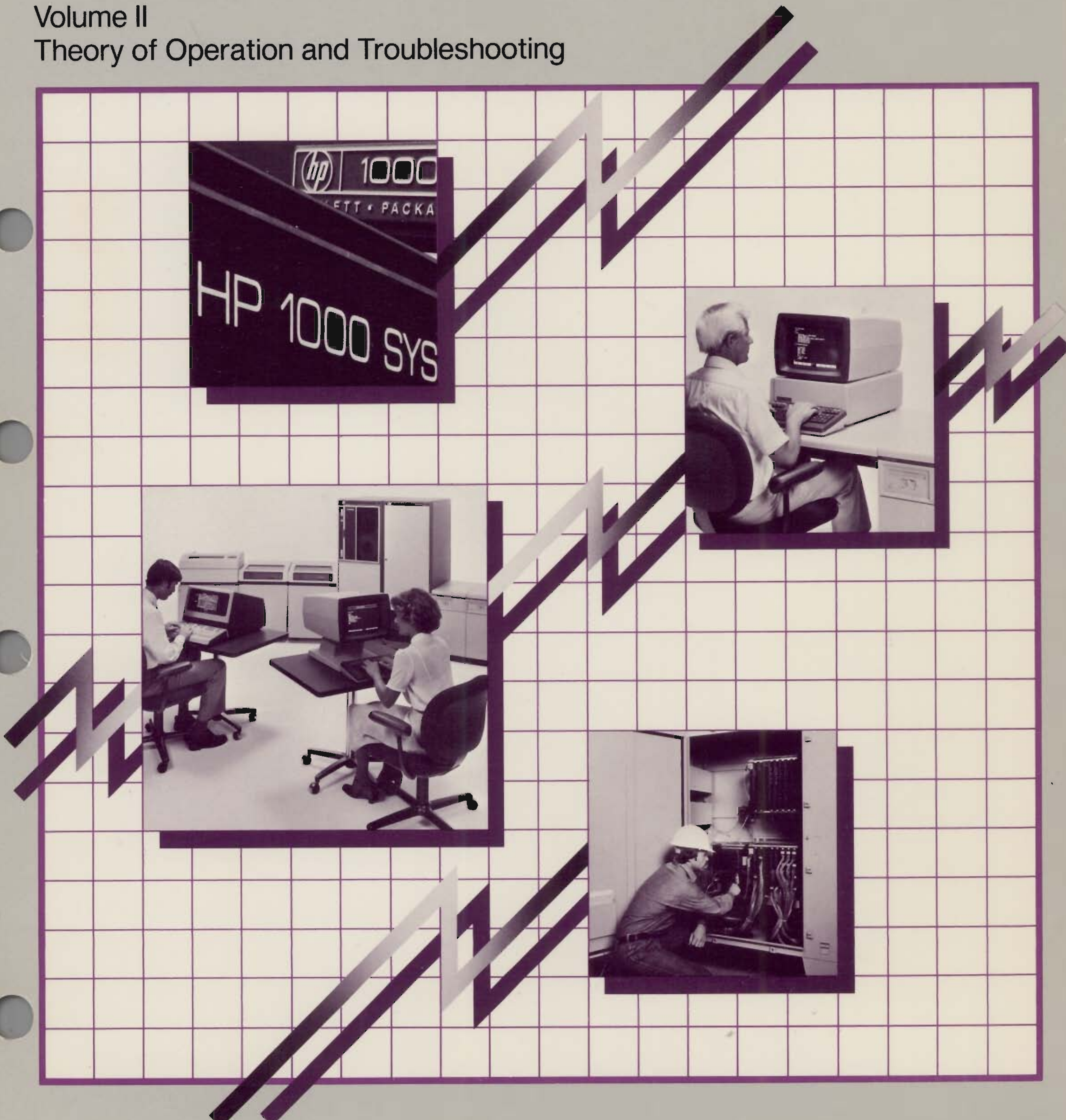
HP AdvanceNet

HEWLETT PACKARD

DS/1000-IV
Network Manager's Manual
Volume II
Theory of Operation and Troubleshooting



DS/1000-IV
Network Manager's Manual

Volume II
Theory of Operation
and Troubleshooting

HEWLETT
PACKARD
19420 Homestead Road, Cupertino, California 95014

Part No. 91750-90011

Printed in U.S.A. March 1985

PRINTING HISTORY

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates, however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

To determine what manual edition and update is compatible with your current software revision code, refer to the appropriate Software Numbering Catalog, Software Product Catalog, or Diagnostic Configurator Manual.

Second Edition .... Mar 1982
Update 1 .... Jan 1983
Update 2 .... Jun 1983
Update 3 .... Nov 1983
Update 4 .... Jun 1984
Third Edition .... Mar 1985

NOTICE

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Preface

Who needs to use this manual?

Volume II of the DS/1000-IV Network Manager's Manual is intended for the Network Manager or anyone assigned the responsibilities of a Network Manager, or anyone involved in troubleshooting a DS/1000-IV network.

What does it cover?

This volume provides the information required to understand the internal operation of a DS/1000-IV system. This includes the the distributed processing aspects of the application's software design. Also included in Volume II are troubleshooting tools and techniques.

Volume I of the DS/1000-IV Network Manager's Manual covers the generation and initialization of a DS/1000-IV network. This would include system planning as well as the software necessary for generating a DS/1000-IV network.

It is recommended that both volumes of the DS/1000-IV Network Manager's Manual be read once by all persons assigned any of the responsibilities of the Network Manager.

What does it assume?

This manual assumes that the Network Manager or the person assigned those responsibilities is familiar with DS/1000-IV and its capabilities. This person should have read and understood the DS/1000-IV User's Manual along with attending the DS/1000-IV Level I and Level II training courses (or equivalent experience). Volume I of the Network Manager's Manual should be read before reading Volume II. It is also assumed that this person be familiar with the appropriate RTE operating system (RTE-IVB, RTE-IVE, RTE-MIII, RTE-L, RTE-XL, RTE-A, RTE-6/VM), including File Manager, FMP routines, and RTE generation procedures.

If HP 3000 computers are to exist in the network, then familiarity with this system is also required.

How is it organized?

Volume II:

Chapter 1: Network Maintenance Utilities. This chapter discusses the utilities available for network monitoring.

Chapter 2: This chapter concentrates on troubleshooting. It provides troubleshooting tools and techniques which can be used to minimize system down-time in the event of a system failure.

Chapter 3: Discusses Remote I/O mapping. Chapter 3 describes the use of the Remote I/O Mapping software which provides programs with transparent access to I/O devices on other nodes in the network without using DS/1000-IV subroutine calls.

Chapter 4: This chapter discusses the remote control of the L-Series Processor Front Panel. Forced cold load of the E/F Series computers is also included.

Chapter 5: DS/1000-IV Internal Organization and Data Paths are the topics of this chapter. It discusses the program internals, as well as system organization and data paths.

Appendix A and B: Describes DS/1000-IV message formats, showing the sizes and layouts of each field in the message holder.

Appendix C: DS/1000-IV internal table formats are included in this chapter.

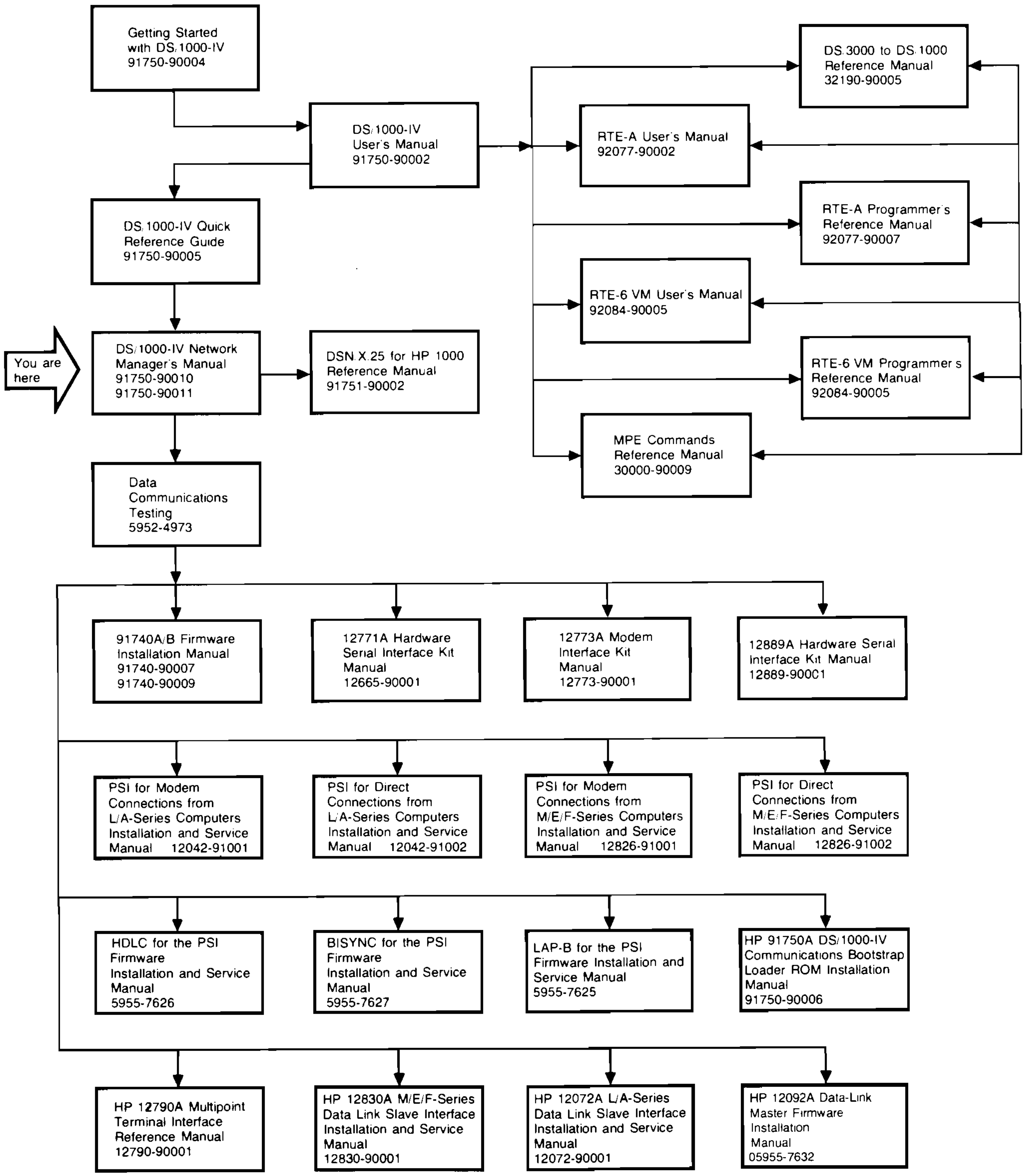
Appendix D: Listings of the communications bootstrap loaders for both the HP 12771A card and the HP 12794A/12825A (HDLC) cards.

Appendix E: Compatibility between DS/1000-IV (91750A) and DSIB' (91700A) is discussed in this appendix.

Appendix F: Compatibility between DS/1000-IV (91750A) and DS/1000 (91740A) is discussed in Appendix F.

Appendix G: This appendix notes the special miscellaneous considerations of which the Network Manager should be aware.

DOCUMENTATION MAP
DS/1000-IV



NOTE

DS/1000-IV cannot access non-FMGR files. Files used for CBL, DSVCP, Forced Cold Loads, REMAT, RFA, RMOTE and all other DS/1000-IV utilities must be FMGR files.

Table of Contents

Chapter 1 **Network Maintenance Utilities**

Chapter 2 **Troubleshooting**

Chapter 3 Remote I/O Mapping

Chapter 4 Remote VCP (DSVCP) and Forced Cold Loads

Chapter 5 Internal Organization and Data Paths

Appendix A **DS 1000-IV Message Formats**

Appendix B 1000 – 3000 Message Formats

List of Illustrations

List of Tables

Chapter 1
Network Maintenance Utilities

Volume I of the DS/1000-IV Network Manager's Manual provides the information necessary to generate and initialize a DS/1000-IV network. Volume II provides specific information concerning methods of interrogating your network when problems arise. Also included in Volume II are extensive descriptions of the internal operation and data paths of DS/1000-IV. The network maintenance utility programs are also discussed in this chapter.

The utility programs available with DS/1000-IV enable the Network Manager to determine the state of the DS environment and track communication line activity by recording communication line traffic in disc files for later analysis by additional utility programs.

This chapter presents these utilities which are valuable for use in detecting problems in the network from the communication line to the RTE resources used by DS.

DSINF

Three different versions of DSINF are provided in DS/1000-IV. %DSINL is for L-Series nodes, %DSIN2 is for nodes that only have the 3000 link, and %DSINF is for all other nodes.

These utility programs allow the user to examine the various RTE resources used by DS/1000. Since these programs provide the same functions only DSINF will be described. Any differences between the different versions will be noted where applicable.

DSINF allows you to examine:

* The Available Memory Suspend List
* I/O classes in use
* DS Values, including
    * Resource Numbers
    * Class Numbers assigned to DS programs
    * Time-out values
    * 3000 LU Table

* The Transaction Control Blocks in the DS SAM block

* DS Lists

    * Master Request List
    * Slave Streams
    * Process Number List
    * Transaction Status Table

* Network Routing Vector Specifications

* DS Equipment Table entries

* Synchronous Link Control (DVG67) long term statistics

* Message Accounting Values

* Rerouting Values

* Remote Sessions active at this node

NOTE

DSINF does not work on HP 1000 Data Link. Use DSPMP instead.

Operation of DSINF

Run from RTE with:

RU,DSINF[,INLU,OUTLU,CONWD,NODE,FLAG]

The run-time parameters have these meanings:

INLU — The logical unit number of the input device. The default is the number of the scheduling terminal passed by MTM or 1 in the session environment. If the input device is interactive, a prompt is printed on the device before each read.

OUTLU — The logical unit number of the device where information is printed. The default is the input lu (if interactive) or 6.

CONWD — A control word which, if non-zero, specifies DSINF will be run non-interactively. The functions which take place are determined by the bits set in the control word:

DECIMAL VALUE	PRINT THIS INFORMATION	OCTAL VALUE
1	AVAILABLE MEMORY SUSPEND LIST	1
2	I/O CLASSES	2
4	DS/1000 VALUES	4
8	DUMP OF SAM BLOCK	10
16	DS/1000 LISTS	20
32	NODAL ROUTING VECTOR	40
64	DS/1000 EQT ENTRIES	100
128	MESSAGE ACCOUNTING	200
256	REROUTING	400
512	REMOTE SESSIONS	1000

All these functions may be combined specifying the desired functions by the octal bit combination. Specifying -1 will cause data from all commands to be displayed or printed. For example, to print the I/O class and DS/1000 values on your terminal, type:

```
RU,DSINF,,,6
```

where 6 = 2(I/O classes) + 4 (DS/1000 values)

For those commands which have user specified options such as EQ,nn,yy; the most complete form of the command is printed.

NODE The node number where output is to occur. Default is the local node (-1).

Using the NODE parameter is mainly useful if you have a node that doesn't have a printer and some other node in the network does. Then you can run DSINF and have the output directed to the printer on another node. For example:

```
:RU,DSINF,,6,7,600
```

This string with the command mask of 7 will print the Available Memory Suspend List, I/O Classes, and DS/1000-IV values on LU 6 (printer) at node 600.

FLAG Set to a non-zero value when the node number is 0 (to distinguish it from the default value of 0).

DS information can be obtained from remote nodes with DSINF. Simply enter the following REMAT commands.

```
:RU,REMAT
$SW,20,,DS        [10 is the local node]
#RW,DSINF   (or #RW,DSINF,INLU,OUTLU
                 if at an MTM terminal)

     /DSINF:  FUNCTION?
```

There is no need to specify your own node number, DSINF will find it. DSINF, running at node 20, will reply back to your remote terminal and present the above prompt. At this point any of the commands described below may be entered and DS information from that node obtained. When DSINF is run locally, of course, you will get information about the local node.

?? – Invalid Commands or Help Function

DSINF's response to any invalid function entered by the user (including ??) is to print a list of valid functions. From this list the user can choose a valid function and try again.

```
/DSINF: FUNCTION ??

 /DSINF: VALID FUNCTIONS--
   AV        AVAILABLE MEMORY SUSPEND LIST
   CL        I/O CLASSES
   VA        DS/1000 VALUES
   DU        DUMP OF DS SAM BLOCK
   LI        DS/1000 LISTS
   NR OR /N  NODAL ROUTING VECTOR    - not in %DSIN2
   EQ        DS/1000 EQT ENTRIES    \
   EQ,N      DS/1000 EQT ENTRY # N   >  different in L-Series
   LU,N      EQT ENTRY FOR LU # N   /
   MA        MESSAGE ACCOUNTING      - not in %DSIN2
   RR        REROUTING               - not in %DSIN2
   RS        REMOTE SESSIONS         - not in %DSINL
   EX OR /E  TERMINATE DSINF
```

AV – Available Memory Suspend List

```
AVAILABLE MEMORY SUSPEND LIST  (Different format in L-Series)

    PT  SZ  PRGRM  T PRIOR  AMT.MEM     FATHER
-----------------------------------------------------------------------
     5   5  SON    3    99    3608      F     GF
     5   5  DAVEM  3    99B   3708      DAVEE GYLRD LENDR WILAM JASON
                                        FMGR
-----------------------------------------------------------------------
```

In this example, two programs are in the available memory suspend list. 'SON' is in partition 5, requires 5 pages, is a type 3 program, has priority 99, has requested 3608 words of system available memory, and was scheduled by 'F'. 'F' was scheduled by 'GF'.

'DAVEM' has the same information with the following exceptions: The 'B' following the priority indicates it is being run in batch mode, the amount of memory is 3708 words, and his 'ancester' programs are as follows: FMGR scheduled JASON, who scheduled WILAM, who scheduled LENDR, who scheduled GYLRD, who scheduled DAVEE, who scheduled DAVEM.

If no programs are waiting for memory, DSINF prints:

```
AVAILABLE MEMORY SUSPEND LIST IS EMPTY
```

CL – Print I/O Classes

DSINF prints the following information in response to the CL command:

```
 I/O CLASS INFORMATION
    64 CLASSES IN SYSTEM

    CLASSES IN USE:
      CLASS  STATE   GET      POSSIBLE OWNER
        44    AL              PTOPM
        45    AL              PTOPM
        46    GT    SPOUT     SMP     PURG
        47    AL              EXECM
        48    AL              EXECM
        49    GT    OPERM     DINIT
        50    GT    RFAM      DINIT
        51    GT    EXECM     DINIT
        52    GT    PTOPM     DINIT
        53    GT    EXECW     DINIT
        54    GT    DLIST     DINIT
        55    GT    OTCNV     DINIT
        56    GT    INCNV     DINIT
        57    GT    RSM       DINIT
        58    GT    QCLM      DINIT
        59    GT    RTRY      DINIT
        60    GT    GRPM      DINIT
        61    GT    R$PN$     SMP     PURG
        62    GT    LOGON     SMP     PURG
        63    GT    LGOFF     SMP     PURG
        64    AL              SMP     PURG

    43 CLASSES AVAILABLE
```

Each of the classes in use can be in one of the following three states:

AL -- The class has been allocated but no program is waiting to receive data from it.

