DNSG0.WS4   (= "DR Net System Guide", foreword)

DR Net System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Foreword

DR Net is a network operating system for Concurrent CP/M and CP/M-86 based computers that allows local disk drives, list devices, and queues to be mapped to remote computers. Network transactions are conducted on a system call basis. DR Net traps standard system calls that reference a mapped resource, oversees execution of the function on the remote node, and returns the response to the calling process. This entire sequence of events is controlled entirely by DR Net. No modifications to an application program or the resident operating system are necessary to integrate DR Net into computers based on the Concurrent CP/M and CP/M-86 operating systems.

Like all Digital Research operating systems, DR Net is composed of a proprietary module and an implementation-dependent module. The proprietary module contains the interface to transient prgframs, to the host operating system, and to the implementation-dependent Network Input/Output System (NIOS). The NIOS contains the interface to the network communication hardware.

How to use this manual

This manual provides you with the information necessary to develop a DR Net module for your computer system and network controller. This is a technical presentation that requires you to have a thorough understanding of the CP/M-86 and Concurrent CP/M system calls and their calling conventions. In addition, it is assumed that you have an understanding of the basic principles of data communications.

Section 1, "DR Net overview", introduces the DR Net network operating system. This section contains general information regarding the types of network nodes and operating system supported by DR Net. It also includes a synopsis of the DR Net utilities, and a description of DR Net's computer hardware requirements.

Section 2, "DR Net architecture", describes the modules and processes that provide the network interface. It focuses on the relationship between DR Net's proprietary portion and the implementation-dependent NIOS. Refer to this section for descriptions of significant DR Net internal data structures.

Section 3, "The NIOS", describes the structure and the functions of the implementation-dependent portion of DR Net.

Section 4, "DR Net system generation", describes how to generate the DR Net system image, and how to integrate it with a Concurrent CP/M or CP/M-86 host.

Appendix A describes the DR Net message contents for all system calls that can be executed remotely. Appendix B describes the fundamental considerations required to design a server facility in an operating system other than Concurrent CP/M.

Other DR Net manuals

This is one of four manuals in the DR Net documentation set. The other three are as follows:

- "DR Net User's Guide"
  Describes DR Net and the DR Net utilities for the end user.

- "DR Net Programmer's Guide"
  Describes the DR Net system calls and system call compatibility among the different operating systems supported by DR Net.

- "DR Net System Manager's Guide"
  Describes DR Net installation and maintenance for the technician in charge of a DR Net computer network.

DR Net is upwardly compatible with CP/NET Version 1.2, the network operating system for Digital Research's 8-bit, CP/M and MP/M II operating systems. For information on CP/NET Version 1.2, refer to the "CP/NET Reference Manual".

Conventions used in this manual

The following conventions and terminology are used in this manual to identify functions, programs, and special charactersitics:

- "NET" and "LD" are used as prefixes for the two classes of NIOS functions, as in "NET_WBOOT" and "LD_DRVR". The "NET" prefix indicates a NIOS global network function. The "LD" prefix indicate a line driver dependent function. In all cases, the names of all functions are in uppercase letters.

- DR Net utilities are always in uppercase only. After their first appearance, the CMD filetype is assumed.

- Frequent mention is made to logn and short pointers. A long pointer is always a double-word value, where the first word is the offset and the second is the segment address. Short pointers specify only an offset.

- "Network controller" refers to any hardware component used to control data I/O on and off the network lines.

- "Host operating system" refers to the resident operating system in the computer in which DR Net has been integrated.

- "Process families" refers to a group of processes linked through the parent pointer field in the process descriptor. Process families are significant in the network context, because they automatically share a network environment.

- "Attach" is used to indicate that a process is given a network environment that allows it to log on servers and use remote disk drives, list devices, and queues. Until a resource attaches the network, it only has access to local resources.

- "Detach" is the inverse of attach. It means that the process's network environment is removed. Until the process attaches again, it has access only to local resources.

Table of Contents

Figures

EOF

DNSG1.WS4   (= "DR Net System Guide", section 1)

DR Net
System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Section 1: DR Net overview

DR Net is a network operating system that allows a CP/M based computer to access another CP/M based computer's disk drives, list devices, and queues. Like all Digital Research operating systems, DR Net consists of two basic components: an invariant module and an implementation dependent module.

DR Net's invariant module serves two purposes. The first purpose it to intercept disk drive, list device, and queue related system calls, to determine if the resource referenced is local or remote. A call to a remote resource is assembled into a standard-format message, and sent to another node for execution.

The second purpose of invariant module is to execute the system calls from remote computers. For this task, application processes are dynamically created by DR Net on network nodes that function as servers. The purpose of these transient processes is to present the remote system call to the host operating system, and prepare the response message.

DR Net's implementation dependent module, the Network Input/Output System (NIOS), is the interface between the invariant module and the network hardware. The NIOS interface consists of a set of functions that are called from the invariant module to use the physical components.

These two modules are merged by a system generation program. This program also displays a series of prompts that set network global parameters and a node's map of networked resources. This creates a DR Net system image that is then integrated by another utility with the computer's host operating system to incorporate the networking capability. No modifications to the host operating system, nor application programs, are required to implement DR Net and have remote resources accesses as though they were local.

This section of the "DR Net System Guide" describes the external characteristics of a DR Net network. This overview includes the definitions of the network functional roles and the host operating system that suppor them, DR Net's hardware requirements, and the network parameters. DR Net's invariant module and implementation dependent NIOS are described in subsequent sections.

1.1 DR Net functional node types

DR Net defines two fundamental, network functional node types: requesters and servers.

Requesters initiate all network transactions. A transaction results when an application makes a system call to a disk drive, list device, or queue that is mapped to a server. Resource mapping, the process of defining a local reference for a remote resource, is performed at DR Net system generation. Subsequently, when the network is attached, DR Net references this map of local to remote resources for every system call. Calls to local resources are passed to the requester's host operating system. Calls to mapped disk drives, list devices, and queues are trapped, a DR Net message is constructed, and the message is sent to the designated server.

Servers wait and respond only when called upon. By definition, servers never send requesters unsolicited messages. Conceptually, the DR Net server can be thought of as a process manager that oversees concurrent application processes. When a requester logs on, the DR Net server creates an application process and assigns it to the calling requester. This "shadow" process is the requester's proxy that presents its system calls to the server's host operating system. The shadow remains assigned to a requester until it is no longer needed. At that point, the shadow is terminated, and the system resources that sustained it are freed for allocation to another shadow process.

Note: "Shadow process" is used throughout this manual to reference a server-based process whose function is to make system calls on behalf of a requester process. In all cases, a shadow process represents a single requester process. Multiple processes running on a single requester each have their own shadow process on a server. A shadow process does not exist until a requester logs on. In turn, when the requester process logs off, the shadow process is terminated.

Figure 1-1 illustrates DR Net's two network roles, and shows how the requester and server functions interface with their host operating systems. Table 1-1 summarizes a DR Net requester to server dialogue.

```
   DR Net Requester                          DR Net Server
    +-----------+                            +-----------+
    |   Host    |                            |   Host    |
    | Operating |                            | Operating |
    |  System   |      Shadow process        |  System   |
    +-++--------+      presents requester's  +-----------+
      ||  /\           call to its host          /\
      ||  ||           operating system ---------->||
      ||  ||                                     \/
      || +-++-----+                          +--------+
      || | DR Net |    +------+      +------+    | DR Net |
      || | screens|/--\| NIOS |---> <---| NIOS |/--\| shadow |
      || | system |\--/+------+<--- --->+------+\--/| process|
      || | calls. |        |   |         +--------+
      || +-++-----+        |   |
      ||  ||  /\           +---+--> Physical Networks
      ||---||--||----> Returned Values (Down Arrows)
```

```
  \/  \/ ||----> System calls from application
+----------++-+
| Application |
|  Programs   |
+-------------+
```

Figure 1-1. Network requester and server functional roles

Table 1-1. DR Net transaction dialogue

Requester	Server
Intercept an application's system calls before they reach the host operating system.	Idle in expectation of a message.
Screen each call, to determine if it references a mapped disk drive, list device, or queue.	
Send all calls that reference a local resource to the host; for all calls that reference a mapped resource, make the DR Net logical message.	
Pass the message to the NIOS.	
NIOS sends message.	NIOS receives message.
Idle in expectation of the response message.	Signal shadow process to pick up message from NIOS and decode it.
	Present system call to the host operating system.
	Encode the response DR Net logical message from the returned value and all related data.
	Pass the message to the NIOS.
NIOS receives message.	NIOS sends message.
Pick up the message from the NIOS.	Idle in expectation of another request.
Decode the return value, set the registers, and	

store the response data
according to local
convention.

Return control to the
calling program.

1.1.1 Operating systems supported

The features available from Digital Research operating systems support specific types of network nodes. Table 1-2 lists the operating systems, and indicates the node types supported by each. Figure 1-2 below illustrates the possible communication links.

Table 1-2. Functional type support by operating system

Operating System	Network Functional Role
CP/M-86	Requester only
Concurrent CP/M	Simultaneous requester and/or server
MP/M II	Server only
CP/M 2.2	Requester only
CP/NOS	Diskless requester only
Other	Requester or server engineered to DR Net message format

As Table 1-2 shows, a CP/M-86 based system can be used only as a requester. However, Concurrent CP/M based nodes can function as a network server, a network requester, or as a combination network requester and server. Figure 1-2 below illustrates the options listed in this table, and the possible relationships between 8086 and 8088 based nodes and CP/NET Version 1.2 based nodes.

In addition to supporting the Digital Research family of operating systems, other operating systems can be engineered to support and interpret DR Net messages. For this purpose, all DR Net message formats and contents are published in Appendix A. In addition, Appendix B offers some observations and recommendations for creating a server for a different environment.

```
+-----------------------------------------------------+
|             8086- and 8088-based systems            |
|                                                     |
|      +-----------+          +-----------------+     |
|      |  CP/M-86  |          | Concurrent CP/M |     |
|  +-+--| Requester |       +--|    Requester    |--+  |
|  | |  +-----------+       |  +-----------------+  |  |
|  | |                      |                     |  |
|  | |  +-----------------+<-+  +-----------------+  |  |
|  | +->| Concurrent CP/M |<----| Concurrent CP/M |<-+  |
|  |    |     Server      |<-+  | Server/Requester|<-+  |
|  |    +-----------------+  |  +-----------------+  |  |
+--+-----------------------+----------------------+--+
  |                    |                   |
+--+------------------------+-----------------------+--+
| |                    |                   | |
| |   +---------+        |  +-------------+     | |
|  +--->| MP/M II |        +--| CP/M 2.2 or |     | |
|      | Server  |<------------|   CP/NIOS   |     | |
|      +---------+           |  Requester  |--->--+  |
|                         +-------------+        |
|                                            |
|      8080-, 8085-, and Z-80-based systems           |
+-------------------------------------------------------+
```

(Arrows indicate who can send a request to whom.)

Figure 1-2. Operating system interactions supported by DR Net

Figure 1-2 illustrates the extent to which DR Net and CP/NET Version 1.2 based nodes can communicate with each other. The delimiting factors for dialogues between CP/M-86 based requesters and MP/M II based servers is that the former is limited to DR Net functions 64 through 71 and BDOS functions 0 through 43, 45, and 106. (The DR Net functions are introduced in Section 2 in the description of the NDOS.) Communication between CP/M 2.2 and CP/NOS based requesters and Concurrent CP/M based servers are not restricted, because Concurrent CP/M supports all CP/M BDOS and CP/NET Version 1.2 functions.

1.1.2 Hardware requirements

The minimal DR Net computer requires an 8086/8088 processor, one floppy disk drive, and the CP/M-86 or Concurrent CP/M operating system. The amount of memory required depends upon the computer's functional role, the number of input and output network lines, and the number of concurrent, network processes to be accommodated. To determine your total memory requirements, first determine how much memory is required for your computer's operating system. DR Net's requirements are then added to this base value.

DR Net' memory needs are derived by adding the size of the DR Net module used, the DR Net Static Buffer, the DR Net Dynamic Buffer, and the NIOS.

DR Net module

DR Net's system generation program selects one of following three modules when it creates the DR Net system image: a requester only, a server only, or a simultaenous requester/server. Table 1-3 lists the size of each module. This value does not include your NIOS or buffer space required by DR Net for messages and process environments. A good rule of thumb is to add 2 KB to these amounts for your assembled NIOS. Descriptions of the buffer space requirements follow.

Table 1-3. DR Net modules sizes

DR Net Module	Description	Memory Requirement
RNET.CMD	requester only	22K bytes
SNET.CMD	server only	15K bytes
RSNET.CMD	simultaneous requester/server	26K bytes

Static Buffer

The DR Net Static Buffer is used to hold process and network related data during operation. DR Net determines the size of the Static Buffer by adding the values shown in Table 1-4, and reserves the space internally. However, DR Net cannot have the Static Buffer longer than 64 KB (65,536 bytes). This can become an important consideration when selecting the number of server and requester processes that can be supported on a network node.

Table 1-4. Static Buffer memory requirements

Size	Component
297h	per requester process
297h	per shadow process
284h	per input Line Driver Control Block
29Ah	per output Line Driver Control Block
1A8h	per copy of the Requester Configuration Table
1h	per Dynamic Allocation Unit
+ 282h	
= Total Static Buffer length	

The size values shown in Table 1-4 indicate the amount of memory required on a per component basis. For example, each requester process attached to the network requires 2D7h bytes; each shadow process requires 317h bytes; and each input Line Driver Control Block requires 304h bytes. Consequently, a requester/server node that supports four requester processes and eight shadow processes would require 65Ch bytes plus 18B8h bytes for just the processes alone.

Note: The Requester Configuration Table and Line Driver Control Blocks are described in Sections 2.3.2 and 2.3.4, respectively. Dynamic Buffer Allocation Units are described in Section 4.2.1 in the description of the GENNET system generation utility.

Dynamic Buffer

DR Net's Dynamic Buffer is used for the temporary storage of all messages. You specify the length of the Dynamic Buffer, and hence the number of messages that can be stored simultaneously, during DR Net system generation. How much memory to allocate for the Dynamic Buffer is implementation dependent, and can differ from one node to the newt within the same local area network. For the description of the Dynamic Buffer, Dynamic Buffer Allocation Units, and how to determine an appropriate Dynamic Buffer length for a given node, see Section 4.2.1.

1.2 Network and node limits

```
---------------------------
```

A DR Net network can have up to 255 individual nodes. Each node is uniquely identified by a hexadecimal number in the range 00h through 0FEh. There are no restrictions that affect the number of server versus requester nodes on a particular DR Net network. For example, one configuration can consist of 254 requesters and one server, while another can consist of 255 simultaneous requester/server.

Up to 90 simultaenous requester processes can be supported on a DR Net requester-only node when all other Static Buffer components have been minimized. Each requester process can be logged on to a maximum of 16 servers at a time. Note that DR Net logs on processes, rather than nodes, to servers. Consequently, a Concurrent CP/M user can log on 16 servers from one virtual console, change virtual console, and log on another 16 servers.

Every requester node has a default server. The default server is defined as the node accessed when the user does not enter a specific server node number or name. This is pertinent only in the use of the DR Net NET.CMD, LOGON.CMD, and LOGOFF.CMD utilities. The default server is also the location of the data file used by DR Net's name service. When name service is implemented, users can access nodes by name or number. This is currently pertinent only to DR Net utilities, but could also be used by independent DR Net application packages.

Each requester node can have as many as 16 local drives, list devices, and queues mapped to servers. A resource map, referred to as the Master Requester Configuration Table, is created at system generation time to define a default environment. When a user attaches his node to the network with the NETON.CMD utility, a copy is made of the Master Requester Configuration Table for use by the newly attached process. Subsequently, all system calls to a local resource that is mapped are trapped and sent to the designated server.

Changes are made to the Requester Configuration Table with the DR Net NET.CMD and LOCAL.CMD utilities. However, changes modify only the attached process's copy of the Requester Configuration Table, not the Master. The default configuration in the master reasserts itself when the user runs the NETOFF.CMD utility and subsequently runs NETON, or when the user restarts the computer. See Section 2.3.2 for more information on the Requester Configuration Table.

Up to 83 simultaneous shadow processes can be supported on a DR Net server-only node when all other Static Buffer components have been minimized. Do not confuse this value for the number of requesters that can share a server; a server can be shared by 254 separate requesters. However, only 83 requester processes can be logged on simultaneously.

Up to 16 drives on each server can be isolated from network access. The drive names are specified in a response to a prompt in the DR Net system generation program. Attempts to access this drive result in a BDOS select error.

1.3 DR Net utilities

DR Net's user interface consists of a set of utilities that attach and detach process families to the network in requester nodes, log on requester process families to a specific server node, log off requester process families from a specific server node, change the amp entries of local to remote resources, and display network status and node names.

