROM Developer's Kit

The Novell Boot ROM Developer's kit provides the following object modules:

BSM.OBJ	Module similar to the Media Specific Module (MSM) in the LAN Driver Developer's Kit; it provides a Link Support Layer (LSL) and MSM interface for the HSM and the Boot Strap Program.
BSMETHER.OBJ	Ethernet media module that links with BSM.OBJ to provide a Boot ROM for any Ethernet HSM.
BSMFDDI.OBJ	FDDI media module that links with BSM.OBJ to provide a Boot ROM for any FDDI HSM.
BSMTOKEN.OBJ	Token-Ring media module that links with BSM.OBJ to provide a Boot ROM for any Token-Ring HSM.

In addition, the kit includes the following source code so that the board manufacturer can customize the Boot ROM:

BSM.ASM	Source code for BSM.OBJ.
BSMETHER.ASM	Source code for BSMETHER.OBJ.
BSMFDDI.ASM	Source code for BSMFDDI.OBJ.

BSMTOKEN.ASM	Source Code for BSMTOKEN.OBJ.
ODI.INC	Include file required by the source code.
RPL.INC	Include file required by BSM.ASM.
ROMMAP.INC	Include file used by BSM.ASM to determine the possible ROM locations and adapter configurations. This file allows the adapter to use different interrupts, ports, DMA channels, etc., depending upon the setting of the ROM location. ROMMAP.INC also includes the sample code of the NE1000 and NE2000 drivers. The NE2100 and 1500T drivers must change the ROMPad from 0000 to 9500.

The kit also includes two utilities to allow the board manufacturer to test the Boot ROM before it is burned into a PROM:

ROMSUM.COM	Utility that does a checksum on the ROM and adjusts the size of the ROM image file according to input specifications. The size of the ROM created can be anywhere from 4k (4096) bytes to 127.5k bytes (128k bytes - 512 bytes = 127.5k bytes or 130,560 bytes) in 512 byte pages. (PC architecture imposes a ROM limit of 127.5k bytes.)
TESTROM.COM	Utility that uses the ROM image file as input, places the ROM image in high memory, and simulates the BIOS memory scan performed by POST (Power On Self Test). This utility validates that the ROM image file has a checksum of 00, calls OFFSET 03 from the ROM image file, and issues INT 18h to start the RPL operation.

The Boot ROM Creation Process

To create a Boot ROM:

1. Modify ROMMAP.INC to include the possible ROM locations and configuration options of the adapter.

   Note: This step is not required if the adapter is self-configurable or if the HSM handles the configuration options. However, if you change the configuration information in ROMMAP.INC, you must re-assemble the BSM.ASM file to create a new BSM.OBJ file. Use Borland's TASM.EXE or any compatible assembler.

2. Link BSM.OBJ, the appropriate <Media>.OBJ, and the <HSM>.OBJ using Borland's TLINK.EXE, or a compatible linker, to create a file called RBOOT.ROM. For example, the appropriate command to link a Token-Ring Boot ROM using TLINK.EXE would be:

   ```
   TLINK BSM.obj BSMTOKEN.obj HSM.obj,RBOOT.rom /t
   ```

   Note: BSM.OBJ must be the first object file to be linked.

   You can also create RBOOT.ROM using the Microsoft Linker:

   ```
   LINK BSM.obj BSMTOKEN.obj HSM.obj,BSM.exe,,,,
   EXE2BIN BSM.exe RBOOT.rom
   ```

3. Run the ROMSUM utility on the newly created RBOOT.ROM. This utility adjusts the size of RBOOT.ROM to the nearest 8k boundary, unless you enter a *Size* parameter on the command line. The *Size* parameter specifies the number of 512 byte pages you would like RBOOT.ROM to encompass. For example, to use ROMSUM to size RBOOT.ROM to 16 512-byte pages, enter:

   ```
   ROMSUM 16
   ```

   This specifies an 8k (8192) byte ROM (16 x 512 byte pages). Running the utility with no *Size* parameters entered on the command line,

   ```
   ROMSUM
   ```

   creates an RBOOT.ROM file adjusted to the nearest 8k (8192) byte boundary.

4. Test the Boot ROM's functionality by using TESTROM.COM. Use TESTROM.COM from a DOS prompt without loading the network. You can also load a debugger from the DOS prompt, then load TESTROM.COM which then executes the ROM file.

5. Burn the Boot ROM into a PROM.

Assumptions about the HSM

The HSM you use must conform to the following rules:

- The HSM must conform to the Novell ODI specification for using the MSM tool kit. (See reference *Open Data-Link Interface Developer's Guide for NetWare DOS Workstation Drivers*.)