GT -- The class has been allocated and the program named in the 'GET' column is waiting on it for data via a class I/O 'GET' call.

BU -- The class has been allocated and data is currently waiting. The number of buffers and their total size are printed on one line and the allocation information is printed on a second line. An example of this state can be found in later examples in this chapter.

The possible owner field lists the programs which may have allocated the class. For slave monitors, this would be DINIT (unless DINIT was removed from the system after initialization). Other programs may show up as the possible owner. This is because of the way RTE-IVB and MIII maintain the I/O Class Table, it is impossible to determine exactly which program originally allocated a class, so there may be more than one possible owner listed. In RTE-L the name of the program currently occupying the ID segment of the program that allocated the class originally, is printed.

If the program which actually allocated the class has been removed from the system without releasing the class, or removed and later restored to a different ID segment address, it will not appear as a possible owner. If a program is aborted while waiting with a GET, the GET program name will be <NONE>.

VA – DS Values

The information DSINF prints for the VA command depends upon which monitors and which DS/1000-IV options have been enabled. Here is an example of the output:

```
DS/1000 VALUES:

 RESOURCE NUMBERS:      OWNER     LOCKER
   MA TABLE ACCESS 23 <GLOBAL>    <NONE>       (Shows Resource Numbers
   QUEX CLEANUP    24 <GLOBAL>    <NONE>       in use depending on what
   QUEZ "LISTEN"   25 <GLOBAL>   <GLOBAL>      portion of DS is
   QUIESCENT       26 <GLOBAL>    <NONE>       initialized. MA, 1K-3K
   TCB ACCESS      27 <GLOBAL>    <NONE>       etc.)

 CLASSES ASSIGNED TO PROGRAMS:
   40  M. A.
   39  RSM
   37  OTCNV              (Shows I/O Classes in use by DS programs
   38  INCNV               exclusive of DS Monitor Programs.)
   41  QCLM
   42  RTRY
   43  GRPM
   44  RPCNV
   45  RQCNV
   46  QUEZ
   47  QUEX
```

```
  TIMEOUT VALUES (SEC):
    MASTER T/O          45     (Seconds)
    SLAVE T/O           30     (Seconds)
    REMOTE BUSY RETRIES  3     (Number of Attempts)
    REMOTE QUIET WAIT    0     (Interval in seconds)
    MAX RETRY DELAY   2.00     (Seconds)

  MAXIMUM HOP COUNT     20     (How many nodes can a message travel
                               through before it is considered lost
                               and flushed.)
  MAX LINK DOWN COUNT   10     (Link failures in a five minute
  PROGL MESSAGE LU   0         period.)
  UPGRADE LEVEL  1
  UPGRADE SUBLEVEL 1
      26 RFA FILES MAY BE OPEN

 HP 3000 LU TABLE                   (Printed if a 3000 link exists)
   LU  BUFFER  SIZE  CR FLAG        (CR FLAG will be 1 if HP 3000
  115        304          0         accepts new continuation record for
                                    $STDIN/$STDLIST, 0 if it does not.)
  243     4096            1 (X.25)  (X.25 printed if LU is X.25 LU)
```

NOTE:

The buffer size reported for X.25 HP 3000 LUs is that which was generated into the system. There is no communication buffer size for X.25.

If a portion of DS/1000-IV (HP 3000 links, HP 1000 links, Message Accounting) is not enabled, the information associated with that portion is not printed.

For the resource numbers, the name of the owner program should be <GLOBAL> because all DS/1000-IV RNs are allocated globally. Under normal conditions, the table access and quiescent RNs are not locked (<NONE> is the locker) and the QUEZ RN is locked globally.

DU – Dump of System Available Memory Block

DSINF dumps the locations used for Transaction Control Blocks, HP 3000 Transaction Status Table and the Process Number List. These table formats are found in Appendix A of this manual.

[See material in the DS/1000-IV Lists section for TCB information.]

The Master TCBs are used to keep track of master requests while they are in the network. The Slave TCBs keep track of requests while they are being processed by the Slave Monitors.

DUMP OF TCB BLOCK

LOC	OCTAL CONTENTS OF LOC THROUGH LOC+5					
3515	3531	367	1603	2035	20541	0
3523	3537	373	1607	1606	2	16006
3531	3671	372	1605	1604	2	16006
3537	3553	370	1606	2035	20541	0
3545	3641	367	1575	2035	20541	0
3553	0	367	1610	2035	20541	0
3561	3655	372	1566	1565	2	16006
3567	3617	370	1570	2035	20541	0
3575	3611	367	1567	2035	20541	0
3603	3545	367	1573	2035	20541	0
3611	3567	373	1571	1570	2	16006
3617	3633	367	1572	2035	20541	0
3625	3663	367	1576	2035	20541	0
3633	3603	372	1574	1573	2	16006
3641	3625	372	1577	1576	2	16006
3647	3515	367	1601	2035	20541	0
3655	3575	367	1565	2035	20541	0
3663	3677	367	1600	2035	20541	0
3671	3523	367	1604	2035	20541	0
3677	3647	372	1602	1601	2	16006

DUMP OF HP3000 TRANSACTION STATUS TABLE

LOC	OCTAL CONTENTS OF LOC THROUGH LOC+7						
3705	0	0	0	0	0	0	0
3714	0	0	0	0	0	0	0
3723	0	0	0	0	0	0	0
3732	0	0	0	0	0	0	0
3741	0	0	0	0	0	0	0
3750	0	0	0	0	0	0	0

Every time an HP 3000 user makes a master request to the HP 1000, an entry is made in the Transaction Status Table (TST). If there is no HP 3000 connected to the node, the TST table is not printed. Each TST entry consists of 14 words (two lines of the dump):

WORD 1 - DS/1000 Stream (0 if the table entry is not in use)
WORD 2 - Local Sequence Number
WORD 3 - Holding class number
WORD 4 - Monitor Class Number
WORD 5 - Call Type Code
WORD 6 - Mask Word (POPEN)
WORD 7 - LEFT BYTE: Length of DS/3000 request (words)
RIGHT BYTE: DS/3000 Message Class
WORD 8 - Reserved for future use
WORD 9 - DS/3000 Stream
WORD 10 - Reserved for future use

```
WORD 11 - LEFT BYTE: From Process Number
          RIGHT BYTE: To Process Number
WORD 12 - Local Sequence Number
WORD 13 - Reserved for future use
WORD 14 - LU number request arrived on
```

LI – DS Lists

DSINF prints the number of Transaction Control Blocks (TCBs) and Process Number List entries in each of the lists. If a list has entries, DSINF prints the address of the first TCB in the list. The first word in the entry is the address of the next entry or else zero to indicate the end of the list. If an LI command immediately follows the DU command the lists reflect the state of the TCB's in the previous dump. The information displayed by the DU or LI commands always reflects a snapshot of the data when the data is in a steady state (no lists are being changed).

PNL entries, Master TCBs and Slave TCBs are allocated out of the Transcation Control Block Pool:

1. Process Number List entries:

   An entry is made in this list each time an HP 1000 user issues a successful HELLO to an HP 3000 or logs onto a remote HP 1000. The entry is deleted after a BYE (HP 3000) or a DLGOF (HP 1000) or when the Master Program terminates. The words in the PNL have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Bit 14: when set, indicates an HP 3000 session
WORD 3 - Remote node number (1000) or -LU (3000)
WORD 4 - Logging device LU number
WORD 5 - Bit 15:    When set, indicates an entry UPLIN will
                    release at its next execution.
         Bits 0-14: Program's ID Segment address
WORD 6 - Remote Session ID or HP 3000 Process Number
```

2. Master List entries:

The Master List contains an entry for each master request currently outstanding. The words in the TCB have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Bit 15:      Temporary bit used by UPLIN
         Bit 14:      Set for requests to HP 3000
         Bit 13:      MA acknowledgement
         Bits 8-12:   Reserved for future use
         Bits 0- 7:   Timeout counter. Timeout occurs when the
                      counter reaches octal 377.

WORD 3 - Local Sequence Number
WORD 4 - Bit 15:      When set, indicates long master timeout
                      (about twenty minutes)
         Bits 0-14:   Master Class Number

WORD 5 - Bit 15:      When set, indicates an entry UPLIN will
                      release during its next execution.
         Bits 0-14:   Master program's ID Segment address
WORD 6 - MA Sequence Number (1000) or -LU (3000)
```

3. Slave List entries:

The Slave Lists contain an entry for each outstanding slave request. There is a Slave List for each enabled slave monitor. Entries in these lists represent each remote request waiting for service at the time the lists are printed. The words in the TCB have these meanings:

```
WORD 1 - Address of next entry in list (0 indicates end)
WORD 2 - Same as master list entry
WORD 3 - Local Sequence Number   (Assigned when request arrives)
WORD 4 - Origin Sequence Number  (Assigned at the origin node)
WORD 5 - Origin Nodal Address (1000) or -LU (3000)
WORD 6 - Reserved
```

Any entries which are not in one of these three active lists are not currently being used and are in the Null List.

The following example shows the output of the LI command:

```
/DSI16: FUNCTION?LI

 DS/1000 LISTS

       1 ENTRIES IN MASTER REQUEST LIST, STARTING AT  5060
        PROG  CLASS  T/O CTR
        DSI.A   27       0

  ACTIVE SLAVE MONITORS:                    1ST TCB
      STREAM  CLASS  MONITOR  ENTRIES  LOCATION
         1      38  DLIST          1      5110
         3      37  EXECW          2      5030
         4      36  PTOPM          0
         5      35  EXECM          0
         6      34  RFAM           0
         7      33  OPERM          0
         8      32  VCPMN          0
       3 ENTRIES IN SLAVE LISTS

      16 ENTRIES IN NULL LIST, STARTING AT  5044
       2 ENTRIES IN PROCESS NUMBER LIST, STARTING AT  4772
        PROG  LOGLU
        REM16   16
       *RM020   20
      B TST21   21
```

The asterisks in the Process List and Master Request List indicate the programs that are using a 3000 link. The "B" in the Process List and Master Request List indicates that the associated entry will be deleted by UPLIN. Slave monitor stream lists are not printed if the associated slave monitor has not been enabled by DINIT.

NR or /N – Print Nodal Routing Vector

DSINF prints the LU, EQT, subchannel, select code, and timeout for each node in the NRV:

```
/DSINF: FUNCTION? NR
 NRV SPECIFICATIONS:
 LOCAL NODE#:      2, NO. OF NODES=      9

   NODE     LU    EQT  SUB   T/O(SEC)    TYPE  LEVEL
      1*     9      3   1       10        65     0
      3*    15     15   0       10        66     1
     20      9      3   1       30        65     0
     21      9      3   1        0        65     0
     10      9      3   1        0        65     0
     11      9      3   1        0        65     1
      4      9      3   1        0        65     1
      6*    17     17   0        0        66     1
      2*     0                   0               1

    (* INDICATES NEIGHBOR)
```

The NRV printout has the following field definitions:

NODE - Node address in the network.

LU - Logical Unit used to transmit messages from the local node to the node designated in the NODE column. If zero, indicates either this is the local node or there is currently no path to this node.

EQT - Equipment Table entry associated with the LU.

SUB - Subchannel setting - indicates to driver DVA65 whether to communicate open loop (no echo, if even) or closed loop (echo, if odd). Subchannel setting is not used on other links.

T/O - NRV master time-out override for this node.

TYPE - Driver type, 65 = DVA65 and 66 = DVA66.

LEVEL- Current upgrade level of DS software at the indicated node:
0 = DS/1000 (Product Number 91740),
1 = DS/1000-IV (Product Number 91750).

When setting up the network or troubleshooting problems, the T/O values and level values should be the same between both nodes you're trying to communicate between.

EQ – Equipment Table Information

DS/1000-IV Equipment Table Information (EQ Command)

DSINF prints information associated with each DS/1000-IV Equipment Table entry. The information and format is somewhat different for DVA65, DVA66, and DVG67, and for DVA66 the meaning of values returned by the driver differ for the HDLC and BISYNC (BSC) firmware.

The EQ command has the following format:

```
EQ [,eqt number [,printing option]]
```

If the EQT number is not provided, DSINF prints information on each EQT with driver type 65, 66, or 67. If an EQT number is provided, only that EQT entry is printed (if it is a DS/1000-IV driver).

NOTE: The Data Link slave driver, type 64, is not recognized.

If no option parameter is provided, DSINF prints the LU, EQT, and driver type, followed by flag bits (if type is 66) or Synchronous Link Control (SLC) statistics (if type is 67). Other options are:

IO TO PRINT EQT WORDS AND DESCRIPTIONS (OCTAL VALUES)

PA (DVA66 ONLY) TO PRINT INTERFACE PARAMETERS (DECIMAL)

ST (DVA66 ONLY) TO PRINT INTERFACE PARAMETERS AND STATISTICS (DECIMAL)

AL TO PRINT ALL OF THE ABOVE

When the parameters or statistics are read from a DVA66 card, the driver is called. If the card is busy or down, DSINF must wait. Reading the statistics sets them back to zero.

For all drivers, words 4 and 5 have the standard RTE meanings:

WORD 4-- meaning if set

```
BIT 15:     1 If DMA required
BIT 14:     1 If automatic buffering is used
BIT 13:     1 If driver is to process power fail
BIT 12:     1 If driver is to process time-out
BIT 11:     1 If device timed out
BITS 6-10:  Last subchannel addressed
BITS 0-5:   Select code for I/O controller
```

WORD 5-- meaning if set

```
  BITS 14-15: 0 If I/O controller is available, 1 if disabled (DOWN),
              2 If busy, 3 if waiting for DMA
  BITS  8-13: Driver type (65 for DVA65, 66 for DVA66, 67 for DVG67)
  BITS  0- 7: Status
```

NOTE

For more information on the EQT, see DS DRIVER EQTs in Appendix C.

DSINF prints the following for EQT's with driver type 65:

```
 EQT #    7,  LU #  7, TYPE 65
   WORD  VALUE  MEANING                WORD  VALUE  MEANING
     1       0  I/O LIST ADDRESS         2   30702  INITIATION ADDRESS
     3   31443  CONTINUATION ADDR        4   10125  FLAGS/SUBCHNL/SC
     5   32401  AV/TYPE/STATUS           6     107  CONWD
     7       0  DATA BUFFER ADDRESS      8      14  DATA BUFFER LENGTH
     9       0  REQUEST BUFFER ADDR     10       0  REQUEST BUFFER LEN
    11   31467  COROUTINE ADDRESS       12    2000  CURRENT STATUS
    13   34145  EQT EXTENSION ADDR      14  177775  NOMINAL TIMEOUT
    15       0  TIMEOUT CLOCK           16       1  MICROCODE COUNT
    17       0  LAST WORD RECEIVED      18       0  VERTICAL PARITY WORD
    19       0  DIAGONAL PARITY WORD    20       0  TOTAL BLOCK TRANSFER
    21       0  TOTAL WRITE RETRIES     22       7  LU NUMBER
```

Note that EQT words 16 through 22 are in the EQT extension.

DVA65 maintains status bits in words 0 and 7:

```
WORD 5        BITS 7-0 INDICATE STATUS:

   BIT  0  =  0
   BIT  1  =  Any error (BITS 4-7 indicate type)
   BIT  2  =  Write Request
   BIT  3  =  Non-DS Request
   BITS 4
   THRU 7  =  Error Type(octal)
```

Recoverable Errors

```
 0  =  No error
 1  =  Line failure, a parity or protocol error was
       detected and could not be resolved.
 2  =  Timeout, remote did not respond to a protocol
       character within a line timeout.
 3  =  Local busy, driver is currently processing a
       message going in the opposite direction, or both
       sides of the link attempted to send messages at
       the same time.
 4  =  Message aborted, a "STOP" was received.
 5  =  Remote busy, remote side was unable to schedule
       QUEUE or to allocate SAM.
```

Fatal Errors

```
10  =  Not initialized, an "Initialize Link" command has
       not been received yet. If in response to an
       initialize command, indicates that system tables
       were not configured correctly.
11  =  Wrong mode, wrong type of traffic, (i.e., trying
       to do a download when driver is in DS mode.
12  =  Illegal request, command is not supported by DVA65
       (DVA65 will give this in response to a FPL
       send/receive, break, set mode request with the
       wrong security code).
17  =  Unknown interrupt received
```

```
BITS 8
THRU 13  =  Indicate Driver Type, in this case = 65
```

WORD 12-- meaning if set

```
BIT 15:   Powerfail recovery in progress
BIT 14:   Microcode read/write flag
BIT 13:   Flag for write retry in progress
BIT 12:   Last successful operation (1 = write)
BIT 10:   Listen mode enabled
BIT  9:   Request pending
BIT  6:   Broken line
BITS 0-5: Retry counter or broken line counter
```

DSINF prints the following for EQTs with driver type 67 (DVG67):

```
  EQT #   6, LU # 115, TYPE  67
   WORD  VALUE  MEANING                WORD  VALUE  MEANING
     1       0  I/O LIST ADDRESS         2   10125  INITIATION ADDRESS
     3   10071  CONTINUATION ADDR        4  110043  FLAGS/SUBCHNL/SC
     5   33400  AV/TYPE/STATUS           6       1  CONWD
     7   60511  DATA BUFFER ADDRESS      8       0  DATA BUFFER LENGTH
     9   24500  REQUEST BUFFER ADDR     10       0  REQUEST BUFFER LEN

  SLC LONG TERM STATISTICS
              36  READ REQUESTS                     40  WRITE REQUESTS
              79  MESSAGES TRANSMITTED               0  SPURIOUS INTERRUPTS
               0  LINE ERRORS                        0  NAKS RECEIVED
               1  BCC/PARITY ERRORS                  0  LONG TIMEOUTS
               0  RESPONSE ERRORS                    1  RESPONSE REJ
               0  WACK/TTD RECEIVED
```

All the EQT values are octal. The long term statistics are decimal.