Several system generation utilities are provided with the system. These programs generate a DR Net CMD file from the NIOS.CMD and a DR Net module, and create the name server data that allows operators to use DR Net's name service facility.

1.3.1 User utilities

Table 1-5 lists the DR Net utilities an operator would require for the day to day use of the network. For a complete description of these programs, see the "DR Net User's Guide".

Table 1-5. DR Net user utilities

Format: Name
      Description

LOGON.CMD	Log on a process family to a specific server.
LOGOFF.CMD	Log off a process family from a specific server.
NETLDR.CMD	CP/M-86 requester only; Load DR Net and attach to network.
NETON.CMD	Concurrent CP/M only; Attach process family to network.
NETOFF.CMD	Concurrent CP/M only; Detach a process family from network.
LOCAL.CMD	Remove a local resource mapping.
NET.CMD	Map a local resource to a server.
NETSTAT.CMD	Display node's resource map of local to remote resources, and current list of logged on servers.

NAMES.CMD
Display contents of the node's NAMSVR.DAT file (the names of all requester and server nodes).

1.3.2 System generation utilities

Table 1-6 lists the DR Net system generation utilities. These programs do not replace your Concurrent CP/M or CP/M-86 system generation procedures. They are used exclusively to generate the DR Net system image. All system generation utilities, except NAMESMOD.CMD, are described in Section 4. NAMESMOD is described in the "DR Net System Manager's Guide".

Table 1-6. DR Net system generation utilities

Format: Name
 Description

GENRQR.CMD
The program used to generate a DRNET.CMD file for use as CP/M-86 based requester.

GENNET.CMD
The program used to generate a DRNET.CMD file suitable for use on a Concurrent CP/M based computer as a requester, server, or simultaneous requester/server.

ADDNET.CMD
The program used to merge the DRNET.CMD file with your CCPM.SYS file and produce a new CCPM.SYS file with network capability.

NAMESMOD.CMD
The program used to generate the NAMSVR.DAT data file that provides name service for LOGON, LOGOFF, NET, NETSTAT, and NAMES programs.

EOF

DNSG2.WS4   (= "DR Net System Guide", section 2)

DR Net
System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Section 2: DR Net architecture

DR Net network transactions are conducted on a system call basis. In a Concurrent CP/M based computer, DR Net's invariant module creates a set of interdependent, indefinitely repeating processes that first turn a calling application's system calls into DR Net logical messages. Next, these processes conduct the messages through the network hardware to a remote operating system. There, the message is decoded, and the system call implemented. Finally, the processes build a DR Net logical message from the return values, and present the response to the calling application. Figure 2-1 illustrates the relationships of the NDOS and the repeating processes in a Concurrent CP/M requester/server node.

```
                         /\
                        /  \
                       /    \
+-------------+   +---<---< Watchdog >--->---+
| Application |--+ |      \     /           |
|   Process   | |  V       \   /            V
+-------------+  V /\       \/             / \
               /    \               /Shadow
              <  NDOS  >           < Process >
               \    /  \         ^  \  *  /
                ^ \/    \       /     \ /
+-------------+  | ^     \     /       |
| Application |--+ |      \   /        V
|   Process   |   /\       \/         /\
+-------------+  /  \      / \       /  \
            / Input  \   /     \   / Output \
           /  Message \ /       V /  Message \
           \  Routing  /         \  Routing  /
            \ Process/            \ Process/
             \ *  /                \ *  /
              \/                    \/
               ^         /\          |
               |        /  \         V
               +----<---<  NIOS  >----<----+
                        \  /
                         \/
```

* = There can be more than one of each of these processes

in a single DR Net node.

Figure 2-1. Concurrent DR Net processes

The medium of exchange between all processes, except the watchodg, is the DR Net logical message. Figure 2-2 illustrates its components; see Appendix A for the description of each field. The DR Net logical message is constructed by the NDOS and shadow processes. The input and output message routing processes pad the logical message with optional, un-initialized header and trailer bytes before passing it to the NIOS. However, the logical message is not affected when these empty fields are added. The length of the header and trailer are user-defined, and set at system generation time.

```
Field Length:
    1       1 or 2      1 or 2     1       1 or 2      1-65,536
+---------+-------------+--------+--------+-----------+-------------+
| Message | Destination | Source | System | Length of | Data Field: |
| Format  |   Node &    | Node & |  Call  | Data Field| All informa-|
|   Code  | Process ID  | Process| Number |           |tion required|
|         |   Number    | Number |        |           | to execute  |
|         |             |        |        |           | system call |
|         |             |        |        |           |  remotely.  |
+---------+-------------+--------+--------+-----------+-------------+

Number indicates length of field.
Where two lengths are shown, the first is for format 00 and 01
messages, and the second is for format 06 and 07 messages.
```

Figure 2-2. DR Net logical message contents

In CP/M-86 based computers, the DR Net module performs system call screening and message routing only between the application and the network hardware. The DR Net model in Figure 2-1 is applicable to the CP/M-86 based requester with two exceptions. First, separate processes do not perform the network tasks; all functions are performed by the NDOS. Second, there are no shadow processes in the CP/M-86 requester. (Shadow processes are found only in server nodes.)

Section 2.1 describes the functions associated with the NDOS module and the shadow, input message routing, and output message routing processes found in nodes running Concurrent CP/M. Section 2.2 follows with the description of the CP/M-86 requester. Each of these presentations includes an explanation of the DR Net initialization process. Section 2.3 completes the description of DR Net's architecture with an explanation of DR Net's internal data structures.

2.1 Concurrent CP/M nodes

The NDOS and indefinitely repeating processes shown in Figure 2-1 each depend on another process to contribute a specific task. The components shown in this figure are the full complement found in a simultaneous requester/server node. Requester-only nodes do not have the shadow process, and server-only nodes do not have the NDOS. The processes are created during system initialization by a routine called from the Concurrent CP/M initialization process.

2.1.1 Initialization

DR Net's GENNET system generation program creates a DRNET.CMD file that is integrated with your Concurrent CP/M CCPM.SYS file by the ADDNET utility. This results in a single CCPM.SYS file with DR Net installed. There is no need to keep the DRNET.CMD file on the boot disk to load DR Net.

DR Net initialization occurs when the Concurrent CP/M system is cold booted. As soon as this completes, the DR Net server function is available to all requesters. However, the user must attach the network with the NETON utility before the DR Net requester function is enabled. An "attach" creates the network environment, which provides among other things the resource map in the Requester Configuration Table, and allows requester processes to access the network.

The DR Net initialization routine called by Concurrent CP/M's coldstart procedure examines DR Net internal Parameter Table (see Section 2.3.1) and performs the following:

- The NIOS resident NET_INIT routine is called.

- A pool of Requester Control Block is created. The number of Requester Control Blocks in the pool is derived by adding 1 to the sum of server and requester processes specified in the Parameter Table.

- A watchdog process is created that maintains DR Net's server and requester time-out functions.

- An input message routing process is created for each input Line Driver Control Block.

- An output message routing process is created for each output Line Driver Control Block.

- Each input and output message routing process calls its NIOS resident LD_INIT routine to mobilize the physical network input or output port.

These processes remain active as long as the computer is running.

2.1.2 NDOS requester module

The NDOS, or Network Disk Operating System, is an operating system module loaded along with the other Concurrent CP/M modules at system coldstart. It performs DR Net's screening function, and provides the support for the DR Net system calls. Table 2-1 lists the DR Net functions. See the "DR Net Programmer's Guide" for the description of their use.

Table 2-1. DR Net system calls

Dec	Hex	Mnemonic	Description
64	40	N_LOGON	Log on a process to a server.
65	41	N_LOGOFF	Log off a process from a server.
66	42	?	
67	43	?	
68	44	N_STAT	Display the network status word.
69	45	N_RCT	Return the Requester Configuration Table, or set an entry in it.
70	46	N_ATTRIB	Set program compatibility attributes.
71	47	N_SCT	Return the Server Configuration Table.
72	48	N_ERRMODE	Set network error mode.
73	49	N_ATTACH	Attach process to the network.
74	4A	N_DETACH	Detach process from the current network environment.
75	4B	?	
76	4C	?	
77	4D	N_PARATAB	Return node's Parameter Table.

The DR Net functions, except N_ATTACH, are not available to a process, and the NDOS call screening is not implemented for a process until that process is attached to the network. The operator can attach the current process, all parent processes, and all child processes with the NETON utility. A process can attach itself or another process with the N_ATTACH function.

When a process attaches, DR Net allocates a Requester Control Block from the pool and a Network Data Area from the Static Buffer, and makes a copy of the Master Requester Configuration Table. (The Network Data Area (NDA) is a temporary construct, similar to the User Data Area, necessary to sustain a process's network activity. The NDA is maintained as long as the requester process is attached to the network.) This provides the attached process with its network environment and resource map.

After the network environment is in place, all disk drive, list device, and queue related system calls made by an attached process initiate the same sequence of tasks. The following six steps summarize the decisions made and actions taken in this sequence.

1. Concurrent CP/M's SUP (supervisor) module routes all of its disk drive, list device, and queue related calls to the NDOS. Otherwise, these calls are sent to the local module for implementation.

2. The NDOS references the calling process's Requester Configuration Table to determine if the disk drive, list device, or queue is local or mapped to a server. If the resource is local, the call is sent back to the SUP for routing to the appropriate local module.

3. If the local resource is mapped to a server and if that server is logged on, the NDOS builds a DR Net message in the Dynamic Buffer. To determine which output line is used to access the server specified, the NDOS calls the NIOS resident NET_OUT routine, and passes to it a pointer to the Requester Control Block.

4. NET_OUT fills in the Requester Control Block's LDCB # field with the appropriate output Line Driver Control Block number, and returns to the NDOS. The NDOS then signals the output message routing process associated with that Line Driver Control Block, and passes it to the same Requester Control Block pointer.

5. The NDOS waits for the output message routing process to return with success of failure. When the output message routing process returns success, the watchdog process's trasaction time-out is sent, and the NDOS waits for the input message routing process to signal that the response message has been received. One of two results can occur while the NDOS is waiting:

   - The input message routing process signals that the message has been received.

   - The watchdog process expires and signals that the transaction has not returned within the time limit.

   If the output message routing process returns failure (indicating that the original request never arrived) or the watchdog process expires before the response is received, the NDOS returns a network error message to the calling process.

6. When the NDOS is signalled by the input message routing process that the response message has been received, it does the following:

   - The message is read from the Dynamic Buffer and decoded.

   - The response is returned to the calling application according to Concurrent CP/M return conventions.

   - The response message space in the Dynamic Buffer is deallocated.

2.1.3 Output message routing process

There is one output message routing process for each output Line Driver Control Block in the NIOS. The NDOS and shadow processes use the output message routing process associated with a particular line driver number to send a message. Which line driver to use is returned by the NIOS-resident NET_OUT routine. When the output message routing process is not in use, it is suspended.

Whenver a shadow process or the NDOS signals the output message routing process, they provide a pointer to as specific Requester Control Block. The output message routing process transfers the values for the current message pointer and message buffer size from the Requester Control Block into the Line Driver Control Block. To send the message, the Line Driver Control Block's LD_DRVR function is called.

LD_DRVR returns with either a success or failure code. Success means that the message was sent to the server node with no errors. The output message routing process reportsthis result to the NDOS or shadow process, deallocates the message space in the Dynamic Buffer, and waits for another signal. Failure indicates that LD_DRVR could not send the message.

Before returning after a failure, the output message routing process calls the current Line Driver Control Block's NIOS resident LD_ERR error recovery routine. Only when this routine returns does the output message routing process deallocates the message space in the Dynamic Buffer, and return the network failure code.

2.1.4 Input message routing process

The input message routing process first allocates a message space in the Dynamic Buffer. The size of this space is determined by adding the Parameter Table values for the message header and message trailer to the largest current message size. The input message routing process then writes the message's buffer pointer and size value into its Line Driver Control Block, and calls the NIOS-resident LD_DRVR routine. It has no further responsibilities until this routine receives a message, and returns.

When LD_DRVR returns success, the input message routing process must decide to whom the message belongs. It scans the pool of Requester Control Blocks in search of one that matches the process in the message's source ID (SID) or destination ID (DID) field, depending on whether the incoming message is a request or a response. This search has one of the following four results:

- A match is found. The input transport process copies the message size value and buffer pointer to that Requester Control Block, writes the "good" code to its Receive Status field, and signals the NDOS or shadow process.

- No match is found but the message's function field contains a function 64, N_LOGON, and the password is correct. The input message routing process scans the pool of Requester Control Blocks in search of one that is not allocated. When a blank Requester Control Block is found, it is allocated and a shadow process is created.

- No match is found and the message is not an N_LOGON. This enables the input message routing process's error handling mechanism, which uses the standard message output protocol, to send an error message to the offending requester.

- There is no match, the message contains function 64, but a Requester Control Block is not available, or the password is wrong. Again, the input message routing process's error handling mechanism is enabled to send an error message to the requester.

LD_DRVR can also return a NIOS failure condition to the input message routing process. When this occurs, the input message routing process calls the current Line Driver Control Block's NIOS resident LD_ERR function. After this routine returns, the current buffer is deallocated, a new message space in the Dynamic

Buffer is allocated, and LD_DRVR is called. No message is sent to any waiting process. It is expected that the waiting process's time-out routine will return the failure code.

2.1.5 Shadow process

The input message routing process creates shadow processes as it receives N_LOGON messages with the proper password. Each requester process that sends an N_LOGON is granted a shadow process until the maximum number of shadow processes specified in the Parameter Table is reached. A shadow process is also not created should the N_LOGON message contain an erroneous password.

The shadow process remains assigned to a requester process until one of the following occurs:

- A log-off message is received from its requester process.

- A failure code is returned by the output message routing process when the shadow process attempts to send a message.

- A keep-alive (see following Note) message is not received within 45 seconds.

- The process does not open a file, access a list device, or call F_SFIRST or F_SNEXT within 15 seconds (see following Note).

Note: The last two items in this list are time-out functions monitored by the watchdog process. See the description of the watchdog process below for exaplanations of these two, server-only, timers.

After the shadow has been created, it immediately begins servicing requests. This sequence of events, which is the same for all types of response messages, consists of the following steps:

1. The shadow process makes whatever system calls are necessary to perform the function specified by the request.

2. Using its current message space in the Dynamic Buffer, the shadow builds the response message.

3. The shadow calls the NIOS resident NET_OUT function to get the Line Driver Control Block number for the destination requester.

4. The shadow signals the associated output message routing process to output the message, and waits for it to signal back with a success or failure code.

5. If the output message routing process returns success, the shadow waits for another message. If the output message routing process returns failure, the shadow terminates itself, and its Requester Control Block and Static Buffer space are available for allocation.

Under certain circumstances, shadow processes can wait indefinitely for a message to arrive from their requesters. This is predicated on the continuous receipt of the special keep-alive messages. However, there are circumstances in which a shadow process does terminate itself from lack of use. Bot the keep-alive and in-use functions are controlled by the watchdog process.

2.1.6 Watchdog process

In a requester node, the watchdog process decrements the value in the ticks-to-time-out field in each allocated Requester Control Block. The NDOS sets this field with the transaction time-out value when the output message routing process returns success. If the time expires before the response message has been received, the watchdog writes an error code in the Requester Control Block's send status field, and signals the NDOS. Note that the transaction time-out is a value entered at DR Net system generation time.

In a server node, the watchdog proces monitors two timer functions, the keep-alive and the in-use time-outs. These are both predefined values that cannot be changed.

Keep-alive time-out

Every 17 seconds in every requester node, the watchdog process sends a special keep-alive message to all servers that are currently logged on by any attached processes. In the servers, this message is interpreted to mean that the requester is still active on the network. Should a keep-alive message not be received in 45 seconds, the wathcdog assumes that the network link has failed, or the requester node has crashed.

The purpose of the keep-alive time-out is to release server resources that are allocated to requesters that, for any reason, are no longer active on the network. The implications of a keep-alive failure at that all open files belonging to the shadow process are closed, possibly preventing inclusion of the most recent updates, and all entries in the file lock list are purged. Note, too, that the requester process is not informed that it is no longer logged on.

In-use time-out

The watchdog process also forces each shadow process to determine if it is in use. Unless a shadow has received a message within the past 15 seconds, the watchdog signals the shadow to evaluate itself according to the following criterai:

- Is it servicing a CP/M Release 2.2 or CP/NOS requester?

- Does it have any open files?

- Does it have any list jobs active?

- Does it have any consoles in use?

- Was the last system call serviced a function 17, F_SFIRST, or 18, F_SNEXT?

If the answer to all these questions is NO, the shadow process terminates itself. Such a termination is transparent to its requester. No message is sent by the server to indicate that its shadow process has been terminated.