- The HSM must be capable of sending and receiving 802.2 frames.

- The HSM must not chain or share hardware interrupts.

- The HSM must not do segment fix ups.

- The HSM must not intercept software interrupt vectors or the timer interrupt (08h or 1Ch).

- The HSM can issue the following DOS INT 21h function codes:

  AH = 02h ==> Print char in DL
  AH = 09h ==> Print '$' terminated string in DS:DX.
  AH = 25h ==> Write DS:DX to Hardware Interrupt Vector.
  AH = 35h ==> Read Hardware Interrupt Vector into ES:BX.

Caution: No other DOS function codes are allowed. Using other DOS function codes will halt RPL.

Theory of Operation

The Boot ROM Developer's Kit contains reduced-functionality modules of the LSL and MSM, the Find/Found Frame RPL Protocol Stack, and a loader application.

Booting a Diskless Workstation with RPL

Quick Reference

The following table overviews the basic events involved in using RPL to boot a diskless workstation. These events are explained in more detail following the table.

Table 1. RPL Basic Events

Actor/Agent	Action
BIOS	1. Searches for the 55Ah signature.
Boot ROM	2. Hooks Interrupt Vector 18h.
BIOS	3. Calls Loader Application by using Interrupt Vector 18h.
Loader Application	4. Copies the image of the boot file. 5. Executes the Boot Strap Program. 6. Transfers control to the Boot Strap program.
Boot Strap Program	7. Registers as a protocol stack with the LSL in the Boot ROM. 8. Begins to send and receive packets to download a DOS, OS/2, UNIX or any other operating system image that might be required.

Description of Procedure

When the computer is turned on, it performs the POST (Power On Self Test) operation: the BIOS scans memory between C0000h and EE000h in 2k (2048) byte increments, looking for a 55AAh signature that denotes the presence of a ROM. If the signature is found, the BIOS uses the third byte of the ROM (containing the number of 512 byte pages) to perform a checksum on the ROM. If the checksum results in 0, the BIOS issues a far call to offset 03 (the fourth byte) of the ROM.

In the case of a Boot ROM, the code located at offset 03h hooks Interrupt Vector 18h (the BASIC interrupt) to point to the ROM's relocate routine and returns to the BIOS. Later, when the BIOS has determined that there is no other bootable device, the BIOS will perform an INT 18h instruction.

The INT 18h executes the relocate code, which copies the ROM into RAM just below 32k from the top of memory as reported by INT 12h.

The ROM sets the hardware options in the driver's configuration table according to the ROM location and the information in the ROMMAP.INC file. The ROM then executes the initialization routines of the LSL, MSM, HSM, and the Find/Found Frame RPL Protocol Stack.

Figure 1 is a block diagram illustrating the flow of operations after the BIOS calls the Loader Application. Figure 2 provides the flow diagram for the Find/Found Frame RPL protocol. The following sections describe these flows in detail.