DVG67 maintains status in bits 0-7 of word 5:

OCTAL CODE	ABORT?	MEANING
0	NO	The request completed normally
1	YES	Invalid request
2	YES	Request incompatible with line state
4	YES	Local hardware failure
5	NO	End-of-transmission (EOT) received
6	NO	Disconnect (DLE EOT) received
7	YES	Long timeout occurred
10	YES	ENQ received in response to EOT
11	YES	Data overrun
12	YES	Maximum number of NAKs received
13	YES	Maximum number of ENQs sent
14	YES	Reverse interrupt (RVI) received
15	NO	ENQ received in response to ENQ sent
16	YES	NAK received in response to write inquiry
17	YES	Maximum number of ENQs received in write Conversational situation
20	YES	Incorrect response (not NAK) to TTD
21	YES	Impossible situation
22	YES	Text error

BITS 5-7 are the block specification bits:

BIT 5 = 0 for no heading, 1 for heading

BIT 6 = 0 for nontransparent mode, 1 for transparent mode

BIT 7 = 0 for ETX ending block, 1 for ETB ending block

Here is a sample printout of a DVA66 entry for the BISYNC card:

```
EQT    61, LU   122, TYPE  66
  WORD  VALUE  MEANING                WORD  VALUE  MEANING
    1       0  I/O LIST ADDRESS         2   10144  INITIATION ADDRESS
    3   10155  CONTINUATION ADDR        4   30035  FLAGS/SUBCHNL/SC
    5   33004  AV/TYPE/STATUS           6  151101  CONWD
    7    7201  DATA BUFFER ADDRESS      8      15  DATA BUFFER LENGTH
    9    7216  REQUEST BUFFER ADDR     10       0  REQUEST BUFFER LEN
   11       0  SERVICING PROCESS       12    4346  ASSOCIATED EQT
   13    5763  EQT EXTENSION ADDR      14       0  NOMINAL TIMEOUT
   15       0  TIMEOUT CLOCK           16  177766  RETRY CNTR/READ PNTR
   17      10  1ST READ LEN/SKIP CT    18       0  2ND READ LEN/SKIP CT
   19       0  FRAME LENGTH ON CARD    20    7216  WRITE POINTER
   21       0  WRITE BUFFERS LENGTH    22    4140  MAX PSI FRAME SIZE
   23       0  NUM OUTPUT BUFFERS      24   11100  READ CONT ADDRESS
   25   11674  WRITE CONT ADDRESS      26   13000  FLAG BITS
   27  110172  MISCELLANEOUS BITS

FLAG BITS (EQT WORD 26)
 0 READ ABORTED     0 WRITE ABORTED     0 RD RQ PENDING    0 WT RQ PENDING
 0 BKPL LOCKED RP   0 BKPL LOCKED WP    0 SHORT TO ACTIVE  0 MED. T.O ACTIVE
 0 LONG T.O ACTIVE  1 CONNECTED         1 START OF MSG.    0 NON-DS MODE
 1 ASKED TO CONNCT  0 SEVERE ERROR      0 P-F RECONNECT    0 RFP WAIT

  BSC  BOARD, FIRMWARE REV.2328,  SPEED: 57.6KBPS, INTERNAL CLOCK
  FCL DISABLED, DIAGNOSTIC HOOD NOT SENSED

  PARAMETERS/STATISTICS
        20856  GOOD BLOCKS SENT             17054  GOOD BLOCKS RCVD
            0  BAD BLOCKS RECEIVED              0  NAKS RECEIVED
            0  WACKS SENT                    2030  WACKS RECEIVED
            0  TTDS SENT                        0  TTDS RECEIVED
            0  RESPONSE ERRORS                102  3 SECOND TIMEOUTS
            0  LINE ERRORS                   2144  BLOCK SIZE (BYTES)
            8  RETRY LIMIT                    255  CONNECT TIMER
         2490  TRACE SIZE (BYTES)               2  MODE
            1  AUTOMATIC RE-ENABLE              0  CONNECT AS SECONDARY
```

NOTE:

DSINF reports information on the diagnostic hood only, not the loop back verifier hood. Ignore the diagnostic hood message if you are using the loop back hood.

All counters on the BISYNC card are maintained as 16-bit unsigned integers. If they are not reset, they will roll over at 65,535. Statistics are reset each time the connection is established unless the connection is automatically reenabled.

The following is an explanation of each of the parameters or statistics reported:

GOOD BLOCKS SENT - Number of data blocks that the BISYNC card has sent which were acknowledged as good by the remote HP 3000. A data block is the basic tranmission unit at the BISYNC level. One user message may be sent as several BISYNC data blocks. Each data block sent must be acknowledged by the receiver.

BAD BLOCKS RECEIVED - Number of bad blocks received from the remote HP 3000. (See "GOOD BLOCKS SENT" for an explanation of data blocks.)

WACKS SENT - Number of Wait Positive Acknowledgments sent. A WACK is sent when a good data block has been received but the receiver is not yet ready to receive another. Upon receipt of a WACK, the sender will normally send an ENQ until the receiver indicates it is ready to receive by responding with an ACK 0. A WACK may be sent if the receiver does not have an available buffer.

TTDS SENT - Number of Temporary Text Delays sent to the remote. A TTD is transmitted by the sending station when it wants to retain control of the line but is not yet ready to send. This may occur if the send buffers are not yet full.

RESPONSE ERRORS - This counteris incremented each time the local station receives an unexpected response (i.e., a protocol error). This may be the result of a transmission error.

LINE ERRORS - Number of times an error has been detected with the line (eg., loss of data set ready). Thismay be an indication of faulty lines or connections.

RETRY LIMIT - Number of times a transmission will be retried if errors are detected. Default is 8.

TRACE SIZE (BYTES) - Size in bytes of the buffer area on the BISYNC card which is reserved for state change logging.

AUTOMATIC RE-ENABLE - The automatic enable flag indicates to the firmware that the card should be re-enabled in secondary mode after a disconnect has occurred. This flag is always set by DS/1000-IV softare as 2326.

GOOD BLOCKS RCVD - The number of good data blocks received from the remote HP 3000. (See "GOOD BLOCKS SENT" for an explanation of data blocks.)

NAKS RECEIVED - Number of Negative Acknowledgements received. A NAK may be sent by the receiving station if it receives a bad data frame. A high number of NAKs received or sent may indicate poor line quality or bad connections. Upon receipt of a NAK, the sending station should retransmit the previous data block.

WACKS RECEIVED - Number of Wait Positive Acknowledgements sent. (See "WACKS SENT" for an explanation of WACKs.)

TTDS RECEIVED - Number of Temporary Text Delays received from the remote station. (See "TTDS SENT" for an explanation of TTDs.)

3 SECOND TIMEOUTS - The amount of time the receiver will wait for any response before requesting a retry.

BLOCK SIZE (BYTES) - The maximum number of bytes that the card can send across the line as a single block.

CONNECT TIMER - Number of seconds that the receiver will wait for an initial connection.

MODE - Indicates the type of BISYNC being used. Zero indicates normal generalized BISYNC; 1 is limited conversational BISYNC; and 2 is full conversational BISYNC. For HP 1000 to HP 3000 communication, the mode will always be 2.

CONNECT AS SECONDARY - Indicates wither the card is connected as Primary (0) or as Secondary (1). This is not an indication of which side initiated the BISYNC connection, nor does it indicate that either side is a "master" or "slave."

Different statistics and parameters are printed for the HDLC board (the board parameters indicate which type of firmware is loaded). These line statistics are reset when read by the driver. The form of the speed and clock messages depends on the setting of the switches on the HDLC/BSC interface board:

```
SPEED: 300 BPS, INTERNAL CLOCK
         ^        ^
         !        !
         !        +----- OR EXTERNAL
         !
         +---- This field may contain any of the following:
                  300 BPS
                  1200 BPS
                  2400 BPS
                  4800 BPS
                  9600 BPS
                  19.2KBPS
                  57.6KBPS
                  MAXIMUM  (230K BPS)
```

When using external clock, the actual speed is determined by modem, but setting should match, or be slower than, that of the modem used.

The L-Series version of DSINF does not have the EQ command, instead use the LU command.

The following is an example of the HDLC card printout:

```
/DSI16: FUNCTION?LU,67,AL


 EQT #   4, LU #  67, TYPE  66
   WORD  VALUE  MEANING                WORD  VALUE  MEANING
     1       0  I/O LIST ADDRESS         2     6142  INITIATION ADDRESS
     3    6145  CONTINUATION ADDR        4    30013  FLAGS/SUBCHNL/SC
     5   33004  AV/TYPE/STATUS           6   152101  CONWD
     7   77572  DATA BUFFER ADDRESS      8        0  DATA BUFFER LENGTH
     9   77572  REQUEST BUFFER ADDR     10       21  REQUEST BUFFER LEN
    11       0  SERVICING PROCESS       12     2277  ASSOCIATED EQT
    13    3244  EQT EXTENSION ADDR      14        0  NOMINAL TIMEOUT
    15       0  TIMEOUT CLOCK           16   177766  RETRY CNTR/READ PNTR
    17       0  1ST READ LEN/SKIP CT    18        0  2ND READ LEN/SKIP CT
    19       0  FRAME LENGTH ON CARD    20    77611  WRITE POINTER
    21       0  WRITE BUFFERS LENGTH    22     1000  MAX PSI FRAME SIZE
    23       0  NUM OUTPUT BUFFERS      24     7033  READ CONT ADDRESS
    25    7556  WRITE CONT ADDRESS      26    53000  FLAG BITS
    27     103  MISCELLANEOUS BITS

 FLAG BITS (EQT WORD 26)
   0 READ ABORTED     0 WRITE ABORTED    0 RD RQ PENDING     0 WT RQ PENDING
   0 BKPL LOCKED RP   0 BKPL LOCKED WP   0 SHORT TO ACTIVE   0 MED. T.O ACTIVE
   0 LONG T.O ACTIVE  1 CONNECTED        1 START OF MSG.     0 NON-DS MODE
   1 ASKED TO CONNCT  0 SEVERE ERROR     1 P-F RECONNECT     0 RFP WAIT

   HDLC BOARD, FIRMWARE REV.2040,  SPEED:  MAXIMUM, 1024 BYTE FRAME SIZE
   FCL DISABLED, DIAGNOSTIC HOOD NOT SENSED

   PARAMETERS/STATISTICS
           333  GOOD I-FRAMES RCVD             229  RR FRAMES RECEIVED
             0  RNR FRAMES RECEIVED              0  REJECT FRAMES RCVD
             0  RCV PROC OVERRUNS                0  CRC ERRORS
             0  ABORT SEQ. RECEIVED              0  RECIVER OVERRUNS
             0  RX BUFFER OVERFLOWS              0  FRAMES W/BAD ADDR
             0  CMDR FRAMES RCVD                 7  UNACK FR WINDOW SIZE
            10  N2 RETRY COUNT                  15  T1 T.O. IN 0.01 SEC
```

DSINF reads the following information from the HDLC board and resets the values after it reads them.

For more information on HDLC protocol, refer to the HDLC Firmware installation manual.

GOOD I-FRAMES RCVD - Number of Information frames received by the HDLC card. Information frames carry data.

RR FRAMES RECEIVED - The remote card sends Receiver Ready (RR) frames to indicate that it is ready to receive an information frame or to acknowledge information frames. The remote card can also send an RR frame with the poll bit set to one to solicit a response.

RNR FRAMES RECEIVED - The remote card sends Receiver Not Ready (RNR) frames when it is busy or unable to receive frames (for example, if the board runs out of buffers).

REJECT FRAMES RECEIVED - The remote card sends Reject (REJ) frames to request retransmission of information frames starting from the frame number in the N(R) (Receive Sequence Number) field.

RCV PROC OVERRUNS - Number of times the card ran out of buffer space because the host system could not read information off the card. Check that QUEUE is running, and the amount of SAM in your system.

CRC ERRORS - The Cyclic Redundancy Check (CRC) checks for transmission errors for each frame. A high CRC error rate may indicate a noisy link.

ABORT SEQ. RECEIVED - The remote card sends an abort sequence (seven or more consecutive ones) to prematurely terminate the transmission of a frame.

RECEIVER OVERRUNS - Number of times the card could not read data from the line into a receive buffer.

RX BUFFER OVERFLOWS - Number of times the card received a frame larger than the expected frame side. Check what size frames the remote card is sending.

FRAMES W/BAD ADDR - If the first byte in the address field is not one or three, the card ignores the frame and increments this counter.

CMDR FRAMES RCVD - The remote card sends Command Reject (CMDR) frames if it receives a frame without a CRC error, but it is unable to process, for one of the following reasons: the frame was invalid; the information frame exceeds the available buffer size; or the frame has an invalid N(R) field.

UNACK FR WINDOW SIZE - The unacknowledged frame window size is the maximum number of messages that can be outstanding before the card requires an acknowledgement. This number is set to seven.

N2 RETRY COUNT - The number of times the card will try to retransmit after a T1 timeout. Refer to Chapter 5 of this manual for more information.

T1 T.O. in .01 SEC - The maximum period of time that the sending card will wait for an acknowledgement before retransmitting a frame. This value is determined by the line speed and the selected frame size. Refer to Chapter 5 of this manual for more information.

LU – Print Information from Associated EQT

This command performs the same function as the EQ command and is useful in the RTE-L version of DSINF which doesn't have EQT's. In RTE-L EQT's don't exist but the same information is available from Device Table entries. When running the L-version the EQ command is not provided. The LU command is used instead. The same IO, ST, and AL option parameters apply. The LU command is equally useful in RTE-IVB and MIII versions when the LU is known and the EQT is not.

MA – Message Accounting Values

Message Accounting values are printed with the MA command. When issued the following output is provided:

```
/DSINF: FUNCTION? MA

 MESSAGE ACCOUNTING INFORMATION

   NODE STATE  # UNACK  # LINEDOWNS  TIMEOUT
    1    NONE     0          0          00
    2    UP       0          1          20
    4    PEND     3          3          20
```

In the above example, the local node is node 3. The information provided during initialization indicated nodes 1, 2, and 4 as well as 3, have the Message Accounting option present.

The NODE column lists the addresses of the network nodes indicated to have the Message Accounting option installed during initialization.

The STATE column shows the state of communications between the local node and the node identified in the NODE column. In this column three different entries are possible:

NONE which means the node doesn't have MA initialized.

UP which means the state of communications with that node are up (normal operational state).

PEND which means the Message Accounting software is attempting to establish communications with the destination node.

DOWN which means communications between these two MA nodes has failed (no path to the node exists, or no path has been used since DS initialization).

The # UNACK column shows the number of unacknowledged messages that have been sent from the local node to other MA nodes from which no acknowledgement has yet been received. This field can contain entries from 0 to 15. When 15 unacknowledged messages are pending, any attempt to send additional messages will result in a DS08 (node busy or resource unavailable) returned to the caller.

The # LINEDOWNS indicates how many times since DS was initialized all paths to the destination node have gone down. This value simply provides insight into the link quality, i.e., if the linedown count is small, the link quality must be reasonably good. Note that MA will automatically reinitialize a link which has been down but comes back to service.

The TIMEOUT column shows how long, in seconds, MA will wait for an acknowledgement for a particular message.

When Message Accounting is not present in the system the following message is printed:

```
 MESSAGE ACCOUNTING INFORMATION

         NO ENTRIES
```

RR – Rerouting Values

The RR command produces the following output:

```
 REROUTING SPECIFICATIONS:
                UP/DOWN
    LU  COST   COUNTER  STATUS
    15    1        0      UP
    17    1        0      UP
     9    1        0     DOWN
```

The information is provided by Logical Unit number. The Cost column shows the current Cost Factor associated with messages going out on the particular link. This is the value assigned by the Network Manager for use when the node initializes. These values are used by the Rerouting software to determine the "least-cost" route to a particular node. The sum of the costs for each link in a path is maintained in the Cost Matrix for each path. The summed "costs" are provided in rerouting update messages that are sent during network initialization.

The Up/Down counter value indicating the current count of link failures as the link approaches an up or down state. Finally the status of the link is printed.

When Rerouting is not generated into the system, the response is:

```
    REROUTING SPECIFICATIONS:

       NO ENTRIES
```

RS – Remotely Owned Sessions

```
/DSINF: FUNCTION?RS

 REMOTE SESSIONS ESTABLISHED AT THIS NODE

   SOURCE   SESSION ID
    NODE   SOURCE LOCAL  TIMER   PROGRAM
     600      92   252   0:00:35  DSI.A (CLONE)
     600      83   251   0:08:05
     600      63   250   0:06:10

     4 EMPTY ENTRIES
```

The RS command prints the list of remotely created local sessions active at this node.

SOURCE NODE - Node which originated the session. (This will be the local node number if the session was established from an HP 3000.)

SOURCE SESSION ID - ID of session at the source node that created the local session. This will be the terminal LU at the source node.

LOCAL SESSION ID - ID of the remotely created session at the local node. These session identifiers are in the range 253, 252, 251, etc., and will not be legal LU numbers.

TIMER - Hours, minutes and seconds since session was last accessed.

PROGRAM - Name of program currently active under the local session at the local node.

Note: When the ACCTS program is used to shut down some or all of the remotely created sessions, the DS table being displayed by DSINF will not reflect these terminated sessions until the next DLGON/DLGOF request. At this time the DS tables are verified with the actual session environment.

DSINF – Example

Run REMAT at node 600, switch to node 500, then schedule DSINF.

```
:RU,REMAT
$SW,500,,DS
#RW,DSINF
```

Dump the TCB area:

```
/DSI.A: FUNCTION?DU

DUMP OF TCB BLOCK
 LOC        OCTAL CONTENTS OF LOC THROUGH LOC+5
 4756       5124      367   4317   13026   32041       0
 4764       5044      372   4324    4333    1130   16005
 4772          0      374   4325    4334    1130   16005
 5000       4756      372   4315    4324    1130   16007
 5006       5052      367   4312   13026   32041       0
 5014       5036      367   4304   13026   32041       0
 5022       4764      372   4323    4332    1130   16005
 5030       5006      367   4311   13026   32041       0
 5036       5066      367   4305   13026   32041       0
 5044          0      373   4321   13026   32041       0
 5052       5102      374   4313   13026   32041       0
 5060       5000   100025   4310    4317    1130   16003
 5066       5074      377   4306   13026   32041       0
 5074       5116      377   4307   13026   32041       0
 5102       5060      367   4314   13026   32041       0
 5110          0        0   1130      20   31472     374
 5116       5030   100044   4230    4237    1130   16003
 5124       5022      367   4320   13026   32041       0
 5132          0   100021   4322  107425   31472       0
 5140          0   100023   4316    4325    1130   16003
```

Print the lists in the TCB area:

```
/DSI.A: FUNCTION?LI

 DS/1000 LISTS

    1 ENTRIES IN MASTER REQUEST LIST, STARTING AT  5132
     PROG  CLASS  T/O CTR
     REM16   21    1190


 ACTIVE SLAVE MONITORS:                 1ST TCB
   STREAM  CLASS  MONITOR  ENTRIES  LOCATION
      1      38   DLIST          0
      3      37   EXECW          1       5140
      4      36   PTOPM          0
      5      35   EXECM          1       4772
      6      34   RFAM           0
      7      33   OPERM          0
      8      32   VCPMN          0
    2 ENTRIES IN SLAVE LISTS

   16 ENTRIES IN NULL LIST, STARTING AT  5014
    1 ENTRIES IN PROCESS NUMBER LIST, STARTING AT  5110
     PROG  LOGLU
     REM16   16
```

Using the list information and the TCB dump we can determine the following:

There is currently one master request outstanding and its TCB is at location 5132. The master program is REMAT from terminal LU 16.