The purpose of the in-use time-out is to purge shadow processes that are only taking up space in the server. For example, a requester's TMP retains its shadow process after the user has invoked an application. In this case, the TMP's shadow process is taking up Static Buffer space and a Requester Control Block that could be put to better use by another process. Because of the in-use time-out, server congestion is minimized, and more requesters have more frequent access to the server.

The implications of the in-use time-out are that a requester process that is logged on to a server might not have a shadow process. When this occurs, DR Net automatically sends an N_LOGON message when the requester process attempts to access a resource mapped to that server. If there is no shadow process available, DR Net polls the server until one can be created.

2.2. CP/M-86 requester nodes

Note: The DR Net NDOS adds several Concurrent CP/M functions to nodes running CP/M-86. This allows applications designed to run under Concurrent CP/M to run under CP/M-86 based nodes when they are attached to the network. The additional system calls supported are described in the "DR Net Programmer's Guide".

The DR Net system module is not integrated with the CP/M-86 system file. Instead, the DRNET.CMD file created by the GENRQR system generation utility is recorded separately, and loaded with the NETLDR utility. Figure 2-3 illustrates the relationship between DR Net and its CP/M-86 host after NETLDR has been run.

```
             +--+-----------+
     DR Net  |  |    NIOS   |
            -+  +-----------+
     Module  |  |    NDOS   |<-----+
             +--+-----------+      |
                |           |      |
                |    TPA    |      |
                |           |      |
             +--+-----------+      |
             |  |    BIOS   |      |
             |  +-----------+      |
    CP/M-86 -+  |    BDOS   |<-X-X-+ (disabled)
             |  +-----------+      |
             | |    CCP    |      |
             +--+-----------+      |
                | Interrupt | 224--+
                |  Vectors  |
                +-----------+
                System Memory
```

Figure 2-3. CP/M-86 requester memory organization

2.2.1 Initialization

The NETLDR utility performs the following tasks:

- loads the file DRNET.CMD into the uppermost portion of the computer's memory

- removes the memory required by DR Net from the BIOS's Memory Region Table (MRT)

- saves the BDOS entry point at 0380h, and replaces it with a long pointer to the NDOS entry point

- transfers control to the DR Net initialization routine which calls the NIOS-resident LD_INIT and NET_INIT routines before returning

Conceptually, these changes insert the NDOS between an application program and CP/M-86. From this position, the NDOS screens all system calls to determine if a disk drive, list device, or queue that is mapped to a server is referenced.

2.2.2 CP/M-86 request routine

The operation of the CP/M-86 requester is analogous to that of the Concurrent CP/M requester, with the following broad exceptions:

- Because CP/M-86 is a single threaded system, the input and output message routing processes are simple routines incorporated into the NDOS.

- There is no watchdog process. If you want a time-out mechanism, it will have to be implemented in the NIOS.

- Because there is not an input message routing process to monitor the network input line driver, the NDOS calls the NIOS resident NET_IN function to find out where to expect the incoming response message.

The CP/M-86 NDOS operates in a similar fashion to the Concurrent CP/M NDOS, except that all (rather than just the disk drive, list device, and queue related) system calls are trapped. However, only the disk drive, list device, and queue related calls engage the NDOS's call screening process. As in a Concurrent CP/M requester, the NDOS checks the resource's entry in the

Requester Configuration Table, to determine if it is mapped to a server.

DR Net's network transaction routine in a CP/M-86 requester follows the same sequence of NIOS calls used by the Concurrent CP/M's NDOS, output message routing process, and input message routing process.

Table 2-2 lists the routines in the sequence they are called, and reviews their significance.

Table 2-2. NIOS call sequence in a CP/M-86 requester

Format: Function
 Significance

NET_OUT
Returns Line Driver Control Block for output line to server designated in message.

LD_DRVR
Outputs message.

LD_ERR
Recovers from network error. (This function is called only when LD_DRVR returns an error code.)

NET_IN
Returns Line Driver Control Block for input line of response message.

LD_DRVR
Inputs message.

LD_ERR
Recovers from network erro. (As above, this is called only when LD_DRVR returns an error code.)

When the NDOS receives the response message, it returns the information to the calling application according to CP/M-86 return conventions.

2.3 Internal data structures

DR Net has four internal data structures relevant to system implementation:

1) The Parameter Table contains two types of variables: node dependent and implementation dependent. The node dependent variables contain values that uniquely identify the node, and that have a bearing upon the node's network functional role. The implementation dependent variables contain values that evolve from the network interface, and are the same in each node.

2) The Requester Configuration Table is a node's resource map. The NDOS refers to this table to determine whether the disk drive, list device,

or queue referenced in the system call is remote or local.

3) The Requester Control Block contains network related data used by processes to conduct message between the DR Net processes and modules.

4) The Line Driver Control Block contains pointers to the line driver dependent, message send or receive, initialization, and error recovery routines, as well as message buffer pointers and status information. Each network node has at least two Line Driver Control Blocks: one to send a message, and one to receive a message.

The Parameter and requester Configuration Tables are created, and their values set by your responses to prompts displayed by the DR Net GENNET and GENRQR system generation utilities. These programs are described in Section 4.

Requester Control Blocks are created by DR Net during DR Net initialization. The values in each field are set as the block is allocated to a requester or shadow process. In requesters, this happens when a process attaches. In servers, this happens when an N_LOGON message is received. The Requester Control Block remains allocated until the process detaches in a requester, when a requester logs off in a server, or when the network connection fails.

The Line Driver Control Block is a data structure resident to the NIOS. Most fields are initialized in the NIOS. However, two fields are used for the temporary storage of data pertinent to the current transaction.

2.3.1 Parameter table

Each network node has its own Parameter Table. All values are set during DR Net system generation with the GENNET and GENRQR utility. Where these fields contain a number, the value is always expressed in hexadecimal.

Note: Any and all Parameter Table values can be set from the NIOS-resident NET_INIT routine. When this routine is implemented, the corresponding entries made with GENNET or GENRQR are overwritten.

Figure 2-4 illustrates the organization of the Parameter Table. Table 2-3 contains the field descriptions.

Numbers indicate hex offset from first byte in table.

```
   00      01         02         03    04      05
+------+----------+-------------+---------+--------+--------+
| Node | # Shadow | # Requester | # LDCBs | Maximum Message |
|  ID  | Processes|  Processes  |         |      Size       |
+------+----------+-------------+---------+--------+--------+
 06      07     08      09     0A      0B      0C
+--------+-------+--------+-------+-------+--------+-------+
| Message Buffer | Message Buffer | First | Dynamic Buffer |
|  Header Size   | Trailer Size   | Flag  |      Size      |
+--------+-------+--------+-------+-------+--------+-------+
 0D 0E           0F 10        11 12 13 14 15 16 17 18
```

```
+--------+---------+------+-------+----------------------+
| Ticks to Timeout |   Reserved   |    Password Field    |
|   (in seconds)   |              |                      |
+--------+---------+------+-------+----------------------+
    19       1A    1B 1C 1D 1E
+---------+--------+-----------+
| Default | # RCTs | Reserved  |
|  Server |        |           |
+---------+--------+-----------+
```

Figure 2-4. Parameter Table

Table 2-3. Parameter table field descriptions

Format: Field name
Description

Node ID
The network ID number that uniquely identifies the node. Any value from 00h through 0FEh is valid.

# Shadow Processes
The total number of simultaneous, shadow processes that are supported on this server node.

# Requester Processes
The total number of simultaneous, attached requester processes that can be supported on this requester node.

# LDCBs
The total number of input and output Line Driver Control Blocks in the NIOS.

Maximum Message Size
The maximum length in bytes of a DR Net logical message. (See Appendix A for a description of the DR Net logical message size requirements.)

Message Header Size
The number of bytes in the optional message header. Each DR Net logical message is offset from the message buffer pointer in the Line Driver Control Block by the value of this field. This allows you to insert information necessary to sending the message from node to node.

Message Trailer Size
The number of un-initialized bytes in the optional message trailer. This area is reserved by the NDOS and shadow processes in the Dynamic Buffer at the end of each DR Net message.

First Flag
The first system flag allocated for DR Net's use. This is not the only flag required by DR Net. (Every Requester Control Block is allocated a system flag.) All flags used by DR Net must be initialized under Concurrent CP/M, and reserved for DR Net's exclusive use.

Dynamic Buffer Size
The total length of the DR Net buffer reserved for the temporary storage of DR Net messages as they are input from, and output to, the network.

Transaction Time-out
The number of seconds allowed for a network transaction to complete before the requester aborts the call. See Section 2.1.6, "Watchdog process", for a description of the use of this parameter.

Password
In requester nodes, this is the default password used by the LOGON.CMD utility when the operator does not enter a password in the command line. If the operator enters a password, the contents of this field are ignored. In a server node, this is the password that must be matched before a requester can log on.

Default Server
The hexadecimal server node ID number referenced when no node is specified in the LOGON, LOGOFF, or NET command lines. DR Net utilities also assume that any name service file exists on the default server. For operator convenience, this should usually be the requester node's principal server.

# RCTs
The maximum number of Requester Configuration Tables that can exist at any one time in a requester. This field, therefore, specifies the number of distinct network mappings that a given node can support.

2.3.2 The Requester Configuration Table

Each requester node has a master Requester Configuration Table that defines its default map of disk drives, list devices, and queues. The master table is created by the GENNET or GENRQR program, and is immutable.

Note: Any and all Master requester Configuration Table values can be set from the NIOS-resident NET_INIT routine. When this routine is implented, the corresponding entries made with GENNET or GENRQR are overwritten.

Each requester process attached to the network uses a direct copy, or a derivative copy, of the Master requester Configuration Table as its resource map. A process acquires a direct copy when it attaches to the network and neither the process nor its parent are already attached. This single copy is then shared between that process and all child process it creates after the attach. Note that all child processes of an attached process are automatically attached to the network.

Derivative copies of the Requester Configuration Table are made when an attached process attaches again. When this is done, DR Net makes a copy of the current Requester Configuration Table for exclusive use by the attaching process and all of the child processes it creates. More copies of the Requester Configuration Table are made, until the maximum specified in the Parameter Table is reached.

In most cases, a single Requester Configuration Table for an entire process family is sufficient. However, it is possible to create multiple, and possibly conflicting, environments using the N_ATTACH function. For a description of how a process's network environment is affected by multiple N_ATTACH calls, see Section 1.3.3, "Network environments and process families", in the "DR Net Programmer's Guide".

The user makes changes to individual map assignments with the NET and LOCAL utilities. In addition, the N_RCT system call can be used to make changes from an application. All changes made affect only the process's copy of the Table, and never result in changes to the Master Requester Configuration Table.

Figure 2-5 illustrates the format of the Requester Configuration Table. Field descriptions follow the illustration.

```
     (Numbers indicate hexadecimal offset from first byte in table.)

                         Byte 1    Byte 2
                         ------    ------   Typical of all
       00  01       Bits: 7 6-4 3-0           disk and list
      +---+---+            +-+---+---+---------+ device entries:
      |   |   | Reserved for  |a| * | b |Server ID| a = network bit
      +---+---+ system use.   | | | |   | number  | b = remote device
                           +-+-|-+---+---------+     number
                            \  +--> Reserved  /
                             \              /
Disk Drive Map:               \            /
                               +          +
       02  03  04  05  06  07  08  09| 0A  0B| 0C  0D  0E  0F  10  11
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      | 00(A) | 01(B) | 02(C) | 03(D) | 04(E) | 05(F) | 06(G) | 07(H) |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

       12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F  20  21
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      | 08(I) | 09(J) | 0A(K) | 0B(L) | 0C(M) | 0D(N) | 0E(O) | 0F(P) |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

       22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F  30  31
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      |                            Reserved                           |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

       32  33  34  35  36  37  38  39  3A  3B  3C  3D  3E  3F  40  41
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      |                            Reserved                           |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

List Device Map:

       42  43  44  45  46  47  48  49  4A  4B  4C  4D  4E  4F  50  51
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      |  00   |  01   |  02   |  03   |  04   |  05   |  06   |  07   |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

       52  53  54  55  56  57  58  59  5A  5B  5C  5D  5E  5F  60  61
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      |  08   |  09   |  0A   |  0B   |  0C   |  0D   |  OE   |  0F   |
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

Queue Map: (First 16 separate entries)

       62  63  64  65  66  67  68  69  6A  6B  6C  6D  6E  6F  70  71
      +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
      | * | Local Queue Name              | Remote Queue Name         :
      +-|-+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
       +--> Type Flag

       72  73  74  75
      +---+---+---+---+
      :   |   | Remote|
      : * | * | Queue |
      : | | | |   ID  |
      +-|-+-|-+---+---+
       |   +--> Server ID
       +------> (Last character of Remote Queue Name.)

      Figure 2-5. Requester Configuration Table
```

Disk Drive and List Device Maps

Table bytes 02h through 21h contain the mapping for local drives 00 (A) through 0F (P), respectively. Similarly, bytes 42h through 61h contain the mapping for list devices 0 through 15. Each map entry consists of two bytes. The first byte contains the network bit and remote device number nibble, as shown in Figure 2-5, to indicate the following:

- The network bit value indicates whether system calls should access the local device or the remote device, as follows

```
          bit 7 = 0: send calls to local device
          bit 7 = 1: send calls to remote device
```

- The remote device number nibble contains the hexadecimal number of the replacement device on the server.

The second byte of each field contains the hexadecimal ID number of the destination server node.

Queue Map

Table bytes 62h through 1A1h contain the mapping for as many as 16 local queues. The fields are used as follows:

- The Type Flag byte replaces the network bit, to indicate whether this entry contains a mapping (non-zero) or is unused (0).

- The Local Queue Name contains the 8 character ASCII name of the local queue to be mapped.

- The Remote Queue Name contains the 8 character ASCII name of the server-based, replacement queue.

- Server ID contains the hex ID number of the server node with the mapped queue.

- Remote Queue ID contains the short pointer to the Queue Descriptor from the server's System Data Page.

2.3.3 The Requester Control Block

A pool of Requester Control Blocks is created during DR Net initialization in each network server and requester node. DR Net determines the number of blocks to be created by adding the values for "# Shadow Processes" and "# Requester Porcess" in the Parameter Table. Unlike Requester Configuration Tables, Requester Control Blocks are never shared.

In requester nodes, a Requester Control Block is allocated from the pool when a process attaches to the network. It remains assigned until the process detaches from the network.

In server nodes, a Requester Control Block is allocated from the pool to a shadow process when a log on message is received. It remains allocated until a log off message is received, or until an exceptional event occurs. The exceptional events that cause a shadow process to terminate are described in Section 2.1.5, "The shadow process".

Figure 2-6 illustrates the format of the Requester Control Block. Table 2-4 contains the field descriptions.

```
     Numbers indicate hex offset from first byte in block.

       00  01  02  03  04  05  06  07
     +---+---+---+---+---+---+---+---+
     |   Reserved    | *   * | * | * |
     +---+---+---+---+-|-+-|-+-|-+-|-+
                       |   |   |   |
                       |   |   |   +--> Send Status
                       |   |   +------> Receive Status
                       |   |     Requester ID:
                       |   +----------> Network Process ID
                       +--------------> Node ID Number

       08  09  0A  0B  0C  0D  0E  0F  10  11
     +---+---+---+---+---+---+---+---+---+---+
     | * | * |Number of Ticks|Long Pointer to|
     | | | | | until Time-out| Message Buffer|
     +-|-+-|-+---+---+---+---+---+---+---+---+
       |   |
       |   +--> Flag
       +------> Type

       12  13  14
     +---+---+---+
     |   *   | * |
     +---+---+-|-+
         |     |
         |     +--> LDCB #
         +--------> Current Message Buffer Size
```

Figure 2-6. Requester Control Block

Table 2-4. Requester Control Block field descriptions

Format: Field
Description

Requester ID
This two-byte field contains the hexadecimal requester node ID number and a network process ID number. The process number is arbitrarily assigned by DR Net in the requester node when the process attaches the network. This value is created only for Concurrent CP/M requesters. The network process ID number for CP/M-86 based requesters is always zero.

Receive Status
This byte indicates the current use status of the RCB, or the failure or success of the last attempt to receive a message. This field can have the following hexadecimal values:

0	Empty, this RCB is not allocated.
1	Receiving, the LD_DRVR function has been called, but has not returned success or failure yet.
2	?
3	Receive good, the LD_DRVR function has completed, and a valid message is in the message buffer.
4	?
5	Receive bad, the LD_DRVR function has completed, but the message buffer contents are not valid.
6	?
7	The watchdog process timed out.
8	A valid message is in the message buffer, but it is not a standard DR Net logical message.
9	A keep-alive time-out occurred.

Send Status
This byte indicates the current use status of the RCB, or the failure or success of the last attempt to send a message. This field can have the following values:

2 Sending, the LD_DRVR function has been called and has not returned success or failure yet.
4 Send good, the LD_DRVR function has completed, and an acknowledge has been received from the destination.
6 Send bad, the LD_DRVR function has completed, but there is no assurance that it was received properly.
8 The send routine was called to send a non-standard message.