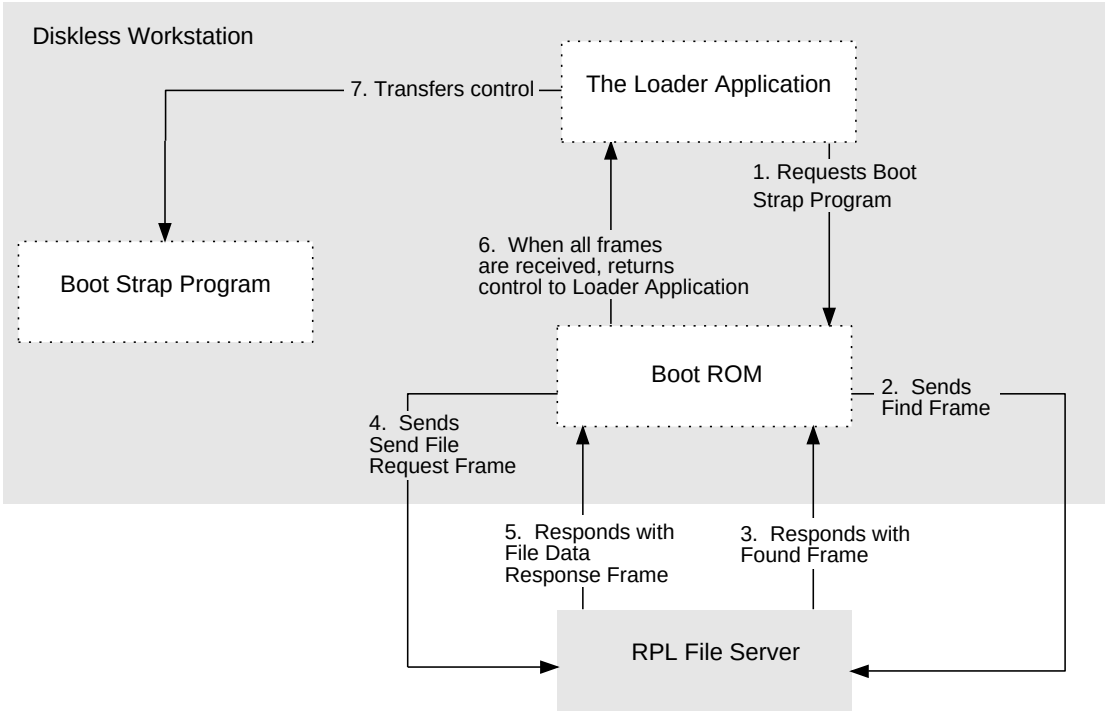

Figure 1. Flow of Operations

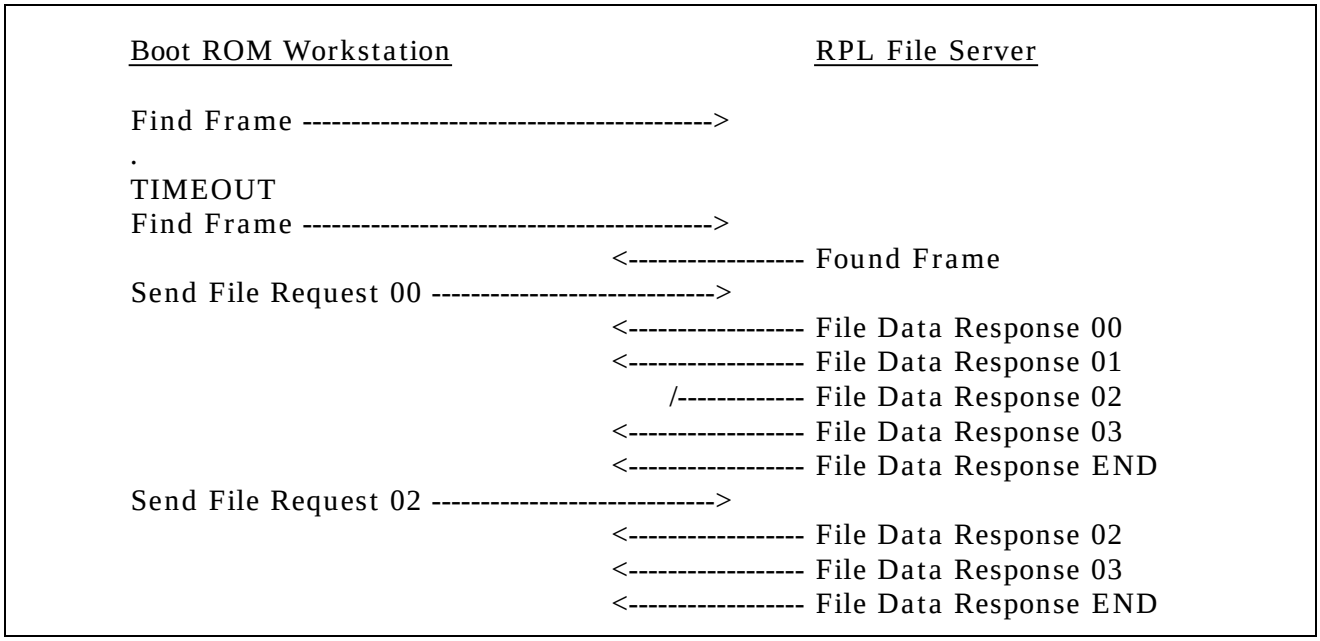

Figure 2. Find/Found Frame RPL Protocol Flow Diagram

Locating An RPL File Server

The Find Frame

The Boot ROM attempts to locate an RPL file server by transmitting a Find Frame using the multicast address 03 00 02 00 00 00h. The Find Frame format is given in Figure 3.

> **Note:** The protocol always requires that values be present in High-Low format. The Boot ROM must swap the values when running on an INTEL processor.