The timeout counter is 1190 meaning this master request has 1190 seconds before it times out.

The master class number is 21.

There is one slave request outstanding for EXECW. (Whenever DSINF is run with wait remotely, there will be a slave request for EXECW.) There is a slave request for EXECM.

There is an entry in the Process Number List for the Remote Session activated by REMAT at node 600.

Which class numbers are in use?

```
   /DSI.A: FUNCTION?CL

 I/O CLASS INFORMATION
    48 CLASSES IN SYSTEM

    CLASSES IN USE:
      CLASS  STATE   GET       POSSIBLE OWNER
        21     GT    REM16     SMP      DSMOD    REM16
        23     GT    LUMAP     LUQUE    QUEX     EDI20
        24     GT    <NONE>    JOB      TMP2A
        25     AL              $YCOM    VCPMN    TMP2
        26     BU[  1 BLOCK(S)  154 WORDS]
               AL              $YCOM    VCPMN    TMP2
        27     BU[  1 BLOCK(S)   28 WORDS]
               AL              $YCOM    VCPMN    TMP2
        28     BU[  1 BLOCK(S)    9 WORDS]
               AL              $YCOM    VCPMN    TMP2
        29     BU[  1 BLOCK(S)   13 WORDS]
               AL              $YCOM    VCPMN    TMP2
        30     AL              RTRY     EXECM    EDI84
        31     BU[  2 BLOCK(S)   52 WORDS]
               AL              RTRY     EXECM    EDI84
        32     GT    VCPMN     QUEZ     DINIT
        33     GT    OPERM     QUEZ     DINIT
        34     GT    RFAM      QUEZ     DINIT
        35     GT    EXECM     QUEZ     DINIT
        36     GT    PTOPM     QUEZ     DINIT
        37     AL              QUEZ     DINIT
        38     GT    DLIST     QUEZ     DINIT
        39     GT    OTCNV     QUEZ     DINIT
        40     GT    INCNV     QUEZ     DINIT
        41     GT    RSM       QUEZ     DINIT
        42     GT    QCLM      QUEZ     DINIT
        43     GT    RTRY      QUEZ     DINIT
        44     GT    GRPM      QUEZ     DINIT
        45     GT    R$PN$     R$PN$    ACCTS
        46     GT    LOGON     R$PN$    ACCTS
        47     GT    LGOFF     R$PN$    ACCTS
        48     AL              R$PN$    ACCTS

    21 CLASSES AVAILABLE
```

There are several buffers waiting of different classes unrelated to DS. Class 21 is in a GT (GET) state, REMAT is waiting in a class get for a completion of it's schedule with wait request to run DSINF. Class 37 is allocated (AL) with no one waiting, this would be EXECW's class. EXECW is not in a get state while a program (DSINF) has been scheduled with wait.

How much activity has gone over the link?

```
/DSINF: FUNCTION?EQ,3,AL

   EQT #   3,  LU # 11, TYPE 65
   WORD  VALUE  MEANING                 WORD  VALUE  MEANING
     1       0  I/O LIST ADDRESS          2   27421  INITIATION ADDRESS
     3   30162  CONTINUATION ADDR         4   10111  FLAGS/SUBCHNL/SC*
     5   32401  AV/TYPE/STATUS*           6       0  CONWD
     7   51354  DATA BUFFER ADDRESS       8      37  DATA BUFFER LENGTH
     9   51354  REQUEST BUFFER ADDR      10       0  REQUEST BUFFER LEN
    11   30206  COROUTINE ADDRESS        12    2000  CURRENT STATUS*
    13   31156  EQT EXTENSION ADDR       14  177774  NOMINAL TIMEOUT
    15       0  TIMEOUT CLOCK            16       1  MICROCODE COUNT
    17  170360  LAST WORD RECEIVED       18   12237  VPW/REPLY REQ LENGTH
    19   63377  DPW/REPLY DATA LEN       20  104633  TOTAL BLOCK TRANSFER
    21       0  TOTAL WRITE RETRIES      22   31607  NEW REQ ID SEQ ADD
```

Word 20 tells us 67,995 (decimal) requests have gone over the line.

SLCIN General Information

The Synchronous Line Control INformation program, SLCIN, prints long-term statistics and the Event Trace Table maintained by the Synchronous Line Control (SLC) package for hardwired links from DS/1000 to HP 3000 systems. This facility is only available for the HSI (12889A) card. Use DSINF LU or EQ commands for PSI cards. In DS/1000, the implementation of SLC is handled by the driver DVG67. The user of SLCIN will need considerable understanding of the BISYNC protocol to interpret the data presented by SLCIN.

The long term statistics and Event Trace Table are maintained in the SSGA module D$EQT. (When SLCIN is loaded on line it must be given access to SSGA.) In the currently released version, there are 100 words set aside for the trace table.

Each entry in the trace table contains two words plus one word for each event and state that occurs during the given request. Here is the format:

```
            -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
Bits       |15|14|13|12|11|10| 9| 8| 7| 6| 5| 4| 3| 2| 1| 0|
            -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
Word 1     |           Address of next entry               |
            ----------------------- ----- -----------------
Word 2     |           v           |  w  |        x        |
            ----------------------- ----- -----------------
Word 3     |           y           |           z           |
            ----------------------- -----------------------
           |           y           |           z           |
            ----------------------- -----------------------
            (y and z word repeated as many times as needed)
```

v = Completion status of request (see below)

w = I/O request code (1=read, 2=write, 3=control)

x = Function code used in request (see below)

y = SLC event number (see below)

z = SLC state number that is the result of the event (see below)

The trace buffer is circular; when the last word is filled, the next entry goes to the first word of the table.

The "Station States" are the states SLC is in after each request when it is waiting for another request.

The SLC initialize control request resets the trace table (all previous entries are destroyed). As requests are received by SLC, it uses the event and current state to address its State-Transition table. This tells it what action to perform and what state it will be in next. Thus, the trace table can show how the HP 1000 got out of step with the HP 3000 if something goes wrong.

Scheduling SLCIN

Scheduling

Run from RTE with:

```
RU,SLCIN,<outlu>
```

Where <outlu> is the LU where the information is to be printed. The default is 1 or the multi-terminal-monitor LU.

If the node has not been initialized by DINIT, SLCIN will not print the SLC information. Instead it will tell the user to "RUN DINIT FIRST!". If the 3000 link is not HSI (12889) the message "HP 3000 LINK NOT HSI".

If the node has been initialized, but the HP 3000 was not enabled, SLCIN prints "HP 3000 NOT ENABLED" and terminates.

If the 3000 link has been enabled, SLC starts printing information. The SLC long term statistics are printed first, followed by the Event Trace Table. The trace table includes the octal status, the function, and the event(s) and state(s) associated with the function.

Meaning of Status

Octal Code	Abort?	Meaning
0	no	The request completed normally
1	yes	Invalid request
2	yes	Request incompatible with line state
4	yes	Local hardware failure
5	no	End-of-transmission (EOT) received
6	no	Disconnect (DLE EOT) received
7	yes	Long timeout occurred
10	yes	ENQ received in response to EOT
11	yes	Data overrun
12	yes	Maximum number of NAKs received
13	yes	Maximum number of ENQs sent
14	yes	Reverse interrupt (RVI) received
15	no	ENQ received in response to ENQ sent
16	yes	NAK received in response to write inquiry
17	yes	Maximum number of ENQs received in write
20	yes	Incorrect response (not NAK) to TTD
21	yes	Impossible situation
22	yes	Text error

Synchronous Link Control (SLC) Functions

SLC Functions

Read Functions:

Read Inquiry

Used to wait for the 3000 to bid for the line with an ENQ.

Read Initial

Used to receive text from the 3000 after an ENQ is received.

Write Functions:

Write Inquiry

Used to bid for the line. Send an ENQ, receive an ACK0.

Write Conversational

The normal way of sending and receiving text with the 3000; send a block, receive a block of text. Continues until EOT is received from 3000 or 1000 performs write reset.

Write Reset

Send an EOT to relinquish use of the line.

Write Disconnect

Send DLE EOT to inform 3000 that there is to be no more transmission of data.

Control Functions:

Clear

Disables the interface board and clears the internal trace table. (Inverse of initialize.)

Initialize

Establishes the link between DVG67 and D$EQT where the long term statistics and Event Trace Table are kept.

Line Open

Readies the line for read/write operations.

Line Close

Disconnects the line and disables it for further read/write operations. (Inverse of line open.)

Zero the Long Term Communications Statistics

Resets all the long term statistics to zero.

SLC Events

SLC Events

These are the events reported by SLCIN.

SLCIN Abbreviation	Full Meaning (if necessary)
LINE OPEN REQ	Line Open Request
LINE CLOSE REQ	Line Close Request
READ INQUIRY REQ	Read Inquiry Request
READ INITIAL REQ	Read Initial Request
WRITE INQURY REQ	Write Inquiry Request
WRITE CONV REQ	Write Conversational Request
WRT RESET(EOT)RQ	Write Reset Request (Write EOT)
WRITE DISCON REQ	Write Disconnect Request (Write DLE EOT)
ACK0 RECEIVED	
ACK1 RECEIVED	
WACK RECEIVED	
ENQ RECEIVED	
NAK RECEIVED	
EOT RECEIVED	
DLE EOT RECEIVED	
TTD RECEIVED	
TEXT RECEIVED	
BCC PRTY/FMT ERR	BCC Parity or Format Error Detected
TEXT OVERRUN	
GARBAGE RECEIVED	
SHORT TIMEOUT	
LONG TIMEOUT	
LOW	HIGH, LOW, and MID are used to expand
HIGH	the range of the SLC states described
MID	below.

SLC States

SLCIN Abbreviation	Full Meaning (if necessary)
UNOPENED	Line has not been initialized yet
CONTROL	
READ ENQ	
READ ENQ ERROR	
CHECK READ REQ	Check Read Request Type
READ	
READ TEXT	
READ RVI	
RESTRICTED READ	

WRITE ENQ	
WRITE ENQ ERROR	
ENQ-ENQ CONTENTN	ENQ-ENQ Contention
WRITE	
WRITE TEXT	
WRITE RESPNS ENQ	Write Previous Response ENQ
CHECK RESPONSE	
BAD ACK RECEIVED	
WRITE RETRY	
ENQ RCV IN WRITE	ENQ Record in Write or Write Conversational Mode
ENQ RCRD IN WRIT	Second State: ENQ Received in Write or Write Conversational Mode
WRITE CONVERSTNL	Write Conversational
WRITE EOT	
READ EOT RSPONSE	Read EOT Response
DISCONNECT	Hang Up (A Disconnect)
WRITE TTD	

LOW, HIGH, and MID Events

When LOW, HIGH, or MID appears in a dump, the meaning is associated with the state on the line above (the state SLC was in when the event started) not the one on the same line (the state after the event completed).

The LOW, HIGH, and MID events have different meanings for different states. They are defined as follows:

Read ENQ Error
- MID -- not defined
- HIGH-- no error counter overflow
- MID -- error counter overflow: bad id

Check Read Request Type
- LOW -- read delay made in read state or read inquiry
- HIGH-- read initial
- MID -- read delay made in restricted read state

Read RVI
- MID -- 2nd RVI request
- HIGH-- 1st RVI request
- MID -- not defined

Write ENQ Error
- LOW -- error counter overflow, no bad id
- HIGH-- no error counter overflow
- MID -- error counter overflow, bad id

```
ENQ-ENQ Contention
  LOW -- retry counter overflow
  HIGH-- no retry counter overflow
  MID -- not defined

Previous Response ENQ
  LOW -- retry counter overflow
  HIGH-- no overflow--write mode
  MID -- no overflow--write conversational mode

Check Response
  LOW -- bad ACK received
  HIGH-- RVI received
  MID -- good ACK received

Bad ACK Received
  LOW -- timeout flag not set or bad-response flag set
  HIGH-- timeout flag set and bad-response flag not set (write
          conversational mode)
  MID -- timeout flag set and bad-response flag not set (write
          conversational mode)

Write Retry
  LOW -- retry counter overflow
  HIGH-- no overflow--write mode
  MID -- no overflow--write conversational mode

ENQ Received (Write or Write Conversational Mode)
  LOW -- text not just received
  HIGH-- text just received and ENQ not just sent
  MID -- ENQ just sent

Second State--ENQ Received in Write
  LOW -- retry counter overflow
  HIGH-- no overflow--write mode
  MID -- no overflow--write conversational mode

Write EOT
  LOW -- line error
  HIGH-- no line error
  MID -- not defined

All Other States
  LOW -- line error
  HIGH-- not defined
  MID -- not defined
```

Examples

```
*RU,SLCIN
SLC LONG TERM STATISTICS:
        0 READ REQUESTS              4 WRITE REQUESTS
        4 MESSAGES TRANSMITTED       0 SPURIOUS INTERRUPTS
        0 LINE ERRORS                0 NAKS RECEIVED
        0 BCC/PARITY ERRORS          0 LONG TIMEOUTS
        0 RESPONSE ERRORS            0 RESPONSE REJ
        0 WACK/TTD RECEIVED

SLC EVENT TRACE TABLE:
           FUNCTION          EVENT             STATE

           INITIALIZE
           COMPLETION STATUS 00: NORMAL COMPLETION

           LINE OPEN         LINE OPEN REQ     CONTROL
           COMPLETION STATUS 00: NORMAL COMPLETION

           ZERO COMM STATS
           COMPLETION STATUS 00: NORMAL COMPLETION

           WRITE INQUIRY     WRITE INQURY REQ  WRITE ENQ
                             ACK0 RECEIVED     WRITE
           COMPLETION STATUS 00: NORMAL COMPLETION

           WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                             TEXT RECEIVED     READ
           COMPLETION STATUS 00: NORMAL COMPLETION

           WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                             TEXT RECEIVED     READ
           COMPLETION STATUS 00: NORMAL COMPLETION

           WRITE RESET       WRT RESET(EOT)RQ  WRITE EOT
                             HIGH              CONTROL
           COMPLETION STATUS 00: NORMAL COMPLETION
```

The first example shows the trace table immediately after the link to the 3000 has been established. Here are the messages that were sent:

```
       HP 1000                               HP 3000

          ENQ     ------------->
                        <--------------    ACK0
         TEXT     ------------->
                        <--------------    TEXT
     NULL MESSAGE ------------->
                        <-------------- NULL MESSAGE
          EOT     ------------->
```

Here is another run of SLCIN, after communication between the 1000 and 3000 has taken place:

```
*RU,SLCIN
SLC LONG TERM STATISTICS:
       22 READ REQUESTS                 42 WRITE REQUESTS
       65 MESSAGES TRANSMITTED           0 SPURIOUS INTERRUPTS
        0 LINE ERRORS                    0 NAKS RECEIVED
        0 BCC/PARITY ERRORS              0 LONG TIMEOUTS
        0 RESPONSE ERRORS                0 RESPONSE REJ
        0 WACK/TTD RECEIVED

SLC EVENT TRACE TABLE:
             FUNCTION           EVENT              STATE

             WRITE CONVERSTNL  WRITE CONV REQ     WRITE CONVERSTNL
                               EOT RECEIVED       CONTROL
             COMPLETION STATUS 05: EOT RECEIVED

             READ INITIAL      READ INITIAL REQ   READ ENQ
                               ENQ RECEIVED       CHECK READ REQ
                               HIGH               READ TEXT
                               TEXT RECEIVED      READ
             COMPLETION STATUS 00: NORMAL COMPLETION

             WRITE CONVERSTNL  WRITE CONV REQ     WRITE CONVERSTNL
                               TEXT RECEIVED      READ
             COMPLETION STATUS 00: NORMAL COMPLETION

             WRITE CONVERSTNL  WRITE CONV REQ     WRITE CONVERSTNL
                               TEXT RECEIVED      READ
             COMPLETION STATUS 00: NORMAL COMPLETION

             WRITE CONVERSTNL  WRITE CONV REQ     WRITE CONVERSTNL
                               TEXT RECEIVED      READ
             COMPLETION STATUS 00: NORMAL COMPLETION
```

```
WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE RESET        WRT RESET(EOT)RQ  WRITE EOT
                   HIGH              CONTROL
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE INQUIRY      WRITE INQURY REQ  WRITE ENQ
                   ACK0 RECEIVED     WRITE
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   EOT RECEIVED      CONTROL
COMPLETION STATUS 05: EOT RECEIVED

READ INITIAL       READ INITIAL REQ  READ ENQ
                   ENQ RECEIVED      CHECK READ REQ
                   HIGH              READ TEXT
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE RESET        WRT RESET(EOT)RQ  WRITE EOT
                   HIGH              CONTROL
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE INQUIRY      WRITE INQURY REQ  WRITE ENQ
                   ACK0 RECEIVED     WRITE
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL   WRITE CONV REQ    WRITE CONVERSTNL
                   EOT RECEIVED      CONTROL
COMPLETION STATUS 05: EOT RECEIVED
```

```
READ INITIAL      READ INITIAL REQ  READ ENQ
                  ENQ RECEIVED      CHECK READ REQ
                  HIGH              READ TEXT
                  TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                  TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE RESET       WRT RESET(EOT)RQ  WRITE EOT
                  HIGH              CONTROL
COMPLETION STATUS 00: NORMAL COMPLETION

READ INITIAL      READ INITIAL REQ  READ ENQ
                  ENQ RECEIVED      CHECK READ REQ
                  HIGH              READ TEXT
                  TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE RESET       WRT RESET(EOT)RQ  WRITE EOT
                  HIGH              CONTROL
COMPLETION STATUS 00: NORMAL COMPLETION
```

There are a number of read and write sequences in the above example. Here is a breakdown of what goes across the line:

```
STANDARD READ OPERATION:
        READ INITIAL      READ INITIAL REQ  READ ENQ
                          ENQ RECEIVED      CHECK READ REQ
                          HIGH              READ TEXT
                          TEXT RECEIVED     READ
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                          TEXT RECEIVED     READ
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                          TEXT RECEIVED     READ
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                          TEXT RECEIVED     READ
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE RESET       WRT RESET(EOT)RQ  WRITE EOT
                          HIGH              CONTROL
        COMPLETION STATUS 00: NORMAL COMPLETION
```

```
        HP 1000                                HP 3000

                         <--------------        ENQ
         ACK0      ------------->
                         <--------------       TEXT
    TEXT OR NULL ------------->
                         <-------------- TEXT OR NULL
    TEXT OR NULL ------------->
                         <-------------- TEXT OR NULL
    NULL MESSAGE ------------->
                         <-------------- NULL MESSAGE
          EOT      ------------->
```

```
STANDARD WRITE OPERATION:
        WRITE INQUIRY     WRITE INQURY REQ  WRITE ENQ
                          ACK0 RECEIVED     WRITE
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                          TEXT RECEIVED     READ
        COMPLETION STATUS 00: NORMAL COMPLETION

        WRITE CONVERSTNL  WRITE CONV REQ    WRITE CONVERSTNL
                          EOT RECEIVED      CONTROL
        COMPLETION STATUS 05: EOT RECEIVED
```

```
        HP 1000                                HP 3000

          ENQ      ------------->
                         <--------------       ACK0
         TEXT      ------------->
                         <-------------- NULL MESSAGE
     NULL MESSAGE ------------>
                         <--------------        EOT
```

The final example shows the driver level activity which goes on when there are no higher level messages being sent. If a system is left alone after initialization, the trace table will fill up with these messages.

```
READ INITIAL       READ INITIAL REQ  READ ENQ
                   ENQ RECEIVED      CHECK READ REQ
                   HIGH              READ TEXT
                   TEXT RECEIVED     READ
COMPLETION STATUS 00: NORMAL COMPLETION

WRITE RESET        WRT RESET(EOT)RQ  WRITE EOT
                   HIGH              CONTROL
COMPLETION STATUS 00: NORMAL COMPLETION
```

The 3000 sends an ENQ approximately every 30 seconds to determine if the line is still up.