Type
This field indicates whether the Requester Control Block (RCB) is bound to a requester or a shadow process. The values foind in this field have the following definitions:

0 RCB is not bound to any process.
1 RCB is bound to a shadow process.
2 ?
3 RCB is bound to a requester process.
4 Rerserved for system use.
8 RCB will be bound to a shadow process, but the shadow is being created.

Flag
The system flag number assigned to this RCB.

Number of Ticks until Time-out
In Requester Control Blocks bound to requester processes, this field contains the transaction time-out's current value. The value is reset when the output message routing process returns success. In Requester Control Blocks bound to shadow processes, this field contains the current value of the kee-alive time-out.

Long Pointer to Message Buffer
The current location of the DR Net logical message in the Dynamic Buffer. The value is copied from the Line Driver Control Block (LDCB) by the input message routing process when a message is received, and copied to the LDCB by the output message routing process when a message is sent. Unlike the message pointer in the Line Driver Control Block, this pointer points directly to the DR Net logical message. There is no intervening message header offset.

Current Message Buffer Size
This parameter represents the sum of the current message length, which is not necessarily the maximum message length, the message header, and the message trailer.

LDCB #
This byte contains the number of the Line Driver Control Block (LDCB) through which the last message was received or sent. The NIOS NET_OUT and NET_IN routines are responsible for setting this field.

2.3.4 The Line Driver Control Block

The NIOS contains a Line Driver Control Block (LDCB) for each network input
and output line. All fields, except the buffer size and the long pointer to
the message buffer, are set in the NIOS. These fields are transient. They are
set by the input and output message routing processes in Concurrent CP/M based
nodes, or by the NDOS in CP/M-86 based nodes, according to current values.

Figure 2-7 illsutrates the format of the Line Driver Control Block. Table 2-5
lists the field descriptions.

```
     Number indicates hex offset from the first byte in table.

       00  01  02  03  04  05  06  07  08
     +---+---+---+---+---+---+---+---+---+
     | * | * | * |   *   |       *       |
     +-|-+-|-+-|-+---|---+---+---|---+---+
       |   |   |     |           |
       |   |   |     |           +--> Long Pointer to the Message Buffer
       |   |   |     +--------------> Current Message Buffer Size
       |   |   +--------------------> Type
       |   +------------------------> Reserved
       +----------------------------> LDCB Number

       09  0A  0B  0C  0D  0E  0F  10
     +---+---+---+---+---+---+---+---+
     |       *       |       *       |
     +---+---|---+---+---+---|---+---+
             |               |
             |               +--> Long Pointer to the
             |                    Send or Receive Routine
             +------------------> Long Pointer to the
                                  Driver Initialization Routine

       11  12  13  14
     +---+---+---+---+
     |       *       |
     +---+---|---+---+
             |
             +--> Long Pointer to the
                  Error Recovery Routine
```

     Figure 2-7. Line Driver Control Block

Table 2-5. Line Driver Control Block field descriptions

Format: Field
     Description

LDCB Number
A unique, hexadecimal ID number of the Line Driver Control Block in the NIOS.
The first Line Driver Control Block must be numbered 0, and multiple Line
Driver Control Blocks must be numbered sequentially. This is the number
written in the Requester Control Block by your NIOS NET_IN and NET_OUT
routines.

Type
This byte has one of the following two hexadecimal values:

00	for an input Line Driver Control Block
01	for an output Line Driver Control Block

Current Message Buffer Size
This two-byte, hexadecimal value defines the size of the message buffer that has been passed to the driver for sending or receiving a message. This size is always equal to the current largest logical message plus extra space for the header and trailer. (The header and trailer values are taken from the Parameter Table.) In all cases, the message sent or received can be less than or equal to the size specified in this field. However, a message can never be greater in length.

Long Pointer to Message Buffer
This field contains the offset and segment of the message buffer in which a message to be sent currently resides, or to which a message that has been received is to be transferred. The DR Net logical message is offset from the pointer by the length of the header specified in the Parameter Table.

Long Pointer to the Driver Initialization Routine
This double word field contains the offset and segment of the line driver's LD_INIT initialization routine.

Long Pointer to the Send or Receive Routine
This double word field contains the offset and segment of the line driver's LD_DRVR driver routine. Depending upon the LDCB type, this is the routine that either sends or receives a DR Net message.

Long Pointer to the Error Recovery Routine
This double word field contains the offset and segment of the line driver's LD_ERR error recovery routine.

EOF

DNSG3.WS4   (= "DR Net System Guide", section 3)

DR Net
System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Section 3: The NIOS

The Network I/O System (NIOS) is the implementation dependent interface between the network hardware and the DR Net proprietary module. The interface consists of a set of functions for which you develop the supporting routines. DR Net calls these functions in one of two ways. A jump table at the beginning of the NIOS provides pointers to NIOS's global functions, and the Line Driver Control Blocks provide pointers to the line driver dependent functions.

The extent to which you support these functions depends upon the features you want to implement and your network hardware. For example, a no-frills Concurrent CP/M requester/server only requires supporting routines for the NET_OUT and the two LD_DRVR functions. In addition, an LD_DRVR routine for an intelligent network controller requires much less support than would a network controller that relied on program control for its lowest level protocols.

The NIOS description in this section is divided into three topics.

- Section 3.1, "The NIOS structure", describes the NIOS organization.

- Section 3.2, "The NIOS global functions", describes the five functions accessed from the NIOS's jump table.

- Section 3.3, "The NIOS Line Driver functions", describes the three line driver dependent functions assessed through the pointers in the Line Driver Control Block.

3.1 The NIOS structure

The DR Net GENNET and GENRQR system generation utilities expect a NIOS.CMD module that adheres to a specific format. There are three rules governing the format of the NIOS.

1) The NIOS must be contained in two CMD groups: a data segment and a code segment. Neither of these can exceed 65,536 bytes of memory.

2) The NIOS data segment must be ORGed at 0100h, and have a specific structure.

3) Any values in the CMD file header record for group descriptors A-BASE, G-MIN, and G-MAX fields are ignored by the DR Net system generation program.

3.1.1 NIOS data segment

The NIOS data segment must be organized as shown in Table 3-1.

Table 3-1. NIOS data segment structure

Address	Definition
0000-00FF	Unused Base Page (The base page can be used for local NIOS data, if so desired.)
0100-0103	Long Pointer to NET_OUT Routine
0104-0107	Long Pointer to NET_IN Routine
0108-010B	Reserved
010C-010F	Long Pointer to NET_STATUS Routine
0110-0113	Long Pointer to NET_WBOOT Routine
0114-0117	Long Pointer to NET_INIT Routine
0118-011F	Reserved
0120-0133	Line Driver Control Block Number 00
0134-0147	Line Driver Control Block Number 01
0148-xxxx	Additional Line Driver Control Blocks
xxxx+	Any Additional NIOS Data

The jump table at the beginning of the NIOS provides access to the five global NET routines. The three line driver dependent routines are accessed through the Line Driver Control Blocks. Each of these contains three long pointers: one to the network port's input or output driver, another to its initialization routine, and the last to an error recovery routine.

The Line Driver Control Blocks have a specific format, and some information must be coded into them by the NIOS writer. The fields that must be defined follow:

- Line Driver Control Block number
- line driver type (input or output)
- LD_INIT initialization routine pointer
- LD_DRVR message send or receive routine pointer
- LD_ERR error recovery routine pointer

The NIOS is not limited to a specific number of Line Driver Control Blocks. In addition, you can have different numbers of input and output Line Driver Control Blocks, and more than one Line Driver Control Block for a single input or output port. The only rules regarding the number and numbering of Line Driver Control Block are as follows:

- The Line Driver Control Block numbers must be numbered sequentially from 00h.

- The number of Line Driver Control Blocks in the NIOS must equal the number of Line Driver Control Blocks specified during GENNET and GENRQR.

3.1.2 NIOS code segment

The five global routines and the three driver dependent routines provide a framework for building the NIOS's code segment. Unlike the data segment, there are no format restrictions on the code segment.

The functions found in the NIOS code segment have some general restrictions. First, some functions must be reentrant. That is, different processes must be able to call these functions simultaneously. When a function must be reentrant, it is noted in the descriptions that follow.

Second, NIOS functions must never call N_ATTACH, even to recover Parameter, or Requester Configuration, Table values. If you foresee that these values might be necessary, we recommend making copies of these tables as described in the NET_INIT function.

Third, some NIOS functions must not make BDOS calls, or make any call that can result in another network transaction. It is noted in the descriptions that follow which calls must observe this rule.

Finally, any function can produce a load-time fixup. For example, an assembly language instruction SEG pseudo-operation is permissible. The DR Net system generation utilities manage these fixups properly.

3.2 NIOS global NET functions

There are five global functions. Table 3-2 lists the functions by name, and summarizes their purposes. Table 3-3 lists the input parameters and the returned values expected by the calling process.

Table 3-2. NIOS global functions

Format: Function
Purpose

Function	Purpose
NET_OUT	Return the output Line Driver Control Block number for the current destination.
NET_IN	Return the input Line Driver Control Block number on which the response message from the current destination can be expected (used in CP/M-86 only).
NET_STATUS	Return an implementation-defined status byte.
NET_WBOOT	

Perform a non-essential, implementation-defined, warm boot routine after a function 0 or 143 is executed.

NET_INIT
Perform an optional initialization routine as a subroutine of DR Net initialization.

Table 3-3. Global function input parameters and returned values

Function	Input parameter	Returned values

NET_OUT	DX = RCB* offset
	DS = RCB segment
		Line driver number
		field in RCB updated.

NET_IN	DX = RCB offset
	DS = RCB segment
		Line driver number
		field in RCB updated.

NET_STATUS	DX = RCB offset
	DS = RCB segment
		Al = Network Status Byte

NET_WBOOT	None	None

NET_INIT	DX = HCB* offset
	DS = HCB segment
		None

* = "RCB" indicates the "Requester Control Block", "HCB" indicates the "Header Control Block".

None of these routines is expected to return values in any registers, except NET_STATUS which returns the Network Status Byte in AL. However, NET_IN and NET_OUT are expected to write a Line Driver Control Block number into the Requester Control Block before returning. All global NET functions should return to the caller by executing a RETF (return far) instruction. No registers, except for SS and SP, need be preserved.

NET_OUT Network Output port

Return the output Line Driver Control Block (LDCB) number for the current destination.

Entry Parameters:
    Register DX: Requester Control Block -- Offset
            DS: Requester Control Block -- Segment

Returned Values: None

    Result: "LDCB #" field in Requester Control Block updated

The NDOS and shadow processes call NET_OUT to determine which Line Driver Control Block should be used to send a message to a particular destination. NET_OUT is provided a long pointer to the Requester Control Block in register pair DX and DS, and must update the Requester Control Block's "LDCB #" field. Figure 3-1 illustrates the "LDCB #" field and other significant fields in the Requester Control Block.

Note: Because it can be called by several processes simultaneously, NET_OUT must be reentrant. It must also make no system calls that could cause the NDOS to send a message, because the NDOS handles recursion only if no network transaction takes place.

```
     Numbers indicate hex offset from first byte in block.

       00  01  02  03  04  05  06  07
     +---+---+---+---+---+---+---+---+
     |   Reserved    | *   * |   *   |
     +---+---+---+---+-|-+-|-+---|---+
                       |   |     |
                       |   |     +----> System use only
                       |   |      Requester ID:
                       |   +----------> Network Process ID
                       +--------------> Node ID Number

       08  09  0A  0B  0C  0D  0E  0F  10  11
     +---+---+---+---+---+---+---+---+---+---+
     | * |  System use only  |Long Pointer to|
     | | |                   | Message Buffer|
     +-|-+---+---+---+---+---+---+---+---+---+
       |
       +------> Type

       12  13  14
     +---+---+---+
     |   *   | * |
     +---+---+-|-+
         |     |
         |     +--> LDCB #
         +--------> Current Message Buffer Size
```

Figure 3-1. Significant Requester Control Block fields

In a simple NIOS where there is only a single output line driver, NET_OUT need set only the number of the output Line Driver Control Block number at offset 14h from the Requester Control Block pointer. In systems with more than two output lines, identifying which Line Driver Control Block is appropriate for the current destination is more complicated. The Requester Control Block (RCB) provides several values that can help your routine locate the right LDCB.

- When the calling process is a shadow process, the current "LDCB #" field in the Requester Control Block contains the number of the line driver from which the request was read. (Recall that the NIOS can detect the calling process from the "Type" field, offset 08h, in the Requester Control Block. A 01h value indicates that the RCB is bound to a shadow process. A 03h value indicates a requester process owns the RCB.) In most cases, there is always a specific correlation between a given input line driver and the output line driver to be used to send the response. However, when the Requester Control Block belongs to a requester process, the "LDCB #" value is undefined.

- When the calling process is a shadow process, offset 04h from the Requester Control Block pointer always contains the destination node ID number. (However, when the RCB is bound to a requester process, this value is the current node number, and has little use.)

- Requester Control Block bytes 0Eh through 11h contain a long pointer to the format field of the message to be sent.

The long message pointer gives NET_OUT direct access to the DR Net logical message. Most important to NET_OUT's purpose is node ID in the message's Destination ID (DID) field. In all cases, the destination node ID is a one-byte value offset 1 byte from the start of the message.

Note: Unlike the LD functions, the message pointer in the Requester Control Block points directly to the DR Net logical message's format byte. Figure 2-2 illustrates the contents of the logical message. A more extensive description is provided in Appendix A.

NET_IN Network Input port

Returns the input Line Driver Control Block (LDCB) number for an expected response message.

```
Entry Parameters:
    Register DX: Requester Control Block -- Offset
             DS: Requester Control Block -- Segment

 Returned Values: None

        Result: "LDCB #" field in Requester Control Block updated
```

This function is called only by the NDOS of a CP/M-86 requester. It is called to return the Line Driver Control Block number of an incoming response message. Because of the host operating system, this functions need not be reentrant. However, it must not make any CP/M-86 system calls.

NET_IN is very similar to NET_OUT. Before the NDOS calls NET_IN, it places the long pointer to the Requester Control Block in register pair DS and DX. All of the Requester Control Block and DR Net message fields described in NET_OUT are available to NET_IN to help identify the appropriate Line Driver Control Block number. However, because the message has not been received, the node ID field in the message buffer is undefined. Note that the "LDCB #" value in the Requester Control Block when NET_IN is called contains the number of the Line Driver Control Block used to output the request message.

NET_STATUS

Return an implementation-defined network status byte.

Entry Parameters:
    Register DX: Requester Control Block -- Offset
             DS: Requester Control Block -- Segment

 Returned Values:
             AL: Network Status Byte

DR Net calls NET_STATUS after an application process has called function 68, N_STAT. The NDOS provides NET_STATUS with a long pointer to the Requester Control Block in register pair DS and DX. DR Net has no requirements for the value returned in register AL. This function must be reentrant in all Concurrent CP/M based nodes, and must make no system calls that result in the NIOS initiating a network transaction.

Should you implement this function, note that DR Net sets several bits before returning the information to the application. This is done after NET_STATUS has returned. The bits initialized by DR Net are shown in Figure 3-2.

```
         7   6   5   4   3   2   1   0
       +---+---+---+---+---+---+---+---+
       | * | * | * | * | * | * | * | * |
       +-|-+-|-+-|-+-|-+-|-+-|-+-|-+-|-+
         |   |   |   |   |   |   |   |
         |   |   |   |   |   |   |   +--> Free
         |   |   |   |   |   |   +------> Free
         |   |   |   |   |   +----------> Reserved (*)
         |   |   |   |   +--------------> Free
         |   |   |   +------------------> Logged On (*)
         |   |   +----------------------> Free
         |   +--------------------------> Free
         +------------------------------> Reserved (*)
```

   (*) Regardless of the value returned by NET_STATUS, these bits are set to
       the appropriate value by DR Net before returning to the calling
       process.

       Figure 3-2. Network Status Byte

When bit 4 in the Network Status Byte is set, the calling process is logged on to at least one server. Bits 7 and 2 are used, but are reserved for system use.

NET_WBOOT Network Warm Boot

Non-essential warm-boot routine called by NDOS after P_TERM and P_TERMCPM.

Entry Parameters: None

 Returned Values: None

NET_WBOOT is called from the NDOS whenever an application calls system function 0 (P_TERM) or 143 (P_TERMCPM). It is provided as a convenience if you want to implement some network operation that should take place at the end of a program. For example, a NET_WBOOT routine could be written that sends a series of special messages to check for electronic mail. This function should be reentrant, and should make no system calls that would result in a network transaction.

Note: DR Net passes no parameters to NET_WBOOT. In addition, DR Net is not equipped to interpret a return value.