Offset (decimal)	Length	Value	Description
00	02	0053h	Frame Length
02	02	0001h	FIND Command
04	04	0008 4003h	Correlator Vector
08	04	0000 0000h	Correlator Value
12	04	0010 0008h	Connect Info Vector
16	04	0006 4009h	Frame Size Sub-Vector
20	02	Max Frame	Max Frame from Driver Config Table
22	04	0006 400Ah	Connect Class Sub-Vector
26	02	0001h	Class I ONLY
28	04	000A 4006h	Address Vector
32	06	Ring Address	Ring address of this Adapter in Media Format
38	04	0005 4007h	Logical SAP Vector
42	01	FCh	Remote SAP Value
43	04	0028 0004h	Search Vector
47	04	0024 C005h	Loader info Sub-Vector
51	08	Configuration	Configuration obtained by issuing INT 15h
59	02	Equipment	Register AX from INT 11h
61	02	Memory Size	Register AX from INT 12h MINUS 32k MINUS the Boot ROM Size
63	02	Version	Major, Minor Version of BSM.obj
65	06	000000000000	Rest of RPL EC
71	02	5342h	Adapter ID
73	10	Short Name	The HSM Short Name from the Driver Config Table

Figure 3. Format of the Find Frame

Figure 4 illustrates a hexadecimal trace of an Ethernet board sending the Find Frame.

```
0000  03 00 02 00 00 00 00 00  1B 04 1A 65 00 56 FC FC
0010  03 00 53 00 01 00 08 40  03 00 00 00 00 00 10 00
0020  08 00 06 40 09 05 EA 00  06 40 0A 00 01 00 0A 40
0030  06 00 00 1B 04 1A 65 00  05 40 07 FC 00 28 00 04
0040  00 24 C0 05 08 00 F8 04  04 F6 74 00 42 21 02 57
0050  01 00 00 00 00 00 00 00  53 42 4E 42 00 00 00 00
0060  00 00 00 00
```

Figure 4. Hexadecimal Trace of a Find Frame from an Ethernet Board

The Find Frame has the following features:

- The *Connection Class* field at offset 26h is set to accept Class I frames only. The Boot ROM is not capable of accepting File Data Response Frames that are broadcast to a Group or Functional Address.

- The *Memory Size* field at offset 61h is set to:

  <RAM_memory_size> – 32k bytes – <boot_ROM_size>

  Because the Boot ROM is relocated from ROM to RAM, the boot strap should not use this memory. 32k bytes is subtracted from the memory size to allow room for the transient portion of DOS.

- The first two bytes of the *Remote Program Load EC* (Engineering Change) field at offset 63h are set to the Major and Minor Version number of BSM.OBJ. The rest of the 8 byte field is set to 00h.

- The *Adapter ID* field at offset 71h is set to 5342h for all adapters. This informs the RPL server to use a generic boot strap.

- If the RPL file server must know the type of adapter, the *Adapter EC* field at offset 73h of the Find and the Send File Request Frames contain the *HSM Short Name* field, taken from the Driver Configuration Table at offset 50h.

The Found Frame

The RPL file server should respond to the Find Frame with a Found Frame. See Figure 5 for the format of the Found Frame.

Offset (decimal)	Length	Value	Description
00	02	003Ah	Frame Length
02	02	0002h	FOUND Frame
04	04	0008 4003h	Correlator Vector
08	04	0000 0000h	Correlator Value
12	04	0005 400Bh	Response Correlator
16	01	00	Response Code
17	04	000A 400Ch	Set Address Vector
21	06	0000 0000 0000	Group Address NOT Supported
27	04	000A 4006h	Loader Address Vector
31	06	Node Addr	RPL Server Node Address
37	04	0010 0008	Connect Info Vector
41	04	0006 4009h	Frame Size Sub-Vector
45	02	Max Frame	Maximum Frame Size
47	04	0006 400Ah	Connect Class Sub-Vector
51	02	0001	Connection Class
53	04	0005 4007h	Loader SAP Vector
57	01	RSAP	SAP Value of the RPL Server

Figure 5. Format of the Found Frame

Figure 6 contains a hexadecimal trace of an Ethernet board sending the Found Frame.

```
0000  00 00 1B 04 1A 65 00 00  1B 24 58 8F 00 3D FC FC
0010  03 00 3A 00 02 00 08 40  03 4E 65 74 57 00 05 40
0020  0B 00 00 0A 40 0C 00 00  00 00 00 00 00 0A 40 06
0030  00 00 D8 24 1A F1 00 10  00 08 00 06 40 09 05 EA
0040  00 06 40 0A 00 01 00 05  40 07 FC
```

Figure 6. Hexadecimal Trace of a Found Frame from an Ethernet Board

The Send File Request Frame

After receiving the Found Frame, the Boot ROM transmits the Send File Request Frame to download the Boot Strap Program. See Figure 7 for the format of the Send File Request Frame.