TRC3K/LOG3K – 1000/3000 Message Logging Programs

The DS/1000-DS/3000 logging facility lets users record all DS/3000 messages and/or DVG67 driver trace entries. The software which does the logging is contained in several different modules:

RES - Within the data storage area of RES there is a five word array starting with the entry point #CL3K. These words indicate the logging LU, whether the LU is spooled, and what information (header, appendage, data, driver) to record.

LOG3K - A program which the user runs interactively to report and modify the logging information in RES. Additionally, LOG3K can set up a spool LU pointing to a user-supplied file. (This feature is not supported on RTE-L/XL/A.)

QUEX (HSI and PSI) - When messages are received from the 3000, the trace records are written by either QUEX or D3KMS depending on the message's DS/3000 class.

QUEX (HSI) or D3KMS (PSI) - When messages are written to the 3000, one of these modules looks at the information in RES (#CL3K) and writes trace records depending on what has been set via LOG3K.

TRC3K - A program, patterned after TLOG, which prints a report on the information recorded.

This software allows users to record logging information on an LU (or, through spooling, in a file in RTE-IVB and RTE-6/VM systems.) This record is later analyzed using TRC3K.

LOG3K

LOG3K

The user checks and modifies the information in RES with LOG3K.

To run LOG3K enter:

```
RU,LOG3K,<Console LU>
```

Where:

Console LU - Is the logical unit LOG3K uses for input and output. If not supplied, LOG3K gets it from a call to LOGLU.

If the 3000 link has not been enabled by DINIT, an error message is printed and LOG3K terminates.

Here are examples of how LOG3K reports current logging options:

```
DS/1000-3000 LOGGING STATUS
 LOG LU <NONE>
```

In this example, no logging has been set up.

```
DS/1000-3000 LOGGING STATUS
 LOG LU 13
 LOGGING NOTHING
```

In this example, the log LU has been set up but no logging type has been set.

```
DS/1000-3000 LOGGING STATUS
 LOG LU 57 (FILE TRCFL1)
 LOGGING 50 WORDS DATA AND DRIVER
```

LOG3K prints the status, asks "CHANGES? ", and waits for the user to indicate what type of changes he wants.

Possible changes:

LU [,lu]	New log logical unit.
FI [,namr]	New log file (SPOOL system must be active.) The file must already exist and must be type 3 or greater. A spool LU will be assigned to this file by LOG3K. (Supported on non-RTE-L/XL/A systems only.)
UP	Reset log LU to be "up"
TY [,logging type]	New logging type

In addition to these changes, the user can type /E, EX, NO, or EN to terminate LOG3K. Typing ?? causes LOG3K to print an explanation of its commands.

When an LU is not provided in the LU command, LOG3K asks for a "NEW LOG LU:". If the operator response is /E, LOG3K goes back to the "CHANGES?" prompt. Otherwise the old entry is cleared (if the old log LU is spooled, the spool is released) and the new information is written into RES.

To turn off all logging, set the log LU to 0.

For FI, LOG3K checks whether SMP is in the system (via PGMAD). If it is, LOG3K asks for a "NEW LOG FILE:". If the response is /E, LOG3K goes back to the "CHANGES?" prompt. Otherwise the old entry is cleared, a SPOPN call is made to obtain an LU number that points to the indicated file, and the new information is written into RES. There are three error messages that may be reported by the FI command:

    SORRY, NO SPOOL

When SMP is not present in the system an error is returned in the following format:

    ***SMP ERROR -XXXX

Where -XXXX is the negative error code returned by SPOPN.

    "FI" COMMAND NOT ALLOWED ON THIS SYSTEM

When the FI command is used on an RTE-L/XL/A system.

The UP command causes LOG3K to clear the up/down flag maintained by QUEX. QUEX sets the flag when an error has occurred writing to the log LU. After the flag is set, no further logging occurs until the UP command is issued by the operator.

The TY command is used to set the logging option flags for QUEX. In response to the "NEW LOG TYPE:" prompt the user can specify:

NO	Nothing
HE	Communication buffer header only
AP	Header and appendage
DA:n	Header, appendage, and "n" words data
DR	Driver events and states (may be first or second option) The DR command is not available on modem (DVA66) links.

One or two options may be set, but NO, HE, AP, and DA:n can appear only in the first position (they are mutually exclusive).

Example: DA:50,DR Traces 50 words data and driver.

After a change is made by any command, LOG3K reports the new logging options and goes to the "CHANGES?" prompt.

If LOG3K is run under an RTE session, the logging LU must be mapped into the user's SST. LOG3K posts the true System LU number in RES for use by QUEX/QUEZ (which is outside of session). When spooling is used, the same session should set up and release the logging file. Otherwise LOG3K will warn the user that it cannot release the spool file, and the user must run GASP to release it (via the KS command). If the session which set up logging terminates before releasing it, the session monitor will release the spool file and QUEX/QUEZ will get an error the next time it tries to write.

QUEX (HSI LINKS)

QUEX checks for logging after it examines a request or reply buffer in the subroutine VERIF. If the proper bits are set in RES, QUEX writes the header/appendage/data on the log LU.

Before issuing a line clear request to DVG67, QUEX logs the long term statistics if the driver trace bit is set.

QUEX records information from the DVG67 trace table (if the driver trace bit is set) following I/O operations.

QUEX performs the EXEC call that logs information using the "no-abort" bit. When the error return is taken, QUEX prints a message on the system console in the form:

```
/QUEX: TRACING ERROR aabb
```

Where aabb is the contents of the A- and B- registers after the error return. QUEX sets the sign bit in error flag in RES upon detection of the error and no further logging is attempted.

QUEX writes the log records in three formats:

DVG67 TRACE ENTRIES RECORD

```
Word 0--Trace type = -1
Words 1&2--$TIME when the call completed
Word 3--Send length
     4--Receive length
     5--Status/Request/Function
     6--Event/State
     :  [Event/State repeated as needed]
     :  -1 indicated end of event/states
     :  There may be another entry, beginning with Status
    50 words maximum
```

DRIVER STATISTICS RECORD

```
Word 0--Trace type = -2
     1--Read requests
     2--Write requests
     3--Messages transmitted
     4--Spurious interrupts
     5--Line errors
     6--NAKs received
     7--BCC/Parity errors
     8--Long timeouts
     9--Response errors
    10--Response rejects
    11--WACKs/TTDs received
```

DS/3000 REQUEST/REPLY RECORD

Word 0--Buffer from 1000=0/Buffer from 3000=1
Words 1 up to 308--DS/3000 message

(For descriptions of the DS/3000 message formats, see Appendix B of this manual.)

QUEX (PSI BISYNC)

For PSI BISYNC links to the HP 3000, either QUEX or D3KMS write records to the trace LU. The driver functions are not recorded.

TRC3K

TRC3K is used to access the logging file created by QUEX/QUEZ and print a report on the contents. This program is similar to TLOG, the program used for DS/1000 messages.

To run TRC3K enter:

```
:RU,TRC3K,<command input>,<logging input>,<trace output>
```

Where:

Command Input - is the LU number or file NAMR to be used for reading TRC3K commands. If an interactive LU is specified, TRC3K prompts for commands with "/TRC3K:". The default is the value returned by LOGLU.

Logging Input - Is the LU number or file NAMR of the logging data recorded by QUEX/QUEZ. If this parameter is not provided, TRC3K prompts for it with "LOGGING INPUT:". For correct operation, QUEX/QUEZ must not be writing to this file while TRC3K is reading from it. If this parameter is not provided you will be asked for the file name or LU. If you provide an invalid namr, the program will ask again for the correct namr.

Any negative number will cause the program to terminate.

Trace Output - Is the LU number or file NAMR to be used for writing the trace results. Default is <command input> (if interactive) or the value returned by LOGLU.

TRC3K commands consist of a command name followed by up to six parameters. Users can enter as little of the command names or keywords as the first letter or as much as the entire word, but it is the user's responsibility to enter as much of the word as is needed to uniquely identify the command. (TRC3K uses the first command or keyword, then matches the input, searching in reverse alphabetical order.)

The command line is free format. Command names and keywords are separated from one another by at least one blank, comma, or equal sign. The entire command line cannot be longer then 80 characters. If the <command input> is not the same as <trace output>, commands are echoed on the output device.

The TRC3K commands are EXIT, FORMAT, ??, LIST, PRINT and SET.

To stop execution of a TRC3K command, use RTE's BR (break) command. TRC3K will ask "CONTINUE?" and wait for a reply. If the reply is not "Y", the command will be aborted and TRC3K will prompt for a new command.

?? Command

This command prints the TRC3K commands along with a brief description. This command works only when listing device is interactive.

```
/TRC3K:??


 COMMAND              DESCRIPTION

  ??               DISPLAY COMMANDS
  EXIT             END TRC3K
  FORMAT           SET OUTPUT LISTING FORMAT
  LIST             SET LISTING
  PRINT            PRINT SELECTED BUFFERS
  SET              LIMIT RECORDS TO BE PRINTED
```

EXIT Command

This command terminates TRC3K.

FORMAT Command

The format command has the following syntax:

```
       [ HEADER   ]
FORMAT [APPENDAGE][,DRIVER]
       [  DATA    ]
```

This command tells TRC3K what to print when the PRINT command is entered.

The following keywords are used with the FORMAT command.

HEADER - a description of the DS/3000 header is printed.

APPENDAGE - An octal and ASCII listing of the appendage is printed in addition to the header.

DATA - An octal and ASCII listing of the data is printed in addition to the header and appendage.

DRIVER - DVG67 trace and long term statistics records are printed. This option is independent of the others.

When TRC3K is first initiated, FORMAT is set to HEADER, DRIVER.

LIST Command

LIST = <output>

This command sets the output device to the Logical Unit number or file NAMR specified by <output>.

When TRC3K is initiated, the output device is set by the second parameter in the run string or defaults to the user terminal.

PRINT Command

The PRINT command has the following syntax:

```
            [ ALL ]
    PRINT   [FIRST]
            [NEXT ]
```

This command prints a trace report on the output device. Only those buffers that meet the qualifications established by the FORMAT and SET commands are printed.

The following keywords are used with the PRINT command:

ALL - all buffers which qualify are printed.

FIRST - only the first record which qualifies is printed.

NEXT - the next buffer in the file which qualifies is printed.

If no keyword is specified, NEXT is the default.

SET Command

The SET command has the following syntax:

```
SET [<keyword>=<value>[,<keyword>=<value>...]]
```

Where:

<keyword> - is one of CLASS, RTENO, ENDREC, STARTREC, SEQUENCE, or STREAM.

<value> - is an integer (to set a value) or the character @ (to release a value).

Using no parameters causes TRC3K to print the current value for each keyword. "SET @" resets all values.

This command restricts the scope of the PRINT command. If DS/3000 request/reply buffers are included in the current FORMAT, only those which meet the qualifications set by this command will be printed.

The keywords have the following definitions:

CLASS - The DS/3000 message class. See Appendix B, for a list of classes.

ENDREC - The record number indicating where to stop reading the logging input file.

RTENO - The RTE process number ("from" number in messages from RTE, "to"number in messages from MPE). This is the terminal LU of the RTE user.

STARTREC - The record number indicating where to start reading the logging input file.

STREAM - The DS/3000 message stream. See Appendix B.

Example TRC3K Output

The following is a sample of the output from TRC3K.

```
:RU,TRC3K
LOGGING INPUT:BIGTRC
                              ***** DS/1000-3000 TRACE *****

/TRC3K:??

 COMMAND         DESCRIPTION

  ??          DISPLAY COMMANDS
  EXIT        END TRC3K
  FORMAT      SET OUTPUT LISTING FORMAT
  LIST        SET LISTING
  PRINT       PRINT SELECTED BUFFERS
  SET         LIMIT RECORDS TO BE PRINTED


/TRC3K:PRINT FIRST

RECORD      1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN     22
    HELLO          REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*       B 1-


/TRC3K:FORMAT DATA
/TRC3K:PRINT FIRST

RECORD      1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN     22
    HELLO          REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*       B 1-
APPENDAGE LENGTH    14 WORDS
  044105 046114 047440 043111 042514 042056 051525 050120*HELLO FIELD.
  047522 052054 044120 031462 031060 034440               *SUPPORT,
                                                          *HP32209
DATA LENGTH      0 BYTES
```

Consult Appendix B for further information on this message.

```
/TRC3K:PRINT

RECORD      5,  MESSAGE FROM 1000:
 CLASS  5  STREAM 23  FROM  66  TO  30  SEQ  66057  TOTAL LEN     10
```

```
    TERMINAL CNTRL REPLY
HEADER
  005005 000000 100023 000000 041036 066057 000000 000004*        B 1/
APPENDAGE LENGTH     2 WORDS
  001000 000000                                       *
DATA LENGTH      0 BYTES
```

Lets try to find all the "HELLO's".

```
/TRC3K:SET STREAM = 20,CLASS=6,STARTREC=1

/TRC3K:PRINT ALL

RECORD     1,  MESSAGE FROM 1000:
 CLASS  6  STREAM 20  FROM  66  TO   0  SEQ  66055  TOTAL LEN     22
    HELLO          REQST
HEADER
  013006 000000 000020 000000 041000 066055 000000 000033*        B 1-
APPENDAGE LENGTH    14 WORDS
  044105 046114 047440 043111 042514 042056 051525 050120*HELLO FIELD.
  047522 052054 044120 031462 031060 034440              *SUPPORT,
                                                         *HP32209
DATA LENGTH      0 BYTES

RECORD     8,  MESSAGE FROM 3000:
 CLASS  6  STREAM 20  FROM  30  TO  66  SEQ  66060  TOTAL LEN      8
    HELLO          REPLY
HEADER
  004006 000000 100020 000000 017102 066060 000000 000000*         B10
APPENDAGE LENGTH     0 WORDS
DATA LENGTH      0 BYTES
```

Looks like there's only one.

```
/TRC3K:SET STREAM = 21    Try to find the BYE messages.
/TRC3K:SET STARTREC = 1   Start at record 1.
/TRC3K:PRINT
** NONE QUALIFIED **
/TRC3K:
```

No BYE messages.

```
/TRC3K:EXIT
```

1000 – 1000 Trace – PLOG and TLOG

PLOG

PLOG is a system diagnostic program that records each HP 1000-HP 1000 message that comes into a node and reply that goes out of a node, into a disc file. Request and reply buffer formats are described in Appendix A of this manual. PLOG records the system time (when PLOG retrieves the request), the entire request/reply header buffer (up to 40 words,) and optionally, up to 84 words of data, if any, associated with the request/reply.

Normally PLOG is used to record messages into a file during a period of time in which certain network activity exists which is to be studied. When the data is captured, PLOG can be terminated and TLOG used to examine the data in the trace file.

PLOG can be initialized to log on either disc or tape media. With the disc as media, PLOG uses the file as a circular buffer, keeping only the most recent requests and replies logged. The user can optionally specify the number of most recent requests/replies to save. However, PLOG will round this number up to an integer number of 128-word blocks and use the entire file for logging.

With the tape as media, PLOG does a continuous dump onto tape until receiving an "EOT" condition, or sensing a "BR" command. In either case PLOG prints a message on the console LU, then terminates.

In order to allow TLOG, the Translating Log Program, to access the binary log, PLOG allocates all its resources, except class number, globally. PLOG keeps a record of these resources in the DS/1000-IV module RES. This record includes its class number, resource number, ID segment address, the options with which it was run, the output (log) LU, and, if disc log, the current record number, record size and number of blocks of the file.

On termination, PLOG deallocates all its resources except its log file in the case of a disc log. The user is responsible for purging the file. If the log file from a previous PLOG run is still accessible when PLOG is reinitialized, PLOG will ask the user if the old file should be purged.

Scheduling PLOG

PLOG must be loaded with the "don't copy" option (DC). Cloned copies of PLOG sidestep DS security checks, and are not allowed.

```
:RU,PLOG[,Console LU[,Log Namr or LU[,Data Flag[,Number of Buffers]]]]
```

Where:

Console LU = Interactive LU for error logging. Default is your terminal LU.

Log Namr = LU, if a tape-like device (mag tape, ctu, etc) is to be used as log medium, or

= Namr (in the form name:security code:cartridge), if a disc file is to be used as a logging medium. PLOG will create the file namr and assign file type and size.

The default is the disc file 'PLOG:DS, type 2, size 102.

Data Flag = 0, if no data from request/reply buffers is to be saved, or

= non-zero, if data from request/reply buffers is to be saved.

The default is 0, no data.

Number of Buffers = Number of most recent request/reply buffers to log if a disc file is the log medium. These buffers are handled in a circular fashion, PLOG can run indefinitely and the last N buffers will be available for viewing. The actual number of buffers may be 1 or 2 greater than asked for due to disk allocation calculations.