NET_INIT Network Initialization

Non-esential DR Net initialization routine.

```
Entry Parameters:
    Register DX: Header Control Block -- Offset
             DS: Header Control Block -- Segment

 Returned Values: None
```

NET_INIT is called from the DR Net initialization procedure when Concurrent CP/M is cold booted, or when the CP/M-86 NETLDR utility is run. This routine is provided a long pointer in register pair DS and DX to the DR Net Header Control Block. Like NET_WBOOT, NET_INIT is provided as a convenience. It has no DR Net or operating system related duties to perform. In addition, DR Net requires no return value from NET_INIT. This function need not be reentrant, and can use all of the system calls.

The Header Control Block is a series of long pointers. Table 3-4 lists the DR Net routines and data structures available from these pointers. The lefthand column in the table indicates the offset of each pointer from the pointer passed by DR Net.

Table 3-4. The DR Net Header Control Block

Pointer Offset	DR Net Routine or Data Structure
00 - 03	DR Net Initialization routine pointer
04 - 07	DR Net Entry routine pointer
08 - 0B	DR Net Data segment pointer
0C - 0F	NIOS Data segment pointer
10 - 13	DR Net Code segment pointer
14 - 17	NIOS Code segment pointer
18 - 1B	Parameter Table pointer
1C - 1F	Master Requester Configuration Table pointer
20 - 23	Static Buffer pointer
24 - 27	Dynamic Buffer pointer

Notice that this table provides access to the Parameter and Master Requester Configuration Tables. Consequently, NET_INIT can insert values in these tables when DR Net is loaded. All values set in this manner supercede the values written by the GENNET or GENRQR DR Net system generation programs.

Another use of NET_INIT is to make copies of the Master Requester Configuration and/or Parameter Tables. Recall that no NIOS function is allowed to call the N_ATTACH function. Consequently, is any NIOS function must reference the original network environment, the NET_INIT routine is the place to preserve the original contents of these tables in a permanent reference copy.

3.3 NIOS driver dependent LD functions

The following three functions differ from the NIOS global functions, because they are line driver dependent. The routines are accessed by DR Net from long pointers in the Line Driver Control Block. Table 3-5 summarizes the input and return values of each function. For the description of the Line Driver Control Block, see Section 2.3.4, "Internal data structures".

Table 3-5. Line Driver dependent function summary

Format: Function
    Description
    Input values
    Returned values

LD_INIT
Network driver initialization
DX = LDCB offset, DS = LDCB segment
AX = 0000h if success, 0FFFFh if failure

LD_DRVR
Input or output line driver
DX = LDCB offset, DS = LDCB segment
AX = 0000h if success, 0FFFFh if failure

LD_ERR
Error recovery
DX = LDCB offset, DS = LDCB segment
None

DR Net does not require a specific interface between the Line Driver Control Blocks and network I/O ports. Figure 3-3 illustrates three NIOS architectures that make use of different combinations of Line Driver Control Blocks (LDCB), input and output LD_DRVR routines, and network controllers. Each sample is described following the figure.

    1) A simple NIOS configuration

```
    +--------+    /--------\
    |  Input |-->|  Input  |--+
    |  LDCB  |   | LD_DRVR |  |   +--------------------+
    +--------+    \--------/  +-->| Network Controller |
    +--------+    /--------\  |   +-------------------+
    | Output |-->| Output  |--+
    |  LDCB  |   | LD_DRVR |
    +--------+    \--------/
```

2) A NIOS with reentrant routines

```
+--------+
|  Input |----+
|  LDCB  |    |
+--------+    |
+--------+    |    /--------\         +------------+
| Output |--+ +-->|  Input  |--+------>|  Network   |
|  LDCB  |  | |   | LD_DRVR |  |   +-->| Controller |
+--------+  | |    \--------/  +-+ |   +------------+
+--------+  | |                | |
|  Input |--+-+                +-+-+
|  LDCB  |  |     /--------\  | |     +------------+
+--------+  +---->| Output  |  | +---->|  Network   |
+--------+  |     | LD_DRVR |--+------>| Controller |
| Output |--+      \--------/         +------------+
|  LDCB  |
+--------+
```

3) A NIOS with multiple LDCBs driving a single controller

```
+--------+
|  Input |----+
|  LDCB  |    |
+--------+    |
+--------+    |    /--------\
| Output |--+ +-->|  Input  |--+
|  LDCB  |  | |   | LD_DRVR |  |
+--------+  | |    \--------/  |   +--------------------+
+--------+  | |                +-->| Network Controller |
|  Input |--+-+                |   +--------------------+
|  LDCB  |  |     /---------\  |
+--------+  +---->| Output  |--+
+--------+  |     | LD_DRVR |
| Output |--+      \--------/
|  LDCB  |
+--------+
```

Figure 3-3. Sample NIOS architectures

1. This simple NIOS architecture uses a single input and a single output LD_DRVR routine to service a single network controller.

2. This NIOS architecture uses multiple network controllers, each of which can handle message input from, and output to, the network. Notice in this model that all controllers are serviced by the same, reentrant LD_DRVR input and output routines.

3. In this model, multiple input and output Line Driver Control Blocks reference the same input and output LD_DRVR routines to service a single network controller. Certain efficiencies may be realized when multiple LDCBs are installed to access a single network controller, such as better message throughput, or better error handling characteristics.

In general, line driver routines need not be reentrant, although some of the architectures described above make reentrancy desirable. Line drivers must never call the N_ATTACH function. However, they can make use of any other system calls, because the input and output message routing processes are not logically attached to the network.

All line driver functions should return to their calling processes by executing a RETF (return far) instruction. No registers need to be preserved, except for the return code in AX, and SS and SP.

LD_INIT Line Driver Initialization

Initialize an input or output line driver.

```
Entry Parameters:
    Register DX: Line Driver Control Block -- Offset
             DS: Line Driver Control Block -- Segment

 Returned Values:
    Register AX:  0000h if success
                  0FFFFh if failure
```

Each line driver's LD_INIT function is called by the input or output message routing process created to manage the driver. (Recall that a separate routing process is created for every LDCB in the NIOS.) In Concurrent CP/M based nodes, this occurs when the computer is cold booted. In CP/M-86 nodes, this occurs when NETLDR is invoked. In both cases, this is the only time LD_INIT is called.

DR Net expects LD_INIT to return a 0000h if initialization of the line driver is successful, or 0FFFFh if initialization fails. The message routing process associated with the line driver terminates if failure is returned, and is not recreated. If a line driver requires no initialization, the LD_INIT routine should return with the success code in AX. For example, a single initialization routine is sufficient to set and test many network controllers. In this case, only one Line Driver Control Block's LD_INIT function needs to be implemented.

Although line driver initialization is called only once by DR Net, this does not preclude its use by other routines. LD_INIT can be called, for example, from LD_ERR to reinitialize the hardware after an LD_DRVR failure.

LD_DRVR Send or receive driver

Send or receive a message over the network.

```
Entry Parameters:
    Register DX: Line Driver Control Block -- Offset
           DS: Line Driver Control Block -- Segment

 Returned Values:
    Register AX:  0000h if success
              0FFFFh if failure
```

DR Net calls the LD_DRVR function to send or receive a message. The action depends upon whether the LDCB called is dedicated to message input or message output. For both message input and output, DR Net provides a long pointer to the Line Driver Control Block which, in turn, contains a long pointer to the message buffer. For input line drivers, this pointer marks the destination address of your LD_DRVR routine. For output line drivers, this pointer contains the message location. In return, the LD_DRVR must write 0000h to the AX register to indicate that the message has been successfully sent or received, or 0FFFFh if success cannot be guaranteed.

The LD_DRVR routines are expected to provide reliable, end-to-end message service. This means that the NDOS input or output message routing process is assured of the following when it sends or receives a message or sequence of messages:

- The message arrived with no errors.

- The message was not longer than the size specified in the Line Driver Control Block.

- A sequence of messages arrived with no duplicate messages.

- The sequence arrived in the same order they were sent.

- The sequence arrived with no missing messages.

If this cannot be guaranteed, the NIOS's output LD_DRVR on the sending node and, whenever practical, input LD_DRVR on the receiving node should return the error code to their respective message routing processes.

A "message size" field is also passed in the Line Driver Control Block. This field can be used by a receiving driver to guarantee that an incoming message is not larger than the buffer allocated to receive it. If the size of the incoming message exceeds the message size specified, the message should not be copied into the buffer, and 0FFFFh should be returned in AX.

It does not matter how messages are sent over the network. In fact, a single logical message can be broken up into a sequence of smaller packets by your NIOS. Should this be implemented, however, the receiving NIOS must be prepared to reassemble the packets with the assurances listed above. The DR Net logical message must conform to the server's expectations regarding size and contents for the server to decode and implement it properly.

Note: DR Net logical messages are always offset from the message buffer pointer in the Line Driver Control Block by a value equal to the header length specified at system generation, and recorded in the Parameter Table. This is unlike the Requester Control Block, where the pointer points directly to the DR Net logical message. In addition, space is reserved at the end of the message corresponding to the Parameter Table value for a message trailer. Figure 3-4 illustrates how the DR Net message is positioned in the message buffer.

```
+---------+-----+--+--+--+--+-----+--+--+-...-+---------+
| Message |     |     |     |     |     | Message |
| Header  | FMT | DID | SID | FNC | SIZ | ... | Trailer |
|  [from  |     |     |     |     |     |  [from  |
|Parameter|     |     |     |     |     |Parameter|
| Table]  |     |     |     |     |     | Table]  |
+---------+-----+--+--+--+--+-----+--+--+-...-+---------+
|         |                          |     |
|         |                          +-----+--> DAT
|         |     DR Net Logical Message      |
|         +---------------------------------+
+--> LDCB pointer points here
```

Figure 3-4. Dynamic Buffer allocation for network messages

The LD8DRVR send routine can write your header directly into the space allocated at the message buffer pointer without affecting the DR Net message. Accordingly, if your header requires information from the DR Net message header (such as the message format code, destination node ID, or source node ID), your LD_DRVR routine must pass over the header bytes to locate the information.

A note of warning regarding message trailers: if your LD_DRVR writes a trailer directly into the buffer that is longer than the length allotted, you might violate memory space allocated to another message.

Using a message header and trailer also affects your LD_DRVR input routine. If they are used, do not strip off the header, or expand the trailer in your input routine. DR Net does not care what values are recorded in these fields, but it does expect to find the DR Net message offset by the header value.

There are two more considerations when constructing your LD_DRVR routines.

- First, a Concurrent CP/M input LD_DRVR routine should not make extensive use of CPU time. This has a detrimental effect on your system performance. Consequently, the input LD_DRVR should make use of synchronization primitives, such as flags and queues, to minimize overhead while waiting for a message.

- Second, DR Net is capable of handling any logical message with formats 0 through 7. DR Net automatically converts all messages to format 6 or 7 and, should a response message be required, compresses the message back to the original format. Your LD_DRVR routine might be affected by these different formats, since the size and location of key fields in the DR Net logical message header may change.

Note: If your CP/M-86 requester intends to send messages to an MP/M II server running CP/NET Version 1.2, you must be aware that this server accepts only format 0 requests. Consequently, your NIOS must be able to automatically convert format 6 to format 0 requests when the destination is the MP/M II server. Accordingly, your NIOS must also expand format 1 to format 7 responses before they are transferred to the message buffer.

LD_ERR Error Recovery

Recover from failure of input or output line driver routine.

```
Entry Parameters:
    Register DX: Line Driver Control Block -- Offset
           DS: Line Driver Control Block -- Segment

 Returned Values: None
```

LD_ERR is called by an input or output message routing process only after an LD_DRVR routine has returned failure. The routine is provided so that you can recover from any consequences of the failed transaction. DR Net expects no return value from this routine.

Note: This function should not be used to resend a message. All retries should be performed by the LD_DRVR routine. Once the 0FFFFh failure code has been received by DR Net, it cannot be changed.

EOF

DNSG4.WS4   (= "DR Net System Guide", section 4)

DR Net
System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Section 4: DR Net system generation

DR Net system generation combines a DR Net invariant module with an assembled NIOS. For Concurrent CP/M based network nodes, system generation requires three steps. The same procedure for a CP/M-86 based node requires only the first two.

1. Create your NIOS.CMD file.

2. Run the DR Net system generation utility appropriate to the node's host operating system. Use GENRQR.CMD for a CP/M-86 based system, or GENNET.CMD for a Concurrent CP/M based system. (CP/M-86 requesters are through at this point.)

3. Run ADDNET.CMD to merge the DR Net module with the CCPM.SYS file.

Besides combining the DR Net invariant module with your NIOS, GENNET and GENRQR utilities also display a series of prompts. The responses to these prompts set all Parameter Table and Master requester Configuration Table values for the node. They also determine implicitly the size of the Static Buffer, and explicitly the size of the Dynamic Buffer.

This section describes how to generate a DR Net system image from your NIOS source file, and the resource files provided with DR Net. The first part describes how to create your NIOS. After that, the system generation programs used to create the different kinds of nodes -- requesters, servers, and requester/servers -- are described. The descriptions include explanation of the prompts displayed by the programs, and the criteria used to respond accurately. The possible GENNET and GENRQR error messages are also described.

4.1 Creating the NIOS

Before you run the DR Net system generation utilities, you must create a NIOS.CMD file. The discussion below assumes a NIOS written in RASM-86 assembly language. However, it is also possible to write a NIOS in a high-level language, provided it obeys the NIOS.CMD format restrictions described in Section 3.1, "NIOS structure".

Assembling the NIOS is the same as assembling any other relocatable, small model (independent code and data segments) program. Use either the RASM86.CMD and LINK86.CMD, or ASM86.CMD and GENCMD.CMD programs, to produce the NIOS.CMD file. The command sequence for each is shown in Table 4-1.

Table 4-1. Command sequence to assemble your NIOS

With ASM-86	With RASM-86
ASM86 NIOS	RASM86 NIOS
GENCMD NIOS	LINK86 NIOS

Note: If ASM86.CMD is used, the beginning of the NIOS data segment must be offset by 0100h with an ORG directive.

The NIOS.CMD file created by both procedures is merged with one of the three proprietary DR Net files by GENNET or GENRQR.

4.2 DR Net system generation

Once the NIOS is created, use the system generation utilities to combine it with a specific DR Net resource file. The utilities and system generation procedures used for Concurrent CP/M and CP/M-86 bases systems are as follows:

- For a Concurrent CP/M based system, use GENNET.CMD to create a DRNET.CMD file. Next, use ADDNET to merge this file with your CCPM.SYS Concurrent CP/M system image file.

- For a CP/M-86 based system, use GENRQR.CMD to create a DRNET.CMD file. (The DRNET.CMD is then dynamically loaded with the NETLDR.CMD utility.)

Before the DRNET.CMD file is constructed, GENNET and GENRQR display a series of prompts. Your answers fill the Parameter Table and Master Requester Configuration Table. Although these values are permanently recorded as part of the system image, they can be changed (when DR Net is loaded) by your NIOS resident NET_INIT routine. The size of the Static and Dynamic Buffers are also set from GENNET. However, these values are considerably more difficult to modify from NET_INIT.

GENNET and GENRQR share many of the same prompts. GENRQR, in fact, differs primarily in that the server related questions have been removed. Table 4-2 lists the complete set of DR Net system generation resource files and utilities for Concurrent CP/M and CP/M-86 based systems.

Table 4-2. DR Net system generation files and utilities

Concurrent CP/M based nodes

Format: File name
Description

GENNET.CMD
The utility to generate the DRNET.CMD system image file.

ADDNET.CMD
The utility to generate a CCPM.SYS file from CCPM.SYS and DRNET.CMD files.

NIOS.CMD
Your assembled and linked NIOS.

CCPM.SYS
Your Concurrent CP/M system image.

RNET.CMD
The requester-only invariant module.

SNET.CMD
The server-only invariant module.

RSNET.CMD
The requester/server invariant module.

CP/M-86 based requesters

Format: File name
    Description

GENRQR.CMD
The utility to generate the DRNET.CMD system image file.

NIOS.CMD
Your assembled and linked NIOS.

RNET.CMD
The requester function source file.

NETLDR.CMD
The user utility that loads DR Net, and attaches the network.

Note: The RNET.CMD provided for CP/M-86 based systems, and the RNET.CMD provided for Concurrent CP/M based systems, are entirely different, and must not be confused.

The files listed under Concurrent CP/M in Table 4-2 are the complete set. However, all of the resource files are not necessary to generate some DR Net systems. For instance, GENNET only requires SNET.CMD to build a server node. RNET.CMD and RSNET.CMD are not used. Similarly, only RSNET.CMD is necessary to make a requester/server node.

Figures 4-1 and 4-2 illustrates the GENNET and GENRQR dialogues, respectively. The prompt descriptions that follow are applicable to both programs. How to integrate the DRNET.CMD file with the local operating system is described in Section 4.2.4, "Generating a Concurrent CP/M based node", and Section 4.2.5, "Generating a CP/M-86 based node", at the end of this section.