Offset (decimal)	Length	Value	Description
00	02	0053h	Frame Length
02	02	0010h	Send File Request Command
04	04	0008 4003h	Correlator Vector
08	04	0000 0000h	Correlator Value
12	04	0010 0008h	Connect Info Vector
16	04	0006 4009h	Frame Size Sub-Vector
20	02	Max Frame	Max Frame from Driver Config Table
22	04	0006 400Ah	Connect Class Sub-Vector
26	02	0001h	Class I ONLY
28	04	000A 4006h	Address Vector
32	06	Ring Address	Ring address of this Adapter in Media Format
38	04	0005 4007h	Logical SAP Vector
42	01	FCh	Remote SAP Value
43	04	0028 0004h	Search Vector
47	04	0024 C005h	Loader info Sub-Vector
51	08	Configuration	Configuration obtained by issuing INT 15h
59	02	Equipment	Register AX from INT 11h
61	02	Memory Size	Register AX from INT 12h MINUS 32k MINUS the Boot ROM Size
63	02	Version	Major, Minor Version of BSM.obj
65	06	000000000000	Rest of RPL EC
71	02	5342h	Adapter ID
73	10	Short Name	The HSM Short Name from the Driver Config Table

Figure 7. Format of the Send File Request Frame

Figure 8 contains a hexadecimal trace of an Ethernet board transmitting the Send File Request Frame.

The File Data Response Frame

The RPL file server responds to the Send File Request Frame with a File Data Response Frame. This frame contains a copy of the Boot Strap Program to be sent to the Loader Application. See Figure 9 for the format of the File Data Response Frame.

```
0000  00 00 1B 24 58 8F 00 00  1B 04 1A 65 00 56 FC FC
0010  03 00 53 00 10 00 08 40  03 00 00 00 00 00 10 00
0020  08 00 06 40 09 05 EA 00  06 40 0A 00 01 00 0A 40
0030  06 00 00 1B 04 1A 65 00  05 40 07 FC 00 28 00 04
0040  00 24 C0 05 08 00 F8 04  04 F6 74 00 42 21 02 57
0050  01 00 00 00 00 00 00 00  53 42 4E 42 00 00 00 00
0060  00 00 00 00
```

Figure 8. Hexadecimal Trace of a Send File Request Frame from an Ethernet Board

Figure 10 contains a hexadecimal trace of an Ethernet board sending the File Data Response Frame.

Offset (decimal)	Length	Value	Description
00	02	0019h+nn	Frame Length nn = File Data Length
02	02	0020h	File Data Response Frame
04	04	0008 4011h	Sequence Header
08	04	0000 nnnnh	Sequence Number
12	04	000D C014h	Loader Header
16	04	Locate Addr	Address of Data
20	04	XFER Addr	Transfer Control Address
24	01	Flags	Bit Significant Option Flag
25	02	0004h+nn	File Data Vector Length
27	02	4018h	File Data Vector
29	nn	File Data	Binary File Data
.			
.			
.			

Figure 9. Format of the File Data Response Frame

The Boot ROM Interfaces

The Boot ROM LSL Interface

An LSL interface is built into BSM.OBJ to provide support routines for both the HSM and the protocol stacks, including the Find/Found RPL protocol and the protocol stack used by the Boot Strap Program. This implementation of the LSL and the MSM has been optimized in BSM.OBJ to create as small a

Boot ROM as possible. The BSM.OBJ LSL actually contains 2 interfaces (illustrated in Figure 11):

- The Boot ROM LSL MLID Support Interface (a version of the ODI LSL with reduced functionality)
- The Boot ROM LSL Protocol Support Interface

```
0000  00 00 1B 04 1A 65 00 00  1B 24 58 8F 05 DC FC FC
0010  03 05 D9 00 20 00 08 40  11 00 00 00 00 00 0D C0
0020  14 00 08 C2 B0 00 08 C2  B0 20 05 C0 40 18 File
0030  Data ...
```

Figure 10. Hexadecimal Trace of a File Data Response Frame from an Ethernet Board

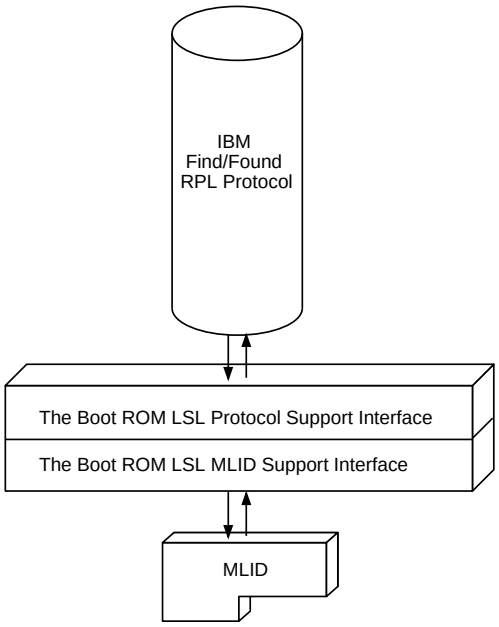

Figure 11. Block Diagram of Boot ROM Interfaces

This section describes the changes and enhancements to the LSL provided by BSM.OBJ.