The default is 300 buffers.

To terminate PLOG:

```
*BR,PLOG
```

PLOG will terminate upon finishing its processing of the next message buffer it retrieves after the "BR" command. Use of the "OFF" command is not recommended as it will cause the loss of PLOG's resource number to the system and will leave the log file open. In a very quiet network it may be necessary to generate traffic for PLOG to notice the Break Flag set by the *BR,PLOG command. One simple way to do this is by entering the following commands:

```
*RU,REMAT
$TI
1979 276  14   7  36
$EX
```

Note: TLOG and PLOG must be of the same revision level to operate correctly.

PLOG can be suspended by TLOG locking its resource number for immediate translation of logged buffers for as long as it takes TLOG to make a copy of the log file. In this case, the most recent request/reply buffers could be backlogged since PLOG will not write out to the disc with a locked RN. These buffers are not lost unless the monitor queue limit is exceeded (see warning), and will be written to the disc file as soon as TLOG's lock of the RN is removed. TLOG can suspend PLOG only when disc is the log medium.

Warning: If the log device is too slow (e.g., paper tape) or there are too many request/reply transactions, and/or the system is busy with many high priority programs, PLOG may get behind in logging the data to the device. If the backlog consumes too much SAM (controlled by #QLIM), some transactions might be lost. Again, this only happens if the logging is too slow for the traffic. Thus, the user should be aware of the traffic rate and should match the log device accordingly.

Each time a transaction is lost, a "lost message" counter is incremented and PLOG saves this information in the log file. If any transactions have been lost, when the TLOG command "PRINT ALL" is used, two types of warnings are issued:

1) ** WARNING XXX MESSAGES NOT LOGGED THIS SESSION

2) ** WARNING YYY MESSAGES NOT LOGGED

The first warning indicates that XXX messages were lost during the entire logging session, while the second warning is printed between two logged transactions and indicates that YYY messages were lost between those 2 transactions. (The sum of these messages should always be less than or equal to the total announced in warning 1.)

The following example will demonstrate both the operation of PLOG and TLOG described in the next section:

```
:RU,PLOG
 ** DUPLICATE FILE -- PURGE OLD FILE? (YE/NO)
YES
 ** START PLOG
```

(Here FMGR is waiting on PLOG to exit before prompting for additional commands. Use the RS break mode command to restart FMGR.)

```
S=20 COMMAND ?RS
FMG20 ABORTED

:RU,REMAT
$SW,600,,DS
#TI
1980 158  11  58  11
#SW,LO
```

(Now to stop PLOG's logging, "BReak" PLOG.)

```
$BR,PLOG
```

(PLOG will exit on the next message that comes in. If no more traffic is coming you will have to generate some to get PLOG to exit. A REMAT remote or local TI command will accomplish this.)

```
$TI
1980 158  11  59  58
$EX
 ** END PLOG
 $END REMAT
```

(Now run TLOG, described in the next section, to see what has been recorded.)

```
:RU,TLOG

TLOG?  FORMAT BOTH

TLOG?  PRINT ALL

REQUEST LOG      TIME: 11:58:7:70
STREAM WORD:  006007  (OR)
SEQUENCE NO.: 277
SOURCE NODE:  500
DEST. NODE:   600
HEADER: 006007 000425 000764 001130 000002 052111 *        X  TI
        000000 000400 042072                      *     D:
```

```
    REPLY LOG       TIME: 11:58:7:82
    STREAM WORD:  056007  (OR)
    SEQUENCE NO.: 277
    SOURCE NODE:  500
    DEST. NODE:   600
    HEADER: 056007 000425 000764 001130 000000 000000 * \      X
            001130 000001 000000 000000 000000 000000 *  X
            000000 000012 030471 030060 020061 032470 *     1980 158
            020040 030461 020040 032470 020040 030461 *   11  58  11
            000000 000400

  REQUEST LOG     TIME: 11:58:16:23
  STREAM WORD:  016007  (OR)
  SEQUENCE NO.: 278
  SOURCE NODE:  500
  DEST. NODE:   500
  HEADER: 016007 000426 000764 000764 000000 000000 *
          000000 000001 000000 000000 000000 177762 *
          012024 000010 041122 026120 046117 031060 *     BR,PL020
          020040 000400                             *

    REPLY LOG       TIME: 11:58:16:31
    STREAM WORD:  056007  (OR)
    SEQUENCE NO.: 278
    SOURCE NODE:  500
    DEST. NODE:   500
    HEADER: 056007 000426 000764 000764 000000 000000 * \
            000764 000001 000000 000000 000000 177762 *
            012024 000000 041122 000400               *     BR

TLOG?  EX
 ** END TLOG
```

TLOG – Interactive Log Translator

TLOG is an interactive translator of the PLOG "trace" file, with limited editing facilities.

TLOG commands are in the form of command names followed by up to six parameters. When entering the command names and keywords, the user may enter the entire command name or keyword or as little as the first letter to indicate which command or keyword is desired. For some commands and keywords only the first character is necessary, but enough characters must be entered to uniquely identify the command name or keyword.

TLOG makes no assumptions about the command names, or keywords. Any part of them which is entered must be typed correctly. Commands and keywords are searched in reverse alphabetic order. The command line is free format and blanks may appear anywhere in the line except blanks may not be embedded in command names or keywords. Command names and keywords must be separated from each other by at least one blank, comma, or equal sign. Command lines may not extend past 72 characters.

All characters following a matched command name, or keyword, until the next blank, comma, or equal sign, are ignored. All TLOG commands are echoed on the list device if the list device is not the same as the input device.

Running TLOG

To initiate TLOG:

```
:RU,TLOG [,Input LU or Namr [,Log File Namr [,List LU]]]
```

Where:

Input Namr = Interactive LU for command input, or

= NAMR of a file containing commands for input.

The default is your terminal LU if activated interactively, otherwise LU 6 is used.

Log Namr = LU of a device containing log information, or

= NAME of a file containing log information.

The default is disc file 'PLOG:DS.

List LU = LU of the list device.

The default is the Input LU if interactive, otherwise LU 6 is used and a hard copy generated (assuming LU 6 is a hard copy device).

The TLOG prompt is "TLOG?". Any of the following commands can be entered at the prompt:

??, BALANCE, EXIT, FIND, FORMAT, LIST, PRINT, or TIME.

TLOG can be run concurrently with PLOG if logging is done to a disc file. However, TLOG will suspend PLOG until it makes a copy of the log file in the temporary file 'TLOG:DS. It is better to terminate PLOG before running TLOG in order to avoid the possibility of excessive logging message congestion. If initiated during a tape log, TLOG will terminate itself if PLOG is still running.

TLOG should only be terminated using the "EXIT" command. Do not use RTE's "OF" command as this could leave any files accessed by TLOG open.

To cause TLOG to break and terminate processing of the current command, enter:

*BR,TLOG

TLOG will terminate the current command and issue a prompt, if in interactive mode, or retrieve the next command from a command file. The "BR" command is effective only on the following TLOG commands:

BALANCE, FIND, PRINT, or TIME.

TLOG Commands

TLOG commands:

?? (HELP) COMMAND

This command prints out the legal TLOG commands followed by a brief description of their usage.

BALANCE COMMAND

The BALANCE command matches the request buffers with their associated reply buffers in the log file. This command operates on all request buffers logged, or if the window is set by the TIME command, all request buffers within the window, against reply buffers. The number of requests which do not have corresponding replies is printed on the list device, and on the input device if interactive and different from the list device. The unbalanced request buffers are also listed as specified by the "FORMAT" command. If more than ten unbalanced requests are discovered at one time, TLOG prints a descriptive message, then lists the ten requests already recorded according to the "FORMAT" specifications.

NOTE:

It is possible, in a busy system, for the most recent requests to be unbalanced by replies merely because logging was stopped before the replies were received.

EXIT [PURGE] COMMAND

This command terminates TLOG and optionally causes the log file to be purged.

KEYWORDS:

PURGE - purge the log file from the system.

NOTE:

The log file is not purged from the system if PLOG was using it when TLOG was invoked. If PLOG was using the log file when TLOG was invoked, PURGE only purges the temporary file 'TLOG:DS and the log file is not effected.

FIND COMMAND

FIND [ITEM[=ITEM VAL[,ITEM[=ITEM VAL[,ITEM[=ITEM VAL]]]]]]

Where ITEM is one of: STREAM, SEQUENCE, SOURCE, DESTINATION, 3000,
LINERR, BUSY, or REJECT.

ITEM VAL is an integer, or character string used as the search value of one of the following items: STREAM, SEQUENCE, SOURCE, or DESTINATION. If any of these four keywords are specified, an ITEM VAL must be given.

This command causes TLOG to search for request/reply buffers which match the specified condition(s). The search is performed in least-recent to most-recent order.

If the window is set by the TIME command, only the buffers within the window are searched. Buffers are translated and interpreted according to the "FORMAT" command, then listed on the list device as they are found.

TLOG also keeps a running count of all buffers which match the condition(s) and prints this count at the end of the search on both the list device and the input device (if interactive).

If more than one search item is specified, the conditions are treated as a logical AND, i.e., the buffer must match all specified conditions.

The following is a list of possible values for "ITEM":

- STREAM - the DS stream type. ITEM VAL must be DLIST, EXECW, PTOP, EXECM, RFA, or OR.
- SEQUENCE - the DS local sequence number. ITEM VAL is an integer.
- SOURCE - the node at which the request/reply originated. ITEM VAL is any non-negative integer.
- DESTINATION - the node to which the request/reply was destined. ITEM VAL is any non-negative integer.
- 3000 - specifies a check on the 3000 bit in the buffers for those which are set; i.e., those for which the HP 3000 is master.
- LINERR - specifies a check on the line error count in the buffers for those which are non-zero.
- BUSY - specifies a check on the remote busy count in the buffers for those which are at the maximum retry count.
- REJECT - specifies a check on the busy reject bit in the buffers for those which are set.

Note: No more than three items may be specified as search values. If the user specifies more than three items, an error message is printed and the command ignored. If no item is specified, FIND will match every buffer (i.e. same as PRINT ALL except a running count will be taken as well).

EXAMPLE: FIND STREAM=OR, SOURCE=1, DEST=2

FORMAT COMMAND

FORMAT [NORMAL/BUFFER/DATA/BOTH/OFF]

This command sets the format for output listing.

The following keywords can be used with the FORMAT command:

- NORMAL - The first 4 words of the request and reply, are listed without data.
- BUFFER - The entire request/reply buffer (up to 39 words) are listed.
- DATA - Any data (up to 84 words) logged with the request/reply is printed.
- BOTH - Both the request/reply and data buffer are printed.
- OFF - The output listing is turned off. This is useful in conjunction with the FIND command. A running count of all buffer which match the condition(s) will be taken without listing the buffers.

The default is NORMAL.

LIST COMMAND

LIST = lu

This command directs further listing to lu. The default is set by LIST LU in run string RU,TLOG.

PRINT COMMAND

PRINT [ALL/FIRST/LAST/NEXT]

This command selects further buffers to print. The data printed is qualified by the FIND and TIME commands.

The following keywords are used with the PRINT command:

- ALL - all logged buffers are listed, or if the window is set by the TIME command, all buffers within the window are listed. The buffers are translated and interpreted as specified by the FORMAT and TIME commands.
- FIRST - the first logged buffer is listed, or if the window is set, the first buffer within the window is listed. The

buffers are translated and interpreted as specified by the FORMAT and TIME commands.

LAST - the last logged buffer is listed, or if the window is set, the last buffer within the window is listed. The buffers are translated and interpreted as specified by the FORMAT and TIME commands.

NEXT - the next logged buffer is listed. The buffers are translated and interpreted as specified by the FORMAT and TIME commands.

If no keyword is specified, default is NEXT. Once the PRINT command has been specified, each additional carriage return will issue the command again and again.

If any messages are lost during logging, warnings will be printed indicating:

1) The total number of messages lost during the entire logging session

and

2) the number of messages lost between any two logged transactions, if any.

These messages will only be printed when the PRINT ALL command is used.

TIME COMMAND

TIME [= time value 1[, time value 2]]

This command sets a window for any subsequent commands until the next TIME command. A count of the buffers which fit in this window is printed on the LIST LU and the interactive input LU.

If time values are defaulted, the lower limit becomes the smallest number the TIME can assume and the upper limit becomes the largest number the TIME can assume. In the case where both are defaulted, a count is not printed. Limits are inclusive. When the TIME command is issued, any previously set limits are removed.

TIME VALUE 1 - The lower bound specified by HH:MM:SS:LL (HH = hours, MM = minutes, SS = seconds, and LL = 10's of milliseconds).

TIME VALUE 2 - The upper bound specified the same as above.

Chapter 2
Troubleshooting

The procedures described in this chapter are designed to help in the event of a failure in your network. They are general in nature because the troubleshooting process cannot possibly be reduced to a small set of "cookbook" procedures. It is hoped that they can be used by on-site personnel to resolve most of the problems which are likely to occur. Hewlett-Packard also offers a number of service policies during and following the warranty period. Hewlett-Packard strongly recommends that one of these policies be utilized. Contact your local Sales Representative for details.

The techniques described in this chapter can be applied by anyone. However, the most effective troubleshooter is one who understands the workings of the system, described in Chapter 5. Understanding the software and the reasons it has for giving various error indications will help you narrow the possibilities quickly. If at all possible, you should attempt local execution of any subroutine call which fails. This eliminates all communication-line-related error possibilities. If the attempt still fails, a software-related problem is indicated. Check that you are executing the subroutine call with the proper parameters and that their values are correct. Generation-related problems may also be indicated if time-out errors, or errors indicating insufficient class or resource numbers, or not enough System Available Memory (SAM) has been provided. Table 2-1 shows a checklist for generation related error possibilities.

With rerouting in the system, link failures may go unnoticed by the user software. Periodic examination of rerouting parameters with DSINF will allow problems to be detected and corrective action to be taken.

Table 2-1. Checklist for RTE Generation

Network Routing Vector:	Use DSMOD's /N command to print the NRV. Compare it to the network configuration. Make sure node numbers and LUs are correct. Make especially sure the "upgrade levels" are correct for each node. If zero is shown for the LU then there is no path to this node at this time (except for the local node).
EQT Table: Check:	1) one EQT for each link for DVA65 or DVG67 links, two consecutive EQTs for each DVA66 link 2) number of words in extent is correct 3) line time-out specified. The same time-out must be specified for both ends of the line (see the Generation chapter). The timeout should normally be set to 0 for DVA66 links. See the recommendations in the Generation and Initialization Chapters for DVA65 links.
Device Reference Table:	1) at least one LU for each DVA65 or DVG67 link 2) 2 consecutive LUs for each DVA66 link 3) even subchannel for modem interface (12773/DVA65) 4) odd subchannel for hardwire interface (12771/DVA65) 5) subchannel settings unnecessary on the DVA66 and DVG67 links.
Interrupt Table:	1000-to-1000 Links and 1000-to-3000 links 1) for DVA65, DVA66 and DVG67: sc,EQT,eqt where: sc = select code eqt = EQT number NOTE: This is a change over the previous product DS/1000. Failure to use the new format will cause problems.

Troubleshooting Utilities

Several utilities are provided with DS/1000-IV to aid in finding problems in the network. These utilities are discussed in detail in the chapter on Network Maintenance Utilities.

Before doing anything to the hardware when trying to bring up a new generation be sure and use the software tools available to check out any errors in your DS subsystem.

Use DSMOD's /N command and display the NRV. Be sure all nodes reflect the correct "level" numbers! (1 = DS/1000-IV and 0 = DS/1000). If these numbers are not correct, the incoming messages to the "true" level 0 side will be flushed because they won't have been converted to DS/1000 format first. The sending side will get time-out errors. If the level is wrong between two level 1 nodes, unnecessary message conversion will be performed.

The answers provided in the DINIT answer file can be a source of problems especially when the network has several nodes. If you first attempt to initialize the large network and have problems, try to initialize two adjacent nodes only. This will simplify your network and answers and maximize your chances of success.

A common problem is the answer provided for rerouting links. When you respond to the NRV questions:

```
/DINIT: CPU#, LU, TIMEOUT, LEVEL, NEIGHBOR, MA
```

You must not provide LUs for the rerouting links or rerouting will not be activated. You must leave the LU parameters blank (not zero). When the LU is enabled, rerouting will determine which LU is best to use to reach each destination. The answers should be provided as follows:

```
100               * local node
200,,10,1,1,MA    * Node 200 rerouting
300,,10,1,1,MA    * Node 300 rerouting
400,24,           * Node 400 no rerouting;
                    no Message Accounting
```

DSINF

The DS Information Utility provides more information than DSMOD provides but does not allow modification of the values. It prints the class table, dumps the SAM block, lists the master and slave transaction control block lists, dumps EQT entries and extracts long-term statistics from the HDLC and BISYNC cards. DSINF will print message accounting values, rerouting values and Remote Session parameters.

The DSINF LU command is especially useful for retrieving flags from the HDLC card to indicate the card state. An example is provided below.

```
EQT     4, LU    67, TYPE  66

FLAG BITS (EQT WORD 26)
0 READ ABORTED     0 WRITE ABORTED   0 RD RQ PENDING    0 WT RQ PENDING
0 BKPL LOCKED RP   0 BKPL LOCKED WP  0 SHORT TO ACTIVE  0 MED. T.O ACTIVE
0 LONG T.O ACTIVE  1 CONNECTED       1 START OF MSG.    0 NON-DS MODE
1 ASKED TO CONNCT  0 SEVERE ERROR    1 P-F RECONNECT    0 RFP WAIT
```

For the card to function for DS, the "ASKED TO CONNCT" flag must be set. This indicates the DS driver has asked the card to establish a connection with the other end of the link. The "CONNECTED" flag indicates the card was successfully able to connect with the other end of the link. One more critical item is the "NON-DS MODE" flag. This flag should be in the zero state to indicate it is in DS mode and not still in a VCP or RPL mode. Aborting DSVCP can leave the link unusable for DS and is indicated by a one state. This and the following utilities are described in the Maintenance Utilities Chapter.

PLOG/TLOG

Plog turns on the 1000-1000 trace feature and records all messages that are processed by the node into a file. This file is later printed with the TLOG (Translate Log) program described in detail in the chapter on Network Maintenance Utilities. The data that TLOG presents can be very valuable in determining software failures in the DS system.

LOG3K/TRC3K

These programs provide a similar trace capability for messages on the 1000-3000 link.

SLCIN

SLCIN prints the Events Trace Table maintained by DVG67, the driver for the 1000-3000 hardwired 12889 card. The trace table shows the last several line transactions that were processed by the driver. This information will prove most useful to the person knowledgeable about the BISYNC protocol and to the HP field personnel analyzing problems with the link.