Note: The square bracketed ("[" and "]") numbers shown in the leftmost columns of Figures 4-1 and 4-2 are not displayed during GENNET or GENRQR execution. They are included for reference purposes only.

```
        Please answer the following questions. All numbers are in
        hexadecimal unless preceded by a pound (#) sign. Defaults
        are in parentheses.

[ 1]   What is this machine's physical node ID?                  :
[ 2]   How many shadow processes can run on this machine?      (10):
[ 3]   How many processes can run as network requesters?       (0C):
[ 4]   How many copies of the Requester Configuration Table?   (08):
[ 5]   How many input Line Driver Control Blocks (LDCBs)?      (01):
[ 6]   How many output Line Driver Control Blocks (LDCBs)?     (01):
[ 7]   What is the first flag available for use by DR Net?     (20):
[ 8]   What is the maximum message buffer size?               (10B):
[ 9]   What is the length of any end-to-end message header?    (00):
[10]   What is the length of any end-to-end message trailer?   (00):
[11]   How big should the message buffer pool be?            (2000):
[12]   How many seconds until a transaction timeout?           (06):
[13]   Network server password (8 characters limit)      (PASSWORD):
[14]   What server should be your default server?              (00):

[15]   Note: Flags nn through nn must be used only by the network.

[16]   Is all of the information above correct?               (Y/N):

[17]   What disks do you want mapped across the network initially?
        (Use a carriage return when you are through mapping.)

        Local Disk Drive (A: through P:)?          :
[18]   Remote Disk Drive (A: through P:)?          :
[19]   Server node ID of the remote disk drive?    :

[20]   List all drives you want protected from network access
        (Use a carriage return when you are through mapping.)
        List a private drive                       :

[21]   What printers do you want mapped across the network initially?
        (Use a carriage return when you are through mapping.)

        Local Printer Number (0 through F)?        :
[22]   Remote Printer Number (0 through F)?        :
[23]   Server node ID of remote printer?           :

[24]   What queues do you want mapped across the network?
        (Use a carriage return when you are through mapping.)

        Local Queue Name (8 characters or less)?   :
[25]   Remote Queue Name (8 characters or less)?   :
[26]   Server node ID of the remote queue?         :
```

```
[27]   Are all of the mappings above correct?     (Y/N):

       Building DRNET.CMD ...
       Network System Generation Complete
```

Figure 4-1. GENNET dialogue

```
       Please answer the following questions. All numbers are in
       hexadecimal unless preceded by a pound (#) sign. Defaults
       are in parentheses.

[ 1]   What is this machine's physical node ID?                   :
[ 5]   How many input Line Driver Control Blocks (LDCBs)?        (01):
[ 6]   How many output Line Driver Control Blocks (LDCBs)?       (01):
[ 8]   What is the maximum message buffer size?                (10B):
[ 9]   What is the length of any end-to-end message header?      (00):
[10]   What is the length of any end-to-end message trailer?     (00):
[11]   How big should the message buffer pool be?              (2000):
[13]   Network server password (8 characters limit)        (PASSWORD):
[14]   What server should be your default server?                (00):

[16]   Is all of the information above correct?                 (Y/N):

[17]   What disks do you want mapped across the network initially?
       (Use a carriage return when you are through mapping.)

       Local Disk Drive (A: through P:)?              :
[18]   Remote Disk Drive (A: through P:)?             :
[19]   Server node ID of the remote disk drive?       :

[21]   What printers do you want mapped across the network initially?
       (Use a carriage return when you are through mapping.)

       Local Printer Number (0 through F)?            :
[22]   Remote Printer Number (0 through F)?           :
[23]   Server node ID of remote printer?              :

[24]   What queues do you want mapped across the network?
       (Use a carriage return when you are through mapping.)

       Local Queue Name (8 characters or less)?       :
[25]   Remote Queue Name (8 characters or less)?      :
[26]   Server node ID of the remote queue?            :

[27]   Are all of the mappings above correct?     (Y/N):

       Building DRNET.CMD ...
       Network System Generation Complete
```

Figure 4-2. GENRQR dialogue

4.2.1 GENNET and GENRQR prompt descriptions

GENNET and GENRQR run under Concurrent CP/M and CP/M-86. They display the prompts shown in Figures 4-1 and 4-2, respectively, that allow you to set the values for the node's Parameter Table and Master Requester Configuration Table, and specify the size of the Static and Dynamic Buffers. All entries are interpreted as hexadecimal values. The values shown in parenthesis in the two figures are the DR Net defaults that are selected by entering a carriage return. Two types of error messages can be displayed. First, an improper entry can cause an immediate error message. Second, GENNET calculations for Dynamic and Static Buffer sizes can generate errors.

In the prompt descriptions that follow, the numbers in the lefthand column reference the square bracketed ("[" and "]") numbers in Figure 4-1 and 4-2. These numbers have no other significance. Following the prompt explanations, the error messages displayed by GENNET and GENRQR are described.

Table 4-3. GENNET and GENRQR prompts and descriptions

Format: Message
Meaning

[1] What is this machine's physical node ID?
The unique hex ID number of this node. Values available are 00h through 0FEh. Nodes need not be numbered sequentially.

[2] How many shadow processes can run on this machine? (10)
This prompt determines whether or not the node functions as a server on the network. Notice that the question does not ask how many requesters can access it. Rather, it asks for the number of shadow processes for which it should reserve memory resources. The difference is important, because one requester can have multiple concurrent processes, each using the network. Each server requires three pages (768 bytes) of Static Buffer memory.

[3] How many processes can run as network requesters? (0C)
This prompt determines whether or not the node functions as a requester on the network. Notice that this question has nothing to do with the number of shadow processes this node can talk to at any one time. Instead, the number specifies how many local processes can be attached to the network at any one time. The value entered impacts upon system memory resources, because each requester process requires 704 bytes of memory, not including the standard Process Descriptor (PD) and UDA requirements, when it is attached.

[4] How many copies of the Requester Configuration Table? (08)
Every process family attached to the network has at least one Requester Configuration Table. When a process already attached calls the DR Net N_ATTACH function, it gets another copy. The value entered stipulates the maximum number of copies of the Requester Configuration Tables to be allowed, so that memory resources can be allocated. Each copy of this table requires 422 bytes.

[5] How many input Line Driver Control Blocks (LDCBs)? (01)
It is essential that the number entered in response to this prompt corresponds exactly to the number of input Line Driver Control Blocks in the NIOS.

[6]   How many output Line Driver Control Blocks (LDCBs)? (01)
It is esential that the number entered in response to this prompt corresponds exactly to the number of output Line Driver Control Blocks in the NIOS.

[7]   What is the first flag available for use by DR Net? (20)
The total number of flags required is equal to the number of shadow processes specified in the second prompt, plus the number of requester processes specified in the third prompt, plus 1. GENNET begins allocating flags from the value entered in response to this prompt, and stops when every process has been accounted for, or the value 0FFh has been exceeded. A message appears on the console later on in GENNET (see the description of prompt 15 below) to indicate which flags are allocated for DR Net's use. These flags cannot be used by another system process.

Note: GENNET assumes that the entire range of flags has already been allocated in GENCCPM. If this is not true, generate another CCPM.SYS file with enough flags before running ADDNET as described below.

[8]   What is the maximum message buffer size? (10B)
This value sets the maximum length of the DR Net logical message. The input message routing process refers to this value and the values entered for the optional message and trailer when it reserves message buffer space to accomodate an incoming message. Note that the default value is also the minimum. Any size smaller than 010Bh is automatically rounded up to this value.

[9]   What is the length of any end-to-end message header? (00)
This value stipulates the offset of the DR Net message from the message buffer pointer passed to the NIOS in the Line Driver Control Block. If 0 is entered, the first byte (the FMT byte) of the message is at the pointer address.

[10]   What is the length of any end-to-end message trailer? (00)
This value indicates the amount of space allocated at the end of each DR Net message in the Dynamic Buffer.

[11]   How big should the message buffer pool be? (2000)
This prompt directly specified the size of DR Net's Dynamic Buffer. The Dynamic Buffer is used only for temporary message storage, as they are transferred to and from the network controller. How much memory to allocate to the Dynamic Buffer depends not only upon the message size, but also the projected number of messages active in the buffer at one time. You will find below formulas for determining the minimum and maximum buffer value, given these considerations. The value entered cannot exceed 64 KB, and is routnded up, so that there are an integral number of Allocation Units. (Allocation Units are described below.)

Minimum Dynamic Buffer value is: Message Block * (#IN + 1)

Maximum Dynamic Buffer value is: Message Block * (R + S + #IN + 1)

The variables used in these equations are defined as follows:

- Message Block

Maximum message length + header + trailer (see prompts 8, 9, and 10) in whole Allocation Units. (Message space is reserved in the Dynamic Buffer in whole Allocation Units. Allocation Units are explained in the description of Figure 4-3 below.)

- #IN
  The number of input Line Driver Control Blocks (see prompt 5).

- R
  The number of requester processes (see prompt 3).

- S
  The number of server, shadow processes (see prompt 2).

Figure 4-3, below, illustrates the components of the Dynamic Buffer.

The factors that decide whether the minimum, maximum, or a value in between should be entered are weighted by concerns for memory resources and the rate of message throughput. The minimum value is enough to prevent message buffer deadlocks. (For a description of deadlocks, see Section 4.2.3, "Error messages", below.) This optimizes memory allocation, but can result in wait periods for an open buffer when message traffic is heavy. Use the minimum in requesters and servers where only occasional use is made of the network. Where you foresee heavy traffic, a buffer size toward the maximum value should be selected. This value impacts upon your memory resources, but it provides unimpeded message throughout.

[12] How many seconds until a transaction timeout? (06)
The value determines how many seconds are allowed to elapse for a transaction time-out. (The transaction time-out is described in the NDOS and watchdog process descriptions, in sections 2.1.2 and 2.1.6, respectively.

The transaction time-out value specified should be long enough that there is absolutely no chance that the response will be received after the time-out has occurred. If a response is ever received after the corresponding time-out has been signalled, DR Net can behave unreliably.

Note: To disable the watchdog process's transaction time-out, enter a 0 in response to this prompt. This causes a requester to wait indefinitely for a response, and is therefore not advised.

[13] Network server password (8 characters limit) (PASSWORD)
In a requester node, the value entered in response to this prompt is used by utilities such as LOGON.CMD when no password is explicitly specified. In a server node, this value is compared against the password in the N_LOGON message before access is allowed. In a simultaneous requester/server, this value serves both purposes.

[14] What server should be your default server? (00)
This hex value indicates a requester node's default server. (It has no meaning in server-only nodes.) The default server has significance in two areas. First, the default server is assumed to have the NAMSVR.DAT file used by many DR Net utilities for substituting logical ASCII names for hex DR Net node ID numbers. Second, the default server is the server node referenced when the

user omits the server ID number from the command line of a utility.

[15]   Note: Flags nn through nn must be used only by the network.
This statement is displayed as a convenience to indicate the range of flags that are reserved for exclusive use by DR Net. "nn" in both contexts represents the first and last flag numbers of the range. 5the initial flag is evoked in prompt 7.)

[16]   Is all of the information above correct? (Y/N)
An "N" response returns you to prompt 1 above. All previous entries are ignored. Only a "Y" response allows you to proceed. Note that upper- and lower-case "N" and "Y" are accepted.

The previous prompt concludes the GENNET questions that set the Parameter Table, Dynamic Buffer, and, implictly, Static Buffer values. The remainder of prompts fill in the Master Requester Configuration Table's map entries. The values entered in response to these questions are permanently recorded as part of the DR Net system image. You can change these values when DR Net is loaded with a NET_INIT routine, or users can temporarily change the disk drive and list device maps with the NET and LOCAl utilities. In both cases, however, the values in the system image are not affected.

GENNET cycles repetitively through the next three prompts, and accepts a different mapping for disk drives A through P. This same style of prompts is displayed to map list devices and queues. To terminate any of the mapping prompts for drives, list devices, or queues, enter a carriage return in response to the first prompt of the series.

[17]   What disks do you want mapped across the network initially?
    (Use a carriage return when you are through mapping.)

    Local Disk Drive (A: through P:)?

Enter the name of a local drive to which you wish to refer when accessing a networked disk drive. This could be a local drive name for which you have a physical drive (drive B for example), or a virtual drive. In either case, all references to that drive are trapped and sent to the drive specified by the next two prompts. Enter a carriage return to complete the mapping of disk drives.

[18]   Remote Disk Drive (A: through P:)?
Enter the logical name of the disk drive that you want to serve as the drive specified in the previous prompt.

[19]   Server node ID of the remote disk drive?
Enter the hex ID node number of the node with the replacement drive. Node names cannot be used in GENNET; only hex numbers in the range 00h to 0FEh are accepted by this program.

[20]   List all drives you want protected from network access
    (Use a carriage return when you are through mapping.)
    List a private drive (A: through P:)

Drive names entered in response to this prompt are withheld from access by requesters. After each entry, the prompt is redisplayed asking for another. A carriage return terminates this prompt, and allows you to proceed. Attempts by any requesters to access a drive marked as private return a BDOS select error, not a network extended error.

[21] What printers do you want mapped across the network initially?
(Use a carriage return when you are through mapping.)

Local Printer Number (0 through F)?

Local printers, like drives, are mapped individually to a specific, remote list device. A series of three prompts very similar to those displayed for drives (only the resource is different) are displayed, allowing you to map up to 16 list devices. In response to the first prompt, enter the physical list device number of the local printer you wish to refer to when accessing the networked printer. Any hexadecimal value from 0 through F is valid. Enter a carriage return in response to this prompt to terminate list device mapping.

[22] Remote Printer Number (0 through F)?

Enter the physical list device number of the replacement printer. Any hexadecimal value from 0 through F is valid.

[23] Server node ID of remote printer?
Enter the hex ID number of the server node to which the replacement printer is attached. Only hex values 00h through 0FEh are valid; node names cannot be used.

[24] What queues do you want mapped across the network?
(Use a carriage return when you are through mapping.)

Local Queue Name (8 characters or less)?

Local queues, like list devices and disk drives, are mapped individually to a specific, remote queue. A series of three prompts just like those for printers and drives are displayed, allowing you to map up to 16 different queues. In response to the first prompt, enter the name of the local queue you wish to refer to when accessing the remote queue. Because both upper- and lowercase queue names are valid, be sure to use the proper case for each letter, and observe the 8 character limit. Enter a carriage return in response to this prompt to terminate queue mapping.

[25] Remote Queue Name (8 characters or less)?
Enter the name of the replacement queue, being careful to match the case of each character. The name need not be the same as the Local Queue Name entered above. Again, there is an 8 character limit.

[26] Server node ID of the remote queue?
Enter the hex ID number of the server node where the replacement queue is located. Only hex values 00h through 0FEh are valid; node names cannot be used.

[27] Are all of the mappings above correct? (Y/N)
The last prompt allows you to reconsider your entries for the Master Requester

Configuration Table. An "N" response returns you to the first prompt of the
drive mapping series. Otherwise, a "Y" response proceeds with the building of
the DRNET.CMD file. This prompt accepts either upper- or lowercase responses.

```
                       Dynamic Buffer
                  +--            +-----+
                  |              |     |
                  |              +-----+-+
                  |              |     | |
                  |   Actual     +-+-----+ | Message Block
      Total length   |   Message     | |     | |
      set in GENNET --+  Requirement  +-+-----+-+
                  |              |     |
                  |   Actual     +-+-----+-+
                  |   Message     | |     | | Allocation Unit
                  |  Requirement  +-+-----+-+
                  |              |     |
                  +--            +-----+

Where:
                  Maximum Message Length + Header + Trailer
      Message Block = ---------------------------------------
                  Allocation Unit

      Allocation Unit = 010Bh + Header + Trailer

      Figure 4-3. Dynamic Buffer allocation
```

Notice in Figure 4-3 that the entire buffer is divided into Allocation Units.
An Allocation Unit is equal to 010Bh (267) bytes, plus the length of the
message header and message trailer specified in the Parameter Table. As
mentioned above, DR Net allocates space in the Dynamic Buffer in whole
Allocation Units only. Consequently, DR Net's internal message allocation
mechanism does on occasion allocate more buffer space than actually required
by the message.

4.2.2 System Manager information

Because a computer network requires a higher order of care and attention than
a single user system, we have compiled the descriptions of DR Net installation
and maintenance into the "DR Net System Manager's Guide". This manual explains
in general terms DR Net architecture, network topology, software installation,
and other site dependent subjects. It also describes DR Net installation using
the GENNET or GENRQR utilities.