The Boot ROM LSL MLID Support Interface

The Boot ROM LSL MLID Support Interface is functionally equivalent to the interface described in the *Open Data-Link Interface Developer's Guide for NetWare DOS Workstation Drivers, Volume III* (part number: 107-000010-001). Table 2 contains the exceptions to the standard LSL functionality.

Table 2. Exceptions to Standard LSL Functionality		
BL Register Equals:	**Function**	**Difference**
06h	***DeRegisterMLID***	This function is NO-OPed.
15h	***AddProtocolID***	This function will return BAD COMMAND in register *AX*.
16h	***GetStackECB***	This function will return BAD COMMAND in register *AX*. To get an ECB to fill in, the HSM should use ***MSMGetRCB.***

In addition, BSM.OBJ provides the HSM IOCTL interface with the limitations outlined in Table 3.

Table 3. Limitations of the HSM IOCTL Interface		
BL Register Equals:	**Function**	**Difference**
02h	***AddMulticastAddress***	Only one multicast address will be active at any given time.
03h	***DeleteMulticastAddress***	BSM.OBJ assumes only one multicast active at a time.
09h	***SetLookAheadSize***	The *LookAhead* size is set to 128 bytes during initialization and is not changed afterwards.

The Boot ROM LSL Protocol Support Interface

The Find/Found Frame RPL protocol stack and the Protocol Stack implemented by the Boot Strap Program use the LSL Protocol Support Interface, described in detail in the *Open Data-Link Interface Developer's Guide for DOS Workstation Protocol Stacks, Volume III*. This section describes the changes implemented by BSM.OBJ.

Table 4. Changes to the LSL Protocol Support Interface			
BL Register Equals:	**Function**	**Difference**	
06h	***RegisterStack***	This function requires the following input:	
		AX = Protocol ID	For 802.2, the Protocol ID is the Destination SAP value in AH. AL should be equal to 00. For SNAP or EII, the Protocol ID is the two byte value used in the media header.
		BH = Board Number	BH = 00 ==> 802.2 board BH = 01 ==> 802.2 SNAP board BH = 02h ==> Ethernet_II board
		ES:SI = The protocol stack receive handler	
		The protocol stack should be ready to receive packets when this function is called. This function calls ***BindStack*** automatically.	
08h	***RegisterDefaultStack***	ES:SI must point to the Default Stack Receive Handler.	
10h	***RegisterPrescanStack***	ES:SI must point to the Prescan Stack Receive Handler.	

Table 5 contains the functions not implemented in the Boot ROM LSL. If any of these functions are called, register *AX* will return BAD_COMMAND (8009h).

Table 5. Functions Not Implemented in the Boot ROM LSL	
BL Register Equals:	**Function**
16h	***GetStackIDFromName***
17h	***GetPIDFromStackIDBoard***
19h	***GetProtocolControlEntry***
20h	***GetLSLStatistics***
21h	***BindStack*** (Use ***LSLRegisterStack***)
22h	***UnbindStack*** (Use ***LSLDeRegisterStack***)
23h	***AddProtocolID*** (Use ***LSLRegisterStack***)

Table 5. Functions Not Implemented in the Boot ROM LSL	
BL Register Equals:	**Function**
25h	***GetLSLConfiguration***
26h	***GetTickMarker***

In addition, the Boot ROM LSL does not provide the General Services Interface. If the General Services entry point is called, register AX returns BAD_COMMAND (8009h).

The Boot Strap Program

Boot Strap Program Machine State

Once the RPL file server has been found, it sends a copy of the Boot Strap Program to the Loader Application which then copies the program into RAM and transfers control to that program. Table 6 contains the state of the registers in effect when the Boot Strap Program gains control.