Isolating Hardware Failures

There are three ways to isolate hardware failures: substitution of spare pieces of equipment, self-test, and the use of the diagnostics. Diagnostics for each piece of equipment are described in the hardware documentation covering the equipment.

In a store-and-forward environment such as DS/1000-IV, a failure at one node or at one link may cause error messages to appear at nodes which are not even directly connected to the node or the link which failed.

As a troubleshooter, your first task is to determine which node or link has failed. If the suggestions in the DS/1000-IV User's Manual have been followed, the applications software will print the node number which reported the error (certain errors are reported by one node due to conditions at another), as well as the error code. Use the Error Codes section in the DS/1000-IV User's manual to determine possible causes for this error.

If hardware failure is a possible cause, utilize alternate equipment, if available, to isolate the failure as quickly as possible. The hardware diagnostics can be run quickly and are easier to perform than looking for bugs in software. However, some diagnostics are not designed to function in the RTE operating system environment and must be run with the computer in a stand alone mode (real-time activities must cease). If alternate nodes exist, a node may be taken off-line to run diagnostic checks without unduly disturbing the application users. If not, there may be planned shutdown periods which can be utilized.

The 91711A Diagnostic and Verification Package contains a diagnostic TXDS0 which can be used on-line to verify your DS links. This diagnostic is described in manual part number 91711-90001.

The RTE-L/XL Datacommunication Diagnostic Package, 24600A, provides on-line diagnostics for the PSI card. The package instructs the PSI card to run through self-test and report the card status.

You can usually isolate the problem to the 1000-1000 link level by using REMAT. Send TEllop or TI commands from each of the nodes you suspect to each of their neighbors. If communications function properly to or from any node the chances are good that the computer, DS/1000 (91740) Firmware (used only with DVA65, if present), and disc (if it exists) are OK, so concentrate on the interfaces, cables, and modems (if phone lines are used). To use the TEllop procedure, at least one side of every link needs an operator (but not necessarily both sides). However, if you notice that you can't transmit or receive messages from another node, you can't always tell whether the computer or the link is the culprit. The error message will give you a clue.

You can isolate the cause of a DS02 error if two or more links exist to an unattended node. Send a message via REMAT over both links. If the computer is OK, one of the messages should get through. If you get a DS01 (line parity) or DS02 (line timeout) from the other node, the link is bad. In the case of a DS02 error, try increasing the line timeout (e.g., double it). If the error disappears, then your line timeout value is too low. Increase it from its current value slightly and test the system, increasing it until a satisfactory value is found. Remember, both sides of the link should have the same EQT timeout values.

Intermittent failures can be detected using the same procedures above, but change TEllop to a STore command using a fairly large file. Sending a large file should exercise the communications line and interface cards enough that an intermittent failure can be identified. Loose interface card hoods or damaged cables can be a source of intermittent failures. Try wiggling them a little during the test.

The DSINF EQ or LU command with the statistics option will show the line error statistics which are useful in determining if the line has deteriorated noticeably. Refer to Chapter One for these statistics.

1000-3000 HSI Links

After a link to an HP 3000 is enabled, the "UP" message from QUEX should be printed on the system console within about one minute. If the message is not printed, check the following:

- o Are the proper versions of QUEX and QUEZ loaded for the link type being used? (If the wrong version is loaded, the program will print an error message and go to RTE state 6-suspended.)
- o Has the DSCONTROL command to open the line been entered at the 3000 system console? (Check the device with the MPE SHOWDEV command.)

- o Are the cables and connectors properly connected? (A continuity check of both cables may be necessary.)
- o Does SLCIN inidicate that the HP 3000 is sending BISYNC messages? If not, see if the DSCONTROL command was entered at the HP 3000.

Information about the type of line failure is printed by QUEX in the "DOWN" message.

If neither an "UP" or "DOWN" message is printed by the HSI version of QUEX and it remains I/O-suspended (state 2) the timer on the board may not be working. Run the HSI hardware diagnostic.

1000-3000 PSI Links

When troubleshooting a 1000-3000 PSI link, check the following items:

- o The PSI card.
- o The LU, EQT and SC mappings (IFT, DVT, LU and SC mappings in L/A-series)
- o The physical connection between the HP 1000 and the HP 3000 (the cable or modems).
- o Software compatibilty between the HP 1000 and the HP 3000 (code versions, speed, frame size).
- o Are all the necessary DS/1000-IV monitors running?
- o The HP 3000--is it ready for communication?

The first step in troubleshooting a 1000-3000 PSI link is to test the PSI card. Run the card through its lighting sequence and diagnostic hood self-tests. Refer to the appropriate firmware manual for instructions.

Next, run DSINF and issue the LU command with the AL option (LU,xx,AL-- described in Chapter 1). If you get the message "DRIVER REPORTS ERROR", either the system cannot communicate with the driver and/or the driver cannot communicate with the card. Check the LU, EQT and SC mappings (IFT, DVT, LU, SC mappings in L/A-series). Check that you enabled the correct LU at network initialization time. Test the card again.

When you can issue the DSINF LU,xx,AL command without getting the "DRIVER REPORTS ERROR" message, note the value for "SPEED" to compare later with the HP 3000.

Next, run DSLIN.

- o If you run DSLIN in primary mode and get the message "PRIMARY CONNECT TIMED OUT", DSLIN was unable to communicate with the PSI card. Check the interface mappings again.

- o If you run DSLIN and get the message "LINE IS UP WITH BUFFER SIZE xxx", the PSI link should be available for 1000-3000 communication You may want to check the buffer size with the DSINF VA command (described in Chapter 1).

- o If you run DSLIN and get the message "LINE IS UP BUT HP3000 NOT REPLYING", check the cable or modems. Be sure that the DSCONTROL command was issued at the HP 3000. See if QUEX and QUEZ are active (state 3). If you ran DSLIN in secondary mode, you must also check that the DSLINE command is issued at the HP 3000.

If you still have problems when you run DSLIN, test the low-level (BISYNC) link. To do this, issue the DSINF LU,xxx,AL command. Look at the PSI card's "CONNECTED" and "ASKED TO CONNCT" fields.

- o If there is a zero in both fields, the system and the card may not be communicating. Using DSINF, check the counts in the card's TTD, NAK and WACK fields. If the counts are incremented when you run DSLIN, the card and the software are communicating. However, the BISYNC protocol is in error recovery mode. To clear the error, reset both cards. (On the HP 1000, run DSMOD and use the "/L" command. This closes the line, then re-enables the card.) Run the cards through self-tests.

- o If there is a zero in the "CONNECTED" field and a one in the "ASKED TO CONNCT" field, create a file to use as a log file. Run LOG3K (as described in Chapter 1) and set up the log file. Run DSLIN in primary mode. Run LOG3K and close the log file. Run TRC3K (as described in Chapter 1). Check that an initialization request was sent by the HP 1000. If not, check your card again, including its TTD, NAK and WACK counts. If an initialization request was sent, see if it was received by the HP 3000. If so, troubleshoot the HP 3000. If it was not received by the HP 3000, troubleshoot the HP 1000 also. See if QUEX and QUEZ are active (state 3).

- o If there are ones in both fields, your PSI link should be available for 1000-3000 communication. If not, set up a log file as above and trace the messages sent.

After Identifying a Bad Link

Once you've determined the faulty link, the following possibilities may exist:

- One or both interface cards may be bad

- The cable may have an open- or short-circuit

- Modem may have failed (phone line connections only)

- Line may be extremely noisy, or has drifted considerably from nominal parameters.

- The DS/1000 Driver (DVA65) Firmware, if installed, may be faulty or incorrectly installed (particularly if the RTE "crashes" immediately upon running DINIT to initialize the node, specifically when DVA65 LUs are enabled). To verify that your problem is the DS/1000 firmware, run DINIT interactively. When you enable LUs, enable all the DVA66 LUs first, then the DVA65 LUs that require the firmware. If the system crashes at this point, either the firmware is not installed, not installed correctly or is faulty.

The failure could exist at either side. If spares exist, the fastest way to identify the problem is to begin substituting cables, interfaces, and modems, one piece at a time. If your RTE was generated with LU, EQT, and interrupt table entries for a spare interface board, then you may be able to switch over the LU assignment of the bad link to the other EQT, move the cable to the spare interface, and enable the link using DSMOD. Limitations on this are:

- The spare interface board must be the same type as the suspect one (i.e., you can't substitute a modem interface for a hardwire interface, or vice versa, unless substitution is made on both sides).

- Interfaces on both sides of the link must be configured identically (speed, asynchronous/synchronous, etc.). If not, you'll have to remove the I/O board to set the switches properly. Be sure to remove power from the I/O interface first. Your application will have to be able to get along without the computer for the few minutes it takes to change the switches, as turning off the power to the I/O interfaces is required before removing any of them. Upon restoration of power, a power-fail restart will be made automatically (HP computers that support

DS/1000-IV require battery backup for this) if the power-fail software has been set up properly (See the RTE manual. UPLIN will automatically re-enable the line within 5 seconds.). See the interface hardware manual for a description of the switch settings. The HDLC and modem BISYNC boards allow the speed settings, etc., to be checked on-line; if incorrect you'll still need to shut down to make the changes.

- The LU number of the spare communications interface must have the same subchannel and the EQT must have the same line timeout.

If substitution solves the problem, proceed according to the applicable service procedures depending on whether your HP warranty is still in effect, a maintenance contract is in force, or you are using your own in-house repair facility.

Troubleshooting the HDLC Links

Troubleshooting the HDLC links is much easier, but quite different from troubleshooting 12771/12773 links, which is covered later in this chapter.

The HDLC cards have built-in self-test firmware that executes whenever the card is reset (e.g., power-up, preset, CLC 0, etc.). During a preset or CLC 0, all four LEDs on the front edge of the card go "on". During the self-test you will see LED0 off and all other LEDs "on". Upon completion of the self test, if no errors occured, the lights will all go off.

The DS software will enable this link telling it to connect with the card on the other end of the link. If the card can connect, LED"0" will light, indicating the link is initialized and functioning. This means the DS driver was able to communicate with the card and the card has communicated with the card at the other end of the link. You must also check the lighting sequence on the other interface to see that two-way communication has been established. This light may also be checked with DSINF EQ or LU commands, specifying the PA option. Look for a 1 in the "ASKED TO CONNECT" and "CONNECTED" fields.

The DSINF "LU,,ST" command, described in Chapter 1, should be used to examine the firmware Revision Code to determine that both cards have the same firmware. The baud rate and internal/external clock switch settings are also displayed and must be the same at both ends of the link in order to function properly.

This command also prints card statistics which are useful in evaluating the quality and predicting failures of a link. Refer to Chapter One for these statistics.

HDLC Card Loop-Back Hoods

All versions of the HDLC cards are supplied with a loop-back verifyier hood, one for the "DIRECT CONNECT" cards, one for the "RS-232" modem cards, and one for the "RS-449" modem cards. Since the cards are full-duplex, the installation of the hood will allow the output port of the card to communicate directly with the input port of the same card.

If, after installation of the firmware, self-test is not successful (lights on the board don't go out), performing the loop-back tests will be of no value. If the self-test is passed, this is a reasonable indication that the cards are functional. The purpose of the loop-back test is to verify that the line drivers, receivers and backplane are functioning properly. It is also used to determine if the DS/1000-IV Software and RTE Operating System have been properly generated.

To utilize the loop-back feature, install the proper hood on the I/O card which you wish to test. (There are different hoods for the modem and direct-connect boards.) The power MUST be turned off before connecting the hood. The hood is installed connector side up so the words on the hood ("DIRECT CONNECT" or "RS-232") can be read, facing up, after installation.

Apply power. The board should, again, successfully complete the self-test sequence.

Initialize DS by running DINIT, and LED "0" (counting from the right) should light. If not, check your generation and your DS/1000-IV initialization file, you may be enabling the wrong LU. At any rate the card has not been told to "connect" by the driver or download firmware. Table 2-1 provides additional things to check in your generation.

1) Are you using the right LUs when you initialize?

2) Is the card select code indicated correctly in the generation?

3) On an L-Series, do you have your select code switches set up correctly?

If everything did work and the connect light comes on here are some additional exercises to verify the generation. Using DSMOD's "CN" command, change the local node's LU from "0" to the LU of the card under test. For example, if the local node is 500 enter:

```
/DSMOD: CN
/DSMOD: NETWORK MANAGEMENT SECURITY CODE? DS   Enter DS or whatever
/DSMOD: NODE # TO CHANGE? 500                  code is correct.

NODE = 500,     LU = 0     TO = 0    LEVEL = 1,(N)

/DSMOD: LU,TIMEOUT,UPGRADE LEVEL, [N]? 64      Enter LU being tested.

NODE = 500,     LU = 64     TO = 0    LEVEL = 1,(N)
```

At this point all REMAT commands to the local node will be sent via LU 64. What this does is cause the requests and replies to be sent through the interface card with the loopback hood installed instead of just being processed locally in the node. If this works it provides further verification that the local generation and card are OK. If a failure occurs, examine the error messages and try to determine which software in the local node is not functional. Be sure to run DSMOD and change the LU for the local node back to 0 afterwards.

Loop-Back Connectors for the Cable (HDLC Cards)

If the interfaces pass the previous test, and your local node is functioning relatively well you are ready to verify your cable linkage.

The construction of the loopback connectors for the male and female ends of your direct-connect cable are described in the hardware manuals for the direct-connect cards. These connectors are not supplied by Hewlett-Packard but are very useful for cable testing purposes.

You can install your DS cable (80-pin hood to 24-pin connector) on the card. Install the appropriate loopback connector (male or female) on the end of the cable connected to the card.

At this point the same loop-back test procedures should be attempted as were done previously with the loop-back hood on the card. Don't forget to use the DSMOD "CN" command to set the LU from 0 to the LU of this card. Then don't forget to put it back to 0 when finished with the test.

Execute a "TI" command, or use a "DL" command. With the DL command you'll have time to observe the lights on the DS board. If DLIST is not available a series of TI or TE commands will do also. If the board is actually processing data LED1 (second from the right) will be lighting very briefly for every transfer from HP 1000 memory to card memory.

If the results of the "DL" command are not displayed on your screen you either have a bad cable or you've set up the test to the wrong LU or some other error. Check your set-up with DSINF /N command. Check that the card has has "connected" to itself (LU,,PA command). Otherwise get another cable and try again.

If this exercise is successful at both sides of the link then the next step is to connect up the cable connecting the two computers.

Now the cable between computers can be tested. This test is very useful for finding the right cable if the cables are kept in a trough with many others. If you think you have the right cable on the "other computer" end, and the loop back test is not working, try another cable end.

If your loop-back works, it's time to hook up to the other computer. The DS software is set up so that you can loop-back through a neighbor computer in order to verify a complete link. This is accomplished by setting the local nodes LU in the NRV to the LU used by a neighbor node using the DSMOD "CN" command. This causes all the commands destined for the local node to be transmitted to the neighbor who will forward them (loop them) back to the origin node. This procedure verifies one link (and the DS software in both nodes) and can be performed with the other links existing in the local node.

These same procedures should be followed at the other computer. Verify that its card passes self-test, check the connected light (LED0), it should light when the loop-back connector is installed on the cable or the hood installed on the card. Somewhere in the link a failure should show up and a link component (card, cable, etc.) can be replaced. If all the loop-backs work, the problem is probably in the switch settings on the cards or in the software. Never forget that many times the problem is in how the software was generated.

Troubleshooting the DVA65 Links

The following procedure is recommended for HP 12771/12773 links only. It involves having two HP 12771/12773 interfaces installed so the half-duplex transmit data can be sent out one interface and received by the other. For HP 12773 cards a modem eliminator cable will be required. Testing HDLC cards is accomplished much more easily by the tests described earlier. The HP 12771/12773 interfaces being half-duplex a more complicated procedure is required.

The self-test procedure may be used to isolate equipment failures between directly-connected RTEs. The same test is run at both suspected nodes. It operates by forcing one node to "talk to itself" and runs on-line so that real-time activities (and communications over the other lines) need not be affected. Figure 2-1 shows the configuration used by the following discussion.

Some of the advantages of the self-test are:

- The test runs on-line, so communications activity over other lines and real-time activity may proceed unaffected.

- The special Communications Firmware as well as the hardware are tested under actual real-time conditions. Off-line diagnostics cannot do this.

Each node must have at least two communications interface cards of the same type, jumpered for the same transmission speed and mode. If the node has both modem and hardwired links connected to it, two of each type of interface are required. Figure 2-1 shows the configuration.

The following test procedures will only work if the local node is the first entry in the NRV. Set up your DINIT answer file so this will be the case.

Use the DSMOD "CN" procedure described earlier to change the LU of the local node from 0 to one of the HP 12771/12773 LUs you have installed. At this point all REMAT local commands will be transmitted out of the indicated LU. With the configuration set up in Figure 2-1 the requests will loop back through the second interface and be processed by the local node.

In the following paragraphs, LUs 20 and 21 are the two interface cards used in the loop-back tests.

After using the DSMOD "CN" command to enter LU 20 for the local node, a sample REMAT dialog for the self-test on each side might look like this:

```
*ON,REMAT
$TI (or $CL)
```

Now set up the NRV so LU 21 is used for the local node.

```
$TI (or $CL)
$EX
```

The message will be transmitted out one interface and received by the other. (Note: you must have OPERM to process the TI request or DLIST for the CL request.) If everything goes well the message will appear on the system console.

Operation (DVA65 HP 12771/12773 Links)

In operation, the spare interface is connected to the suspect interface. Use DSMOD to change the NRV so that the local node uses the LU of the suspected bad interface.

Using REMAT, store information from a terminal into a file and list it back. Run this test at both suspected nodes. Reverse by switching the NRV entry for the local node to the spare interface LU. Rerun the test. This checks both interface cards for transmission and reception. Compare the results to Table 2-2.

Table 2-2. Analysis of Self-Test Results

RESULTS	INDICATIONS
Both nodes work self-test OK	Indicates cable or modem may be bad (neither of these items are checked by this method). May also indicate problem is not hardware-related.
Either or both sides fail (general)	May indicate interface failure or bad cable. (Hood to canon connector.) Check interface with diagnostic or substitute spare. If node has just been regenerated, check applicable RTE generation examples in this manual and check that the suggestions made were either followed or bear no relationship to the failure reported. Pay particular attention to generation-time answers which were different from the examples in the manual, especially if the system has recently been regenerated and link communication never verified. Check that cables are connected firmly, the plastic sheath on all cables is intact (a torn or damaged cable may indicate damage to the wires inside). Check the cable for continuity and shorts (see the interface manual for connecting pin diagrams). May also indicate improper installation of DS/1000 (DVA65) Communications Firmware if in use in these nodes.