As mentioned elsewhere, the GENNET and GENRQR programs fill in values to the
Requester Configuration Table (RCT) and Parameter Table. The RCT values are
the site dependent parameters that establish the node's resource map. On the
other hand, many Parameter Table values are DR Net implementation dependent,
and an explanation of their options is beyond the scope of the "System
Manager's Guide". Consequently, the proper values for these parameters are not

apparent to the installer who is unfamiliar with your implementation.

Table 4-4 below highlights parameters that are not always obvious to the system installer. There are three ways to ensure that the proper values are incorporated:

1) Provide the system installer with a list of the prompts, with the proper values.

2) Use the NIOS-resident NET_INIT function to write the appropriate values into the table when DR Net is loaded.

3) Create a small GENNET program that operates only on the user-modifiable parameters of the finished system image. Distribute this program with the network systems, instead of the GENNET and ADDNET programs supplied by Digital Research.

If the second option is selected, it does not matter what the system installer enters, values written by your NET_INIT routine override values entered by GENNET or GENRQR. Your NET_INIT routine has no rules regarding how the values are determined. For example, the data can be set through special network transactions, read from a local file, or entered by the user from console interaction.

Note: The list in Table 4-4 is by no means all inclusive. To ensure that your DR Net system is installed correctly, it might be necessary to map queues, specify an exact Dynamic Buffer length, set the number of server and requester processes, and so forth. Whatever is necessary, all Master Requester Configuration Table and Parameter Table values can be incorporated with the NET_INIT routine.

Table 4-4. Implementation bound Parameter Table values

Prompt Number*	Description
5	The number of input Line Driver Control Blocks
6	The number of output Line Driver Control Blocks
7	The first flag available to DR Net
8	The maximum message buffer size
9	The message header length
10	The message trailer length

* = This number refers to the prompt explanation immediately above. Those numbers are not displayed by either GENNET or GENRQR.

4.2.3 Error messages

The system generation utilities prevent you from entering numbers outside the range of acceptable values, and from specifying a configuration that cannot be accommodated. Following is the list of error messages that can be encountered. Preceding the message is the number of the prompt with which it is associated.

(The numbers listed apply only to the above prompt descriptions. These numbers are not displayed by the program.)

The two types of error messages are listed in Table 4-5. The first type is characterized by the immediate display of the message after the offending value has been entered. This class is further distinguished in that the prompt is redisplayed to solicit the entry of another value.

For all errors of this type, the number in the lefthand column corresponds to the numbers in Figure 4-1 and 4-2.

The other type of error message occurs after a series of related prompts, and indicates a more profound configuration problem. When this happens, it might be necessary to repeat the Parameter Table or Requester Configuration Table portion of the GENNET prompts. This second type of error is preceded by an asterisk ("*"), rather than a number, in the list below.

Table 4-5. GENNET and GENRQR prompt-phase error messages

Format: Number and Message
Meaning

Invalid number.
No specific prompt is associated with this error message. It is displayed whenever an illegal number, such as 0FMh, is entered. This message is not displayed, however, if a valid number is displayed that is out of the range allowed for that parameter.

1. Please specify a node ID in the range 00 - FE.
The hex node number entered was 0FFh or greater. Re-enter the node number with a number in the given range.

2. Can't have more than 83 shadow processes.
More than 53h shadow processes were requested. Re-enter the parameter with a value less than or equal to 53h.

3. Can't have more than 90 requester processes.
More than 5Ah networked, requester processes were requested. Re-enter the parameter with a value less than or equal to 5Ah.

4. Can't have more than 149 local RCTs.
More than 95h copies of the Requester Configuration Table were ordered. Re-enter the parameter with a value less than or equal to 95h.

5. Can't have more than 16 input line drivers.
More than 10h input Line Driver Control Blocks were specified. Re-enter the parameter with a value less than or equal to 10h.

6. Can't have more than 16 output line drivers.
More than 10h output Line Driver Control Blocks were specified. Re-enter the parameter with a value less than or equal to 10h.

* Parameters given need more than 64K of static storage.
The memory needs required to support the number of requester and shadow

processes, of Requester Configuration Tables, and of input and output message routing processes, exceeds 64 KB. One or more of these values will have to be reduced. See the equation for the Static Buffer in Section 1, "Hardware requirements", for more information on DR Net's memory requirements.

7. Flags must be in the range 00 - FF.
The first flag entered was out of the range shown. Re-enter the parameter, and specify a beginning flag number within the range.

* Too many flags are required to run this configuration.
The total number of requester and shadow processe specified requires a flag number in excess of 0FFh. To correct this situation, enter a lower beginning flag number, or specify fewer requester and/or shadow processes.

Note: This last message can also occur in response to an erroneous answer to prompt 7.

11. Can't have a buffer pool larger than 64K.
The Dynamic Buffer specified was greater than 0FFFFh. Re-enter the parameter with a number equal to or below this value.

* Buffer pool isn't big enough to guarantee no deadlocks.
GENNET uses the minimum value formula shown in Table 4-3 to determine if the Dynamic Buffer specified in prompt 11 is large enough to prevent deadlocks. A deadlock is a situation where there are no message buffers available and a process is waiting to receive a message, so that it can deallocate a message buffer. After this message is displayed, you are returned to the prompt 1.

14. Passwords should be 8 characters or less.
The password entered was in excess of 8 characters in length. Re-ebter the password, and observe the 8 character limit.

15. Node IDs must be in the range 00 - FE.
The hex node ID number specified was FF or greater. Re-enter the parameter with a value in the range shown.

16. Please type yes (Y) or no (N).
This prompt, and the last one, respond to an upper- or lowercase "Y" or "N" only. Any other response results in this error message.

17. Disk drives should be in the range A: through P:.
This error message can also result from an errant entry to prompts 18 and 20. It indicates that the entry was either a number, or outside the range shown. Re-enter the value, and be sure to use the logical drive name, rather than a number. Colons (":") are not necessary.

21. List devices must be in the range 0 - F to be networked.
This message indicates that the entry was a number outside the range shown. Re-enter the value with a number within the range.

24. Queue names should be 8 characters or less.
The queue name entered was in excess of 8 characters in length. Re-enter the name, and observe the 8 character limit. This message can also result from responses to prompt 25 that exceed 8 characters.

After the prompt phase has been completed, GENNET and GENRQR display the message

    Building DRNET.CMD ...

and construct the DR Net system image file. Table 4-6 lists and explains the error messages associated with this phase of DR Net system generation.

Table 4-6. GENNET and GENRQR file system error messages

Message	Meaning
Can't open network or NIOS input file.	GENNET cannot find or open either the RNET.CMD, SNET.CMD, RSNET.CMD, or NIOS.CMD file. Confirm that these files are on the current disk, and ensure that there is nothing to prevent them from being opened.
Can't allocate space to build the output file.	GENNET has run out of memory. The utility will have to be run in a segment with a larger memory allocation, under a smaller operating system, or on a machine with more memory.
Can't create the DRNET.CMD output file.	There is something that is preventing the file DRNET.CMD from being written to the disk. Corroborate that the drive is not set to read only, that the write protect notch is covered, and that there is enough space in the directory.
Error reading one of the input files.	A BDOS error was encountered while reading RNET.CMD, SNET.CMD, RSNET.CMD, or NIOS.CMD. Verify that these files do not contain hard errors.
Unexpected error writing the DRNET.CMD output file.	A BDOS error was encountered while writing the DRNET.CMD file. Verify that the disk has enough space.

4.2.4 Generating a Concurrent CP/M based node

GENNET success is indicated by the message

    Network System Generation Complete

and the return of the command prompt. The file DRNET.CMD is recorded on the disk in the current drive. This does not, however, complete the installation of DR Net with the Concurrent CP/M system. Before the system can be used as a network node, you must run ADDNET.

The ADDNET utility merges the newly created DRNET.CMD file with your CCPM.SYS file. The result is a new CCPM.SYS file with DR Net installed. ADDNET has a single option that determines what is to happen to your original CCPM.SYS file.

If you enter just

```
        ADDNET
```

your original CCPM.SYS is consumed by ADDNET, and disappears, giving way to the new CCPM.SYS with DRNET.CMD installed.

If you enter

```
        ADDNET <filename.typ>
```

ADDNET assumes that the filename specified contains a Concurrent CP/M system image identical to the one found in a CCPM.SYS without DR Net installed. The output from the ADDNET is still written to a file named CCPM.SYS. Rename your CCPM.SYS file, and use the optional ADDNET command line to preserve your original Concurrent CP/M system image.

While ADDNET is running, the following messages are displayed to assure you that the program is progressing without error. Table 4-7 lists the error messages, and their meanings, displayed when ADDNET encounters a difficulty, and cannot proceed.

```
        Reading in system image
        Reading in network system image
        Writing completed system to CCPM.SYS

        Addition of network to system image is complete
```

The new CCPM.SYS file is loaded by the Concurrent CP/M bootstrap loader, and leaves the user with an initialized network node. This means that the server function is available but DR Net must first be attached using NETON before the operator can use the requester function.

Table 4-7. ADDNET error messages

Format: Error
        Meaning

ADDNET has already been run on this system file.
This indicates that the CCPM.SYS file provided to ADDNET was previously merged with the DRNET.CMD.

Can't create output CCPM.SYS file.
This indicates that the destination disk, or the original CCPM.SYS, was set to read only, or that there was no space left in the directory to create a new CCPM.SYS file.

Can't open an input system file.
ADDnet could not open the CCPM.SYS file or the file specified in the ADDNET command line. Confirm that the file is present on the current disk, and that there are no restrictions preventing ADDNET from opening it.

Can't open the network input file DRNET.CMD.
ADDNET could not open or find the DRNET.CMD DR Net system image file. Confirm that the DRNET.CMD file is present on the current disk, and that there is nothing preventing ADDNET from opening it.

Encountered an error reading input file.
This indicates that an unrecoverable BDOS error occurred while reading either the CCPM.SYS or DRNET.CMD file.

Encountered an error writing CCPM.SYS.
This indicates a disk full condition, or a unrecoverable BDOS error occurred. Confirm that there is room on the disk at least equal to the total of your CCPM.SYS and DRNET.CMD files.

Not enough memory to add the network to the system.
This message indicates that there is not enough memory for ADDNET to run and merge the two system files. ADDNET requires a total of the following amounts of memory to complete:

- 31 KB of memory for the ADDNET program
- memory to accomodate the entire CCPM.SYS file
- memory to accomodate the entire DRNET.CMD file
- 5 KB to 10 KB for overhead

4.2.5 Generating a CP/M-86 based node

GENRQR success is indicated by the message

```
        Network System Generation Complete
```

and the return of the command prompt. The file DRNET.CMD is recorded on the disk in the current drive. This completes the DR Net system generation procedure for a CP/M-86 based requester. Unlike the Concurrent CP/M based nodes, the host operating system and DR Net system images are not merged with the ADDNET program.

To load and attach DR Net in a CP/M-86 based node, the NETLDR program is used. This reads the DRNET.CMD file, and initializes the network, as described in the description of the CP/M-86 requester in Section 2.

EOF

DNSG5.WS4   (= "DR Net System Guide", appendixes)

DR Net
System Guide

First Edition: March 1984

(Retyped by Emmanuel ROCHE.)

Appendixes

Appendix A: Network message contents

Messages output by DR Net requesters and servers have a fixed format. In turn, DR Net requesters and servers expect all messages received from the network to be in this same format. This appendix describes the DR Net logical message format and the message contents for all system calls that can be executed remotely.

A.1 DR Net logical message format

The DR Net logical message is the standard medium of exchange between requesters and servers. The requester module automatically gathers all pertinent data, and assembles the message into this format when a system call references a remote resource. All the information required by the server to perform the function is transferred in the DR Net.

The same format is used for servers' response messages. In this case, the return value and all data requested in the initial call are returned, so that the requester can present the response according to the calling convention.

Figure A-1 illustrates the format of the DR Net message. Each field is described in Table A-1.

```
             0     1     2     3     4     5+
Formats   +-----+-----+-----+-----+-----+-----+     +-----+
00h, 01h  | FMT | DID | SID | FNC | SIZ | DAT | ... | DAT |
          +-----+-----+-----+-----+-----+-----+     +-----+

             0     1     2     3     4     5     6     7     8+
Formats   +-----+-----+-----+-----+-----+-----+-----+-----+-----+     +-----+
06h, 07h  | FMT |    DID    |    SID    | FNC |    SIZ    | DAT | ... | DAT |
          +-----+-----+-----+-----+-----+-----+-----+-----+-----+     +-----+
```

Figure A-1. DR Net logical message formats

Table A-1. DR Net logical message field descriptions

Format: Field
 Description

FMT -- The format code
A message's format code indicates the number of bytes per field, and whether the message is a request or a response. Table A-2 below lists the different FMT codes. Even-number FMT codes indicate that the message is a request; odd-number FMT codes indicate that the message is a response.

DID -- Destination node ID number
The DID field is a 1- or 2-byte value that indicates the message's destination node, or its destination node and process. In a single-byte DID, only the destination's hexadecimal ID number is transferred. In a 2-byte DID, the node ID is recorded in the low-order byte, and the contents of the high-order byte depend upon whether the message is a request or a response. In a request, the high-order byte is meaningless. In a response, the high-order byte contains the requester process's arbitrarily assigned process ID.

SID -- Source node ID number
The SID field is a 1- or 2-byte value that indicates the message's source node, or its source node and process. In a single-byte SID, only the source node's hexadecimal ID number is transferred. In a 2-byte SID, the node ID is recorded in the low-order byte, and the contents of the high-order byte depend upon whether the message is a request or a response. In a request, the high-order byte contains the requester process's arbitrarily assigned process ID. In a response, this field is meaningless.

FNC -- System call function number
The FNC field is always a single byte, and contains the system call's hexadecimal number.

SIZ -- Data field size
The SIZ field is a 1- or 2-byte value that indicates the length of the data field that follows. This number is always the number of bytes in the field minus 1.

DAT -- Data field
The DAT field is variable length, and contains all of the input values and data required by the server to perform the function, or all the return values and data returned by the function.

Table A-2. DR Net logical message formats

FMT	Number of bytes in					
Code	DID	SID	FNC	SIZ	DAT	Description
00	1	1	1	1	1-256	CP/NET V1.2 Request
01	1	1	1	1	1-256	CP/NET V1.2 Response
02	1	1	1	2	1-65536	Not used (Request)
03	1	1	1	2	1-65536	Not used (Response)

04	2	2	1	1	1-256	Not used (Request)
05	2	2	1	1	1-256	Not used (Response)
06	2	2	1	2	1-65536	Concurrent CP/M Request
07	2	2	1	2	1-65536	Concurrent CP/M Response
08-127						Undefined
128-255						User definable

Equally as important to the server and requester alike as the DR Net message format is the format of the DAT filed. Table A-3 lists the message contents for all system calls that can be networked.

A.2 Special characters, symbols, and terms in Table A-3

The following characters, symbols, and terms are used in the following DAT field descriptions:

xx
Appears occasionally in the DAT field descriptions, to indicate that the value is irrelevant.

- EE -
Indicates that the function can return BDOS extended errors ("EE"), as well as extended network errors 0CFFh, 0DFFh, or 0EFFh. Any message can return an extended network error.

+
Appears as a suffix to the SIZ field value, and indicates that the value is variable. When it is used, consider the SIZ value shown to indicate the minimum.

Simulation Count
The largest number of 128-byte records that can fit in a single transaction. This value is calculated by the NDOS when function 44, F_MULTISEC, is called to reconcile the size of the DR Net message buffer (through which all messages flow to and from the network) with the length of the multisector data block. Dividing the Multisector Data Block by the Simulation Count gives you the number of network transactions required to complete a multisector data transfer.