Table 6. Registers in Effect During the Boot Strap Program	
Register	**Definition**
DS:AX	LSL Protocol Support Entry Point. Use these registers to obtain the LSL Support routines in the Boot ROM.
DS:BX	A pointer to the *This Ring Only* variable. Source Routing uses this one byte variable to control the number of Spanning Tree Explorer frames sent with broadcast frames. Immediately following this variable is the *This Ring Only Initial Value.* The Boot Strap Program might wish to modify these variables to control Source Routing.
DS:DX	A pointer to the adapter's six byte Node Address in the HSM's Driver Configuration Table. The Node Address is always stored in canonical form by the Boot ROM. A canonical Node Address means that the *Broadcast* bit is the least significant bit of the first byte. Token-Ring and FDDI present the Node Address in Non-canonical form and, therefore, must perform a bit swap. The Boot Strap Program can use this pointer to access other fields of the Driver Configuration Table, or it may use the *LSL Get MLID Config Table* interface to access the Driver Configuration Table. The *Media ID* field at offset 60h of the Driver Configuration Table will be set to 03h for Ethernet, 04h for Token-Ring, and 20h for FDDI.
DS:SI	A pointer to the RPL File Server's six byte Node Address, also stored in canonical form. The thirty byte *Source Routing* field used to access the RPL file server immediately follows these six bytes. For Ethernet adapters, the *Source Routing* field will be thirty bytes of 0h.

Enabling the Boot Strap Program to Use the HSM

When the Boot Strap Program gets control, it is ready to download an appropriate Disk Image file from the RPL file server. To continue to use the HSM driver in the Boot ROM code, the Boot Strap Program must:

1. Save the address of the LSL Protocol Support Entry Point in a local variable.

2. Initialize all necessary control blocks and work area variables with the adapter's Node Address, the RPL file server Node Address, and any necessary information from the Driver Configuration Table.

3. Call the LSL Protocol Support Entry Point to register an appropriate protocol stack. See Table 7.

Table 7. Status of Registers when Calling a Protocol Stack

Register	Value	Description
AX	Protocol ID	For 802.2 protocols, the Protocol ID is the Destination SAP in register *AH*, with *AL* equal to 0.
BH	Board Number	00h = 802.2 01h = 802.2 SNAP 02h = Ethernet_II
BL	06h	***RegisterStack***
ES:SI	Receive Handler	Boot Strap Program Protocol Receive Handler Entry Point
Call	dword ptr LSL Protocol Support Entry Point	The address of this entry point is given by the Boot ROM program in registers *DS:AX*

The Boot Strap Program is now ready to send and receive data using the HSM in the Boot ROM.

The RPL Display Screen

When the Boot ROM gets control through INT 18, it will clear the video display and present pertinent messages as the information becomes available. The Boot ROM's sign-on message is displayed first, immediately followed by the HSM driver's sign-on message.

The sign-on messages are followed by any messages the HSM driver displays using the ***MSMPrintStringZero*** function. These messages are followed by the HSM driver's configuration information. The configuration information is prefixed by `RPL-ROM-iii` (*iii* is a three character ID signifying

the type of information). Figure 12 illustrates an example display screen.

```
Novell RPL BootROM  v1.00  (920117)
Novell NE2 Ethernet MLID  v1.21 (911104)
(C) Copyright 1991 Novell, Inc.  All Rights Reserved.

RPL-ROM-ADR: 0000 1B24 588F
RPL-ROM-IRQ: 3
RPL-ROM-MM1: C800
RPL-ROM-PIO: 1000
RPL-ROM-SLT: 2

RPL-ROM-FFC: 1
RPL-ROM-SFC: 1
RPL-ROM-SEQ: 3
```

Figure 12. An Example RPL Display Screen

Table 8 describes each three-character ID.

Table 8. Three-Character IDs

Character ID	Format	Description
RPL-ROM-ADR	xxxx xxxx xxxx	The six byte Node Address of the installed adapter. It is a hexadecimal field displayed in media format. For example, it is canonical for Ethernet, and non-canonical for FDDI and Token-Ring. Its value is taken from the Driver Configuration Table.
RPL-ROM-DMA	nn	A one byte decimal field signifying the DMA channel used by the driver. Its value is taken from the Driver Configuration Table. This message is only displayed if the driver uses DMA.