Table A-3. DR Net logical message contents

System call	FMT	FNC	SIZ	DAT
* 3: C_RAWIN	0	03	0000	0000-0000 Server Console Number
	1	03	0000	0000-0000 Character Input
* 4: C_RAWOUT	0	04	0001	0000-0000 Server Console Number

```
                          0001-0001 Raw Character to Output
                1   04  0000    0000-0000 00

 5: L_WRITE          0   05  00nn    0000-0000 Server List Number
   - EE -                       0001-00nn Characters To Be Listed
                1   05  0000    0000-0000 00

                Note: nn = 01h to 80h.

* 11: C_STAT           0   0B  0000    0000-0000 Server Console Number
                1   0B  0000    0000-0000 Console Status Byte

* These functions are supported by Concurrent CP/M servers for
  CP/NET Version 1.2 compatibility only. They are never generated
  by a DR Net requester.

  14: DRV_SET          0   0E  0000    0000-0000 Selected Disk
   - EE -           1   0E  0000    0000-0000 Return Code

  15: F_OPEN           0   0F  002C    0000-0000 User Number
                          0001-0024 FCB
   - EE -                       0025-002C Password
                1   0F  0024    0000-0000 Directory/Return Code
                          0001-0024 Opened FCB

  16: F_CLOSE          0   10  002C    0000-0000 User Number
                          0001-0024 FCB
   - EE -                       0025-002C Not Used
                1   10  0024    0000-0000 Directory/Return Code
                          0001-0024 FCB

  17: F_SFIRST         0   11  0025    0000-0000 Current Disk
                          0001-0001 User Number
   - EE -                       0002-0025 Search FCB
                1   11  0020    0000-0000 Directory/Return Code
                          0001-0020 Directory Entry

                Note: "Current Disk" is valid only when there is
                      a "?" in the search FCB drive byte field.

  18: F_SNEXT          0   12  0001    0000-0000 xx
   - EE -           1   12  0020    0000-0000 Directory/Return Code
                          0001-0020 Directory Entry

  19: F_DELETE         0   13  002C    0000-0000 User Number
                          0001-0024 FCB
   - EE -                       0025-002C Password
                1   13  0000    0000-0000 Directory/Return Code

  20: F_READ           0   14  0024    0000-0000 User Number
                          0001-0024 FCB
   - EE -           1   14  00A5+   0000-0000 Return Code
                          0001-0024 FCB
                          0025-xxxx Data That Was Read
```

```
                    yyyy-yyyy Multisector Transfer
                              Return Code

               Note: xxxx = 00A4 + (80h * (Simulation Count - 1) )
                     yyyy = xxxx + 1

21: F_WRITE        0   15  00A4+   0000-0000 User Number
                               0001-0024 FCB
  - EE -                           0025-xxxx Data To Be Written
               1   15  0025    0000-0000 Return Code
                               0001-0024 FCB
                               0025-0025 Multisector Transfer
                                         Return Code

               Note: xxxx = 00A4 + (80h * (Simulation Count - 1) )

22: F_MAKE         0   16  002D    0000-0000 User Number
                               0001-0024 FCB
  - EE -                           0025-002C Password
                               002D-002D Password Mode Field
               1   16  0001    0000-0000 Directory/Return Code
                               0001-0024 FCB

23: F_RENAME       0   17  002C    0000-0000 User Number
                               0001-0024 FCB in RENAME Format
  - EE -                           0025-002C Password
               1   17  0000    0000-0000 Directory/Return Code

24: DRV_LOGINVEC     0   18  0000    0000-0000 xx
               1   18  0001    0000-0001 Log-in Vector

27: DRV_ALLOCVEC     0   1B  0000    0000-0000 Current Disk
  - EE -           1   1B  nnnn    0000-nnnn Allocation Vector

               Note: The size of the allocation vector is
                     dependent on the current disk's DPB.

28: DRV_SETRO        0   1C  0000    0000-0000 Current Disk
               1   1C  0000    0000-0000 00

29: DRV_ROVEC        0   1D  0000    0000-0000 xx
               1   1D  0001    0000-0001 Read/Only Vector

30: F_ATTRIB       0   1E  002C    0000-0000 User Number
                               0001-0024 FCB with File Attributes
  - EE -                           0025-002C Password
               1   1E  0000    0000-0000 Directory/Return Code

31: DRV_DPB        0   1F  0000    0000-0000 Current Disk
  - EE -           1   1F  000F    0000-000F Disk Parameter Block

33: F_READRAND     0   21  0024    0000-0000 User Number
                               0001-0024 FCB
  - EE -           1   21  00A5+   0000-0000 Return Code
```

```
                                0001-0024 FCB
                                0025-xxxx Data That Was Read
                                yyyy-yyyy Multisector Transfer
                                          Return Code

                  Note: xxxx = 00A4 + (80h * (Simulation Count - 1) )
                        yyyy = xxxx + 1

34: F_WRITERAND       0   22  0024    0000-0000 User Number
                                0001-0024 FCB
  - EE -                            0025-xxxx Data To Be Written
                  1   22  0024    0000-0000 Return Code
                                0001-0024 FCB

                  Note: xxx = 00A4 + (80h * (Simulation Count - 1) )

35: F_SIZE            0   23  0024    0000-0000 User Number
                                0001-0024 FCB
  - EE -              1   23  0024    0000-0000 Return Code
                                0001-0024 FCB

36: DRV_RANDREC       0   24  0024    0000-0000 User Number
                                0001-0024 FCB
  - EE -              1   24  0024    0000-0000 Return Code
                                0001-0024 FCB

37: DRV_RESET         0   25  0001    0000-0001 Drive Vector
                  1   25  0000    0000-0000 Return Code

38: DRC_ACCESS        0   26  0001    0000-0001 Drive Vector
  - EE -              1   26  0000    0000-0000 Return Code

39: DRV_FREE          0   27  0001    0000-0001 Drive Vector
                  1   27  0000    0000-0000 Return Code

40: F_WRITEZF         0   28  00A4+   0000-0000 User Number
                                0001-0024 FCB
  - EE -                            0025-xxxx Data To Be Written
                  1   28  0024    0000-0000 Return Code
                                0001-0024 FCB

                  Note: xxxx = 00A4 + (80h * (Simulation Count - 1) )

42: F_LOCK            0   2A  0026    0000-0000 User Number
                                0001-0024 FCB
  - EE -                            0025-0026 File ID
                  1   2A  0024    0000-0000 Return Code
                                0001-0024 FCB

43: F_UNLOCK          0   2B  0026    0000-0000 User Number
                                0001-0024 FCB
  - EE -                            0025-0026 File ID
                  1   2B  0024    0000-0000 Return Code
                                0001-0024 FCB
```

```
44: F_MULTISEC       0   2C  0001    0000-0000 Multisector Count Requested
                                     0001-0001 Count Used for Simulating
                                               Multisector I/O Across
                                               Network
                     1   2C  0000    0000-0000 Return Code

46: DRV_SPACE        0   2E  0000    0000-0000 Drive ID
                     1   2E  0003    0000-0000 Return Code
  - EE -                             0001-0003 Number of Free Records

64: N_LOGON          0   40  0012    0000-0007 Server Password
                                     0008-0009 Process ID
                                     000A-0011 BDOS Default Password
                                     0012-0012 Compatibility Attributes
                                     0013-0013 Version
                                     0014-0014 First Log Flag
                     1   40  0001    0000-0000 Return Code
                                     0001-0001 Reserved for System Use

                     Note: The Process ID is a number arbitrarily
                           assigned by the requester. It is not
                           the process descriptor address.

65: N_LOGOFF         0   41  0009    0000-0007 Unused
                                     0008-0009 Process ID
                     1   41  0000    0000-0000 Return Code

                     Note: The Process ID in this message is a
                           number arbitrarily assigned when the
                           process logged on.

70: N_ATTRIB         0   46  0000    0000-0000 Compatibility Attributes
                     1   46  0000    0000-0000 xx

                     Note: For CP/M-86 requesters only.

71: N_SCT            0   47  0000    0000-0000 xx
                     1   47  004C    0000-0000 Server Temporary File Drive
                                     0001-0001 Server Network Status Byte
                                     0002-0002 Server Node ID
                                     0003-0003 Maximum Possible Req.
                                     0004-0004 Num. of Req. Logged On
                                     0005-0006 Requester Log-on Vector
                                     0007-0017 IDs of 16 Requesters
                                     0017-0018 Requester Log-on Vector
                                     0019-0028 IDs of 16 Requesters
                                     0029-002A Requesters Log-on Vector
                                     002B-003A IDs of 16 Requesters
                                     003B-003C Requester Log-on Vector
                                     003D-004C IDs of 16 Requesters

75: N_BUFSIZ         0   4B  0001    0001-0001 Requested Message Buf. Size
  (For system        1   4B  0002    0000-0000 Return Code
   use only.)                   0001-0002 Maximum Allowable Message
                                   Buffer Size

 77: N_KEEPALIVE      0   4D  0000    0000-0000 xx

                  Note: This message is unique in that there
                      is no return message.

 99: F_TRUNC          0   63  0024    0000-0000 User Number
                                0001-0024 FCB
   - EE -           1   63  0024    0000-0000 Directory/Return Code
                                0001-0024 FCB

100: DRV_SETLABEL     0   64  0034    0000-0000 Unused
                                0001-0024 Directory Label FCB
   - EE -                          0025-002C Old Default Password
                                002D-0034 New Default Password
                  1   64  0000    0000-0000 Return Code

101: DRV_GETLABEL     0   65  0000    0000-0000 Drive ID
   - EE -           1   65  0000    0000-0000 Directory Label Byte

102: F_TIMEDATE       0   66  0024    0000-0000 User Number
                                0001-0024 FCB
   - EE -           1   66  0024    0000-0000 Directory/Return Code
                                0001-0024 XFCB

103: F_WRITEXFCB      0   67  0034    0000-0000 User Number
                                0001-0024 XFCB
   - EE -                          0025-002C Old Password
                                002D-0034 New Password
                  1   67  0000    0000-0000 Directory/Return Code

105: T_GET            0   69  0000    0000-0000 xx
                  1   69  0004    0000-0001 Days since 31 DEC 1977
                                0002-0002 BCD Hours
                                0003-0003 BCD Minutes
                                0004-0004 BCD Seconds = 00

106: F_PASSWD         0   6A  0007    0000-0007 Default Password To Be Set
                  1   6A  0000    0000-0000 Return Code

134: Q_MAKE           0   86  001B    0000-001B Queue Descriptor
                  1   86  0003    0000-0003 Return Code (AX and CX)

135: Q_OPEN           0   87  000F    0000-000F Queue Parameter Block
                  1   87  002E    0000-0003 Return Code (AX and CX)
                                0004-0013 Queue Parameter Block
                                0014-002E Queue Descriptor

136: Q_DELETE         0   88  001B    0000-001B Queue Descriptor
                  1   88  0003    0000-0003 Return Code (AX and CX)

137: Q_READ           0   89  000F    0000-000F Queue Parameter Block
```

```
                 1   89  0003+   0000-0003 Return Code (AX and CX)
                                 0004-xxxx Queue Message

                 Note: The queue message length cannot exceed
                     the maximum message buffer size.

 138: Q_CREAD          0   8A  000F    0000-000F Queue Parameter Block
                 1   8A  0003+   0000-0003 Return Code (AX and CX)
                                 0004-xxxx Queue Message

                 Note: The queue message length cannot exceed
                     the maximum message buffer size.

 139: Q_WRITE          0   8B  000F+   0000-000F Queue Parameter Block
                                 0010-xxxx Queue Message
                 1   8B  0003    0000-0003 Return Code (AX and CX)

                 Note: The queue message length cannot exceed
                     the maximum message buffer size.

 140: Q_CWRITE         0   8C  000F+   0000-000F Queue Parameter Block
                                 0010-xxxx Queue Message
                 1   8C  0003    0000-0003 Return Code (AX and CX)

                 Note: The queue message length cannot exceed
                     the maximum message buffer size.

 158: L_ATTACH         0   9E  0000    0000-0000 Server List Device
                 1   9E  0003    0000-0003 Return Code (AX and CX)

 159: L_DETACH         0   9F  0000    0000-0000 Server List Device
                 1   9F  0003    0000-0003 Return Code (AX and CX)

 161: L_CATTACH        0   A1  0000    0000-0000 Server List Device
                 1   A1  0003    0000-0003 Return Code (AX and CX)
```

Appendix B: Building a server for another operating system

Since DR Net is an open system, it is possible to have requesters accessing servers that do not belong to the CP/M family of operating systems. There are two ways for a DR Net requester to access a foreign server. First, all differences between the systems can be reconciled in the requester nodes. In this case, the translation of the DR Net message into a form palatable to the server and then back again would take place in the NIOS-resident, LD_DRVR output and input routines, respectively. Second, the translation could be performed in the server. Under these conditions, requesters would communicate with the server via the standard DR Net message. This appendix addresses the considerations with respect to the second option.

The broad functions of the DR Net server module, and hence those that must be implemented by the foreign server, can be classified under the four headings described in Table B-1.

Table B-1. DR Net server module

Format: Classification	Meaning
Communication Control	The server's interface to the network must be built to manage message I/O and process control. Message I/O involves first message reception and decoding from the DR Net message format. Second, the response message must be encoded into the DR Net format, and then output. Some sort of process control is required, so that multiple requesters and/or processes can access the server simultaneously.
Function Interpretation	The server must be able to translate the Concurrent CP/M system calls into a call, or series of calls, that can be executed by its host operating system.
File System Conversion	Besides the system calls, the server must also be equipped to reconcile the BDOS File Control Blocks (FCBs) with the host operating system's file management mechanism.
Network Function Management	Finally, the server must be able to respond properly to the DR Net system calls. Specifically, N_LOGON, N_LOGOFF, and N_SCT must be accommodated.

B.1 The DR Net message components

Unless the requester accommodates it with special messages, the foreign server is completely reliant upon the DR Net message to deduce what it has been requested to do. For a description of the DR Net message, see Appendix A.

B.2 Functions to support

To provide complete Concurrent CP/M functionality, all of the functions listed in Appendix A must be supported. Table B-2 lists these functions by their associated resource.

Table B-2. Networked system functions by resource

Value in parentheses ["(" and ")"] is the Hex function number.

Disk drive	File	List device	Queue
DRV_SET (0E)	F_OPEN (0F)	L_WRITE (05)	Q_MAKE (86)
DRV_LOGINVEC (18)	F_CLOSE (10)	L_ATTACH (9E)	Q_OPEN (87)
DRV_ALLOCVEC (1B)	F_SFIRST (11)	L_DETACH (9F)	Q_DELETE (88)
DRV_SETRO (1C)	F_SNEXT (12)	L_CATTACH (A1)	Q_READ (89)
DRV_ROVEC (1D)	F_DELETE (13)		Q_CREAD (8A)

DRV_DPB (1F)	F_READ (14)	Q_WRITE (8B)
DRV_RESET (25)	F_WRITE (15)	Q_CWRITE (8C)
DRV_ACCESS (26)	F_MAKE (16)	
DRV_FREE (27)	F_RENAME (17)	
DRV_SPACE (2E)	F_ATTRIB (1E)	
DRV_FLUSH (30)	F_READRAND (21)	
DRV_SETLABEL (64)	F_WRITERAND (22)	
DRV_GETLABEL (65)	F_SIZE (23)	
	F_RANDREC (24)	
	F_WRITEZF (28)	
	F_LOCK (2A)	
	F_UNLOCK (2B)	
	F_MULTISEC (2C)	
	F_TRUNC (63)	
	F_TIMEDATE (66)	
	F_WRITEXFCB (67)	
	F_PASSWD (6A)	

Besides these functions, the Concurrent CP/M server is also programmed to respond to some console-related functions, and to remote requests for the time and date. Only three console functions are implemented, C_RAWIN (03), C_RAWOUT (04), and C_STAT (0B), and this is done only to maintain compatibility with CP/NET Version 1.2. Concurrent CP/M and CP/M-86 based requesters do not allow the mapping of consoles over the network. The only time and date function supported over by the server is T_GET (105). Finally, the DR Net functions 64 through 77 must be implemented.

Not all functions listed above need be supported by the foreign server. If one or more are removed, a process is restricted only in the use of the function or functions with respect to that server. For instance, if it is decided to drop queue support from the foreign server, a process can still make queue calls to a Concurrent CP/M requester and server.

B.3 Multisector transfers

Multisector transfers are allowed under DR Net, and are generally recommended to speed file data transactions. Before the data blocks can be transferred, however, the requester first must reconcile the length of the data block with any limitations imposed by the maximum message buffer size. The logica that derives the reconciliation is entirely resident in the requester.

The server must emulate the requester's multisector transfer behavior for the data to be received properly after a read file system call. Refer to Appendix A for the message contents of the following system calls.

Table B-3. Functions affected by multisector count

Dec	Hex	Mnemonic
20	14	F_READ
21	15	F_WRITE
33	21	F_READRAND

34	22	F_WRITERAND
40	28	F_WRITEZF
42	2A	F_LOCK
43	2B	F_UNLOCK

The server's key to the requester's behavior during multisector transfers is the message sent by the requester when it has trapped a function 44, F_MULTISEC. As shown in Appendix A, a Simulation Count is included in this message, along with the multisector count. This value indicates the maximum number of 128-byte records that can be contained in a single message. Until another function 44 is called, the multisector count and the Simulation Count remain in effect for the functions listed in Table B-3.

If you choose to implement multisector transfers on a foreign server, there are three things you should account for.

- First, the function 44 message contents contains the multisector count in the first DAT field byte, and the Simulation Count in the second byte. This differs from the local function 44, which only contains the multisector count.

- Second, there is likely to be one or more leftover 128-byte blocks. By comparing the actual count with the simulated count, the number of remaining blocks can be deduced.

- Third, the number of records successfully transferred must be indicated when there is a non-extended error on the server. Refer to the "Concurrent CP/M Programmer Reference's Guide" and the description of the BDOS Error Codes for an explanation of the difference between extended errors (which always return 0FFh in register AL), and non-extended errors (which return a value other than 0FFh in register AL).

EOF