Table 8. Three-Character IDs

Character ID	Format	Description
RPL-ROM-ERR	BADA; RPL Halted	The prefix of a FATAL error. This message always ends with "RPL Halted." The displayed message comes either from the driver (through the ***MSMPrintStringFatal*** function) or from the Boot ROM. If the message came from the driver, it contains the ASCII text of the message. If the message came from the Boot ROM, it contains a two byte hexadecimal number signifying the type of error. The two possible error types are: *RPL-ROM-ERR: BADA; RPL Halted* This message is displayed if the RPL server sent a File Data Response frame with an invalid Locate or Transfer address. *RPL-ROM-ERR: DExx; RPL Halted* This message is displayed if the HSM driver issued an invalid DOS function code to INT 21. *xx* is the hexadecimal value of the offending function code.
RPL-ROM-FFC	nnnn	A decimal field signifying the number of Find Frames sent by the Boot ROM. An excessive Find Frame count indicates that the RPL server either is not present or is congested.
RPL-ROM-HSM	message. . .	The prefix given to a driver-generated message.
RPL-ROM-IRQ	nn	A one byte decimal field signifying the Interrupt level used by the driver. Its value is taken from the Driver Configuration Table. It is only displayed if the driver uses interrupts.
RPL-ROM-MM1	xxxx	A two byte hexadecimal field containing the segment value used by the driver for Memory Address 1. Its value is taken from the Driver Configuration Table. It is only displayed if the driver uses Memory Address 1.
RPL-ROM-MM2	xxxx	A two byte hexadecimal field containing the segment value used by the driver for Memory Address 2. Its value is taken from the Driver Configuration Table. It is only displayed if the driver uses Memory Address 2.
RPL-ROM-PIO	xxxx	A two byte hexadecimal field containing the Programmed I/O (PIO) address used by the driver. Its value is taken from the Driver Configuration Table. It is only displayed if the driver uses programmed I/O.

Table 8. Three-Character IDs		
Character ID	**Format**	**Description**
RPL-ROM-SEQ	nnnn	A decimal field containing the number specifying the last valid sequence number received.
RPL-ROM-SFC	nnnn	A decimal field containing the number of Send File Request Frames sent by the Boot ROM. An excessive Send File Request count indicates that the RPL Server is not responding after it has been found.
RPL-ROM-SLT	xxxx	A decimal field containing the Micro-Channel or EISA Slot Number used by the driver. Its value is taken from the Driver Configuration Table. It is only displayed if the driver specifies a Slot number.

Index

B

Boot ROM
interfaces used in 11
Boot ROM Developers Kit, contents of.
See Contents of Boot ROM Developer's Kit.
Boot ROM, interfaces used in
Boot ROM LSL MLID Support Interface 11
Boot ROM LSL Protocol Support Interface 12
HSM IOCTL interface 13
Boot Strap Program
Downloading. *See* Downloading Boot Strap Program.
Machine state. *See* Machine state, Boot Strap Program.
Using HSM. *See* HSM, enabling Boot Strap Program to use.
Booting diskless workstation
Theory of operations 5

C

Calling a protocol stack
status of registers when calling stack 16
Calling General Services entry point 15
Contents of Boot ROM Developer's Kit
object files included 1
source files included 1
utilities included 2
Creating a Boot ROM
process of 1

D

Disk Image File, downloading.
See Downloading Disk Image File.
Diskless workstation, booting
theory of operations 5
Display screen.
See RPL display screen.
Downloading
Boot Strap Program 11
See also File Data Response Frame; Send File Request Frame.
Disk Image File 16

F

File Data Response Frame
format of 11
hexadecimal trace of from Ethernet board 12
Find Frame
features of 7
format of 8
hexadecimal trace of from Ethernet board 8
Find/Found Frame RPL Protocol Flow
diagram of 7
Found Frame
format of 9, 10
Frame Format
File Data Response Frame 11
Find Frame 8
Found Frame 9
Send File Request Frame 10

G

General Services entry point, calling 15

H

HSM
assumptions which HSM must conform to 4
enabling Boot Strap program to use 16
HSM IOCTL interface
limitations of Boot ROM implementation 13

I

Interfaces used in Boot ROM.
See Boot ROM, interfaces used in.

L

Locating RPL file server 7
See also Operation, block diagram of; Find Frame; Found Frame.
LSL
Boot ROM LSL MLID Support Interface 11
changes to the LSL Protocol Support Interface 14
exceptions to standard LSL functionality 13
functions not implemented in Boot ROM LSL 13
implementation of in Boot ROM 11
LSL MLID Support Interface.
See LSL, implementation of in Boot ROM.
LSL Protocol Support Interface.
See LSL, implementation of in Boot ROM

M

Machine state
  Boot Strap Program 15
  register state of 15

O

Operations
  block diagram of 6
  chart of events 5
  theory of 4

P

Protocol used by Boot ROM 1

R

Registers, state of.
  See Machine State Boot Strap Program; Calling a Protocol Stack.
RPL display screen
  sign-on messages displayed 16
  three character ID of sign-on messages 17
RPL file server, locating.
  See Locating RPL file server.
RPL Protocol
  diagram of Find/Found Frame RPL Protocol Flow 6

S

Send File Request Frame
  format of 10
  hexadecimal trace of from Ethernet board 11
Sign-on messages
  display on RPL screen 16
  three character ID of 17

T

Three character ID
  sign-on messages 17

W

Workstation, booting diskless
  theory of operations 5