When the self-test is run, both sides will be connected as shown in Figure 2-1.

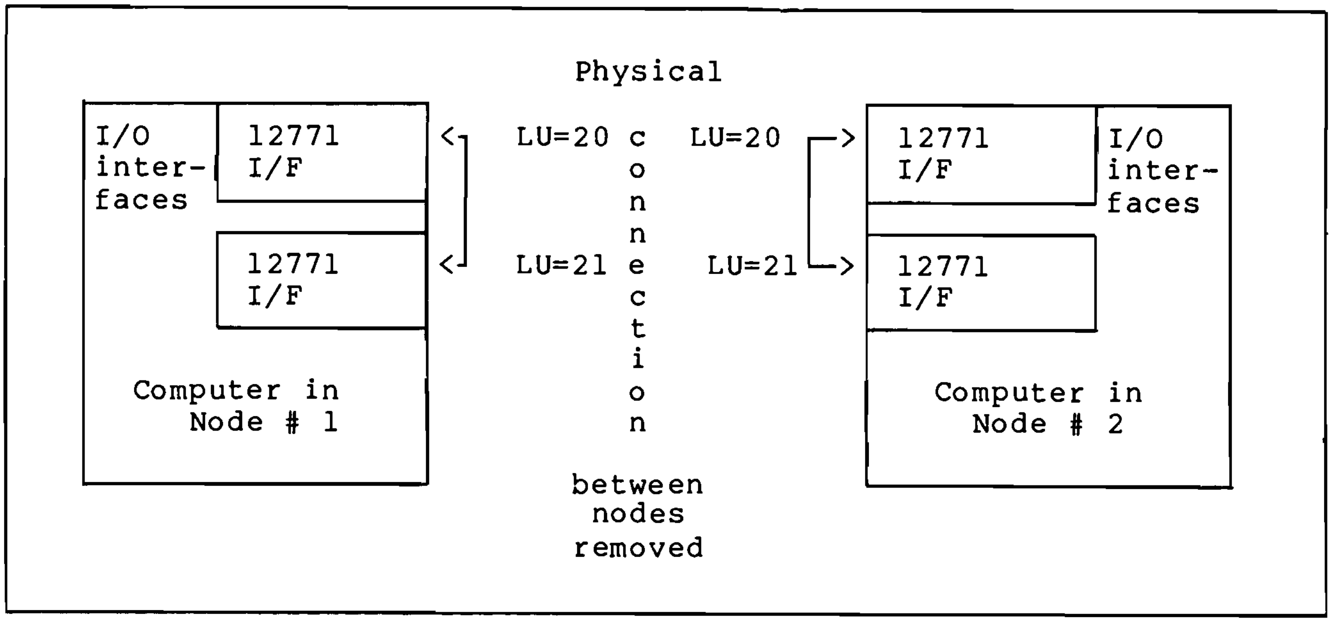


Figure 2-1. Self-Test Node Connection for 12771 Card.

Operation HP 12773 (Modem Links)

The loop-back test procedure may also be used to test modem links (modems, interfaces and cables), but a modem eliminator is required (not available from Hewlett-Packard).

Before starting the loop-back test, connect the modem on one side to the spare HP 12773 interface, test the line and if the test does not fail, then repeat this in the other side. If the test does not fail, then the other interface card is suspect. It is recommended that this be done first simply because it is easiest. This test does not require shutting down power to the I/O boards, as does exchanging interfaces.

NOTE

> After every change in connections, use DSMOD /L command to enable "listen" mode on both interfaces.

If the failure is still to be found, connect the computers as shown in Figure 2-2 and run the loop-back test on both sides.

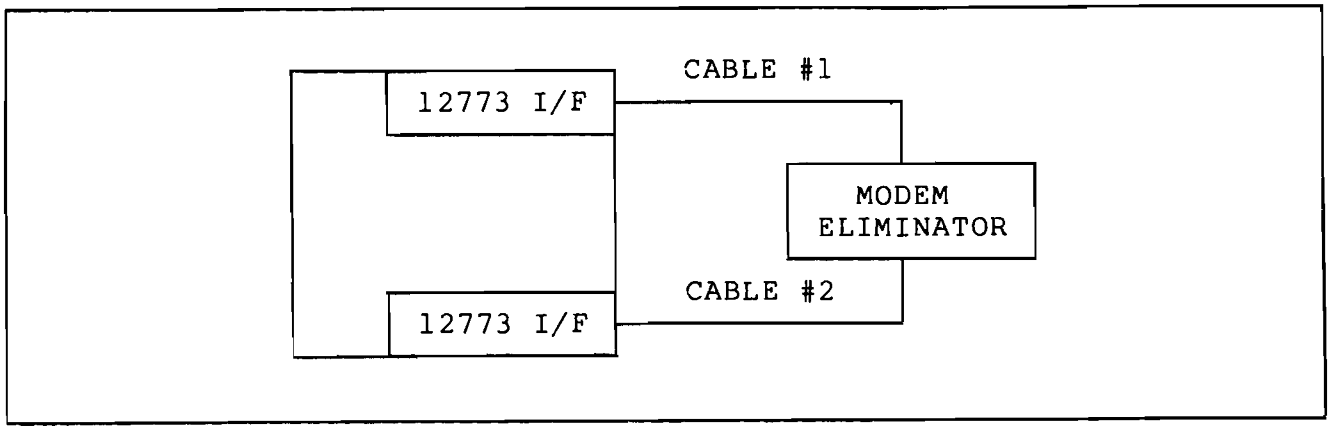


Figure 2-2. Loop-Back Test Connections (HP 12773 Cards)

If the test does not fail (no errors occur) on either side, then a failure of the modem or the telephone line is indicated. If the test fails on either side, then a failure in either the interface or cable is indicated; substitute another cable. If this works then the other cable failed. Otherwise, the interface is probably bad. To verify, connect the modem to the spare interface using either cable and switch the LU normally used for communication to the EQT for the spare interface. If the test produces no error, then the bypassed interface was in fact bad. Otherwise, contact your Hewlett Packard field support personnel.

Bypassing a Bad Link

If rerouting is generated into your system, bypassing of a bad link is performed automatically. However, to manually switchover, just use the DSMOD CN command and go to the new LU on each side.

Using Spare Links

The fastest way to bypass a bad link requires a spare link (2 interfaces and cables, plus 2 modems if phone lines are used). The procedures described below are shown without modems. They may be used either for hardwire or with modems, but in the latter case repeat the general procedure once to test the interfaces and once to test the modems. When rerouting software is generated in the following procedure is automatic.

Suppose you have the configuration shown in Figure 2-3 (other nodes may be connected to either or both sides, but we are concerned only with the failure of a single link).

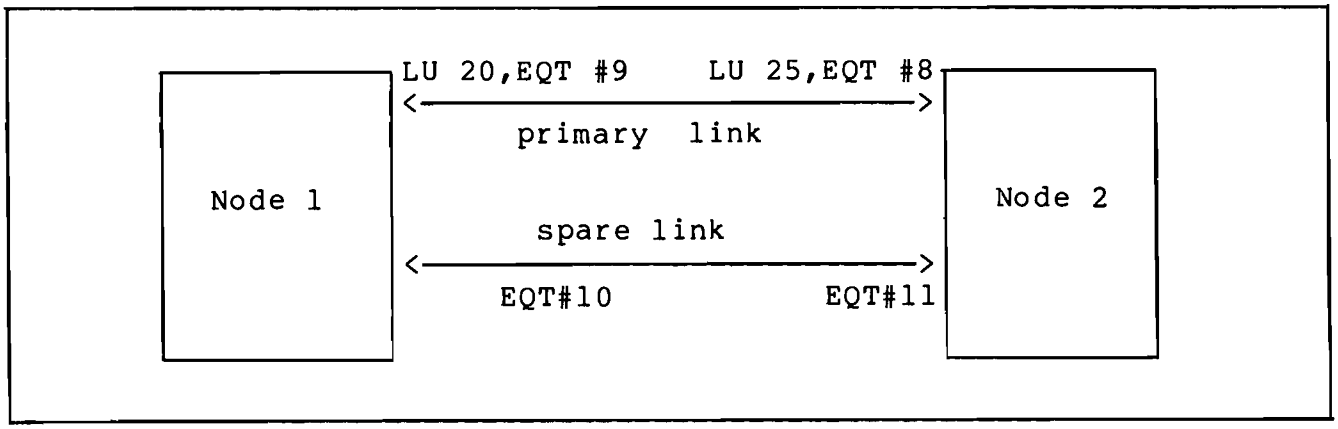


Figure 2-3. Spare Link Configuration

If the primary link fails in the case shown above, the spare link can be utilized by switching LU 20 to EQT 10 in Node #1 and switching LU 25 to EQT #11 in Node #2. This can be done on-line by the operators at each node. Be sure to set an even-numbered subchannel for modem interfaces and odd-numbered for hardwire interfaces. Use DSMOD's /L command to enable the new LU. Switching LUs may also be done with DSMOD's CN command.

This procedure can be used whether the two links are both hardwired, modem, or one of each. Naturally, if the link types are dissimilar, you must move the cables on both sides at once and you cannot move a hardwire cable to a modem interface or vice-versa. If the spare link is of lower speed, you should increase the timeouts.

If both links are of the same type and speed, you can help to localize the problem even further by switching only one side at a time. For example, in the configuration shown in Figure 2-3 above, try switching the primary link to the spare interface at Node #1 as shown in Figure 2-4.

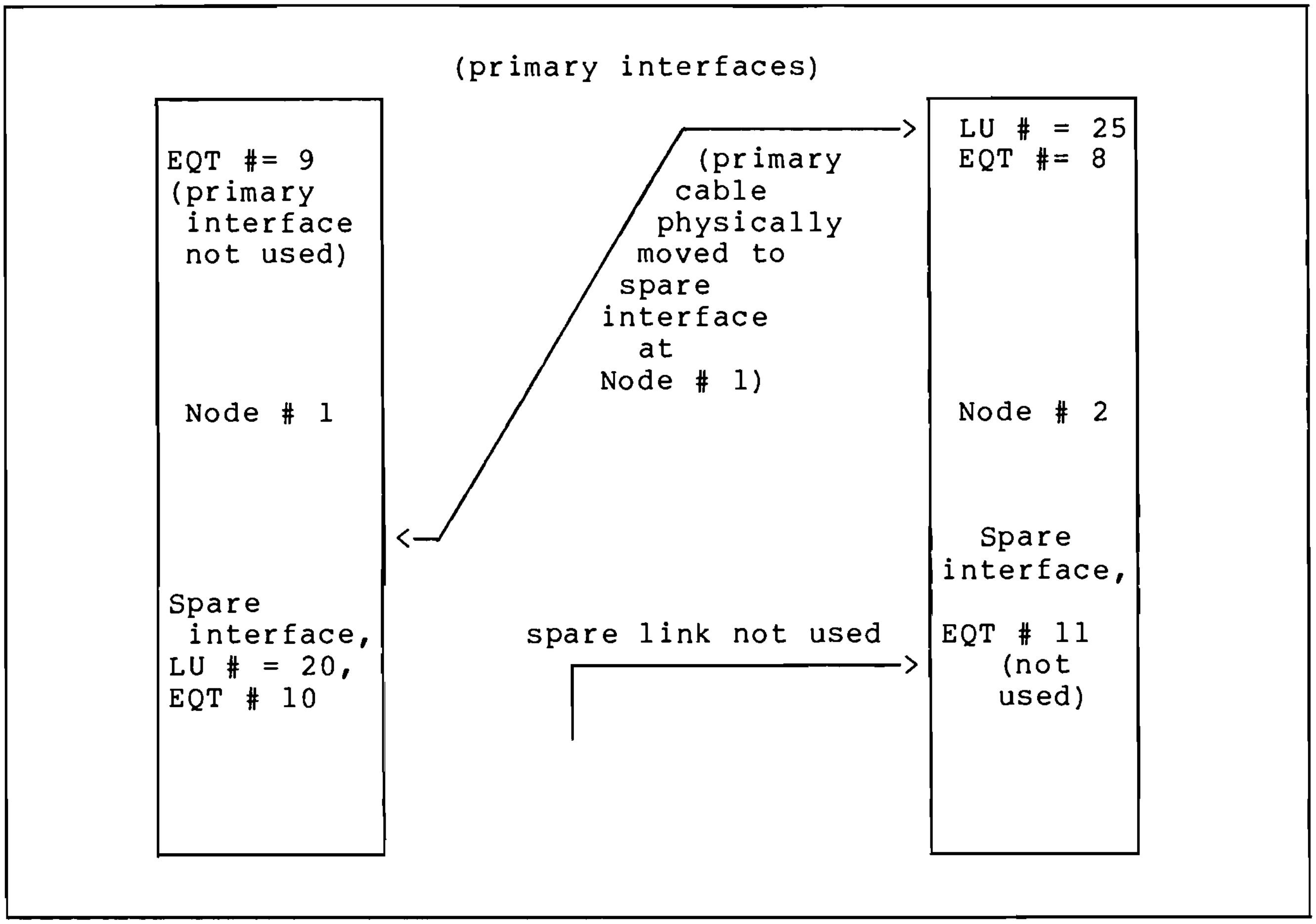


Figure 2-4. Switched Primary Link Configuration

Re-enable LU 20 and use REMAT to send messages between Nodes 1 and 2. If communications resume, the interface card in Node 1 corresponding to EQT # 9 is bad. If the problem still persists, switch to the configuration in Figure 2-5.

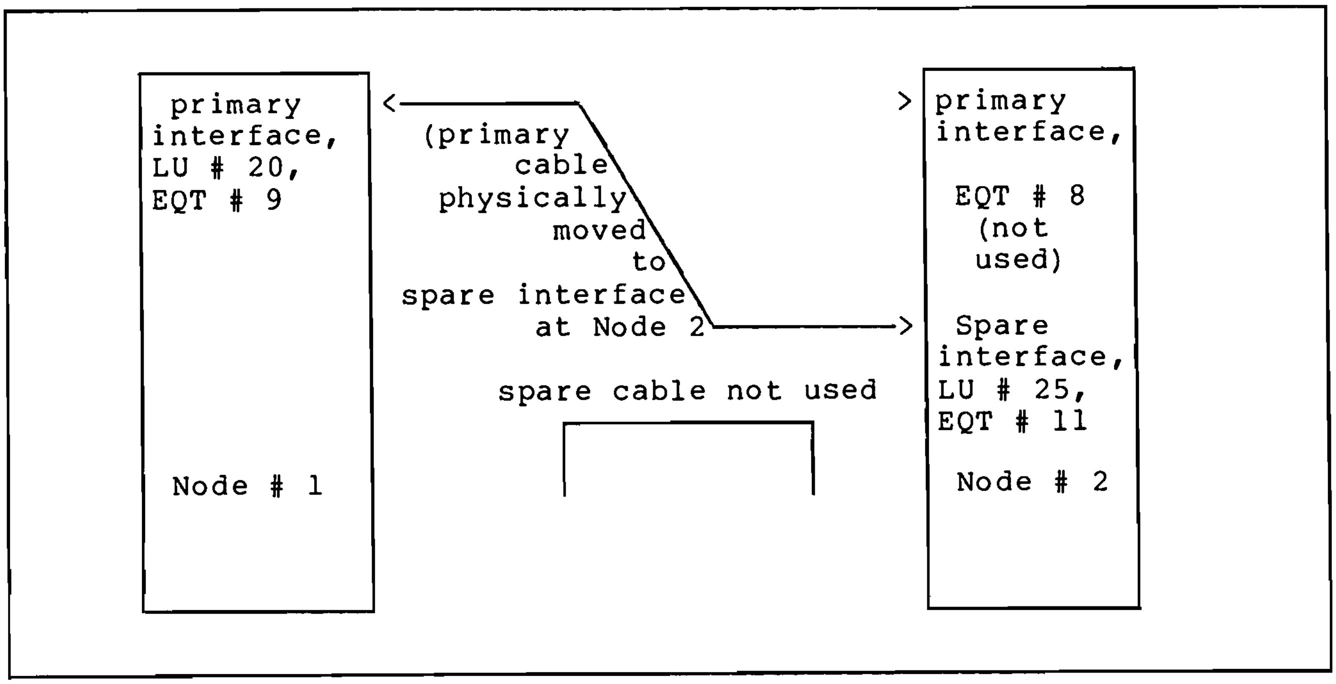


Figure 2-5. Spare Interface Link Configuration

Repeat the procedure, switching LUs and using REMAT to send messages across the line. If this works, then the interface in Node 2 corresponding to EQT # 8 is bad. If the problem persists, disconnect the primary cable from both nodes and substitute the spare cable. Re-enable the LUs on both sides, and try again. If this works, the primary link cable is bad. If the problem persists, check Table 2-3 carefully. Don't assume anything. Check every possibility.

Table 2-3. DS/1000 Communication Failure Checklist

- Was the special DS/1000 Communications Firmware installed correctly and operating properly? If the node supports communications with other RTE nodes via HP 12771/12773 links, this possibility can be eliminated.
- Are all the cables connected together securely, and firmly connected to the interfaces?
- Are the interfaces on both sides jumpered for the same speed and mode?
- Are the subchannels for the LUs on both sides correct?
- Are the Time-outs on both sides the same?
- Are the RTEs on both sides generated properly? If communications over this link have worked before, then generation-related errors are unlikely unless one or both systems have been regenerated (except provision of insufficient resources---SAM, RN, class numbers). Otherwise, check the examples in this manual carefully.

If you cannot find anything wrong and you've checked all of the above carefully, it is possible that two simultaneous failures exist. Either two interfaces in one system, one on each side, or both cables are bad. If this appears to be the case, you should review the operating procedures with great care. The odds against two such failures at the same time are extremely high, unless the procedures regularly used contribute to an extraordinarily high failure rate (e.g., rough handling of equipment, failure to correct a previous problem and returning the board or cable to operational status, high temperature, humidity, vibration or abuse of the communications cable, etc.).

To determine whether two failures exist, repeat the procedures above for all other connection combinations not tried.

The procedures described so far involve simple substitution and can be tried quickly. Other troubleshooting techniques exist, utilizing oscilloscopes and analyzers, which are beyond the scope of this manual. Your service agreement with Hewlett-Packard will describe procedures for diagnosis, repair and replacement.

Bypassing a Bad Link Without Spare Links

With the rerouting feature generated in the following procedure is automatic; without rerouting, use the following procedure.

If redundant links do not exist, you will need to re-adjust the Network Routing Vector (NRV) in much the same way as described in the previous section. The main difference is that you are not eliminating a node from the network, since only a link has failed, but merely changing the routing paths. There are many possibilities for configuring a network, but one of the following general patterns can usually be applied to the affected part of the network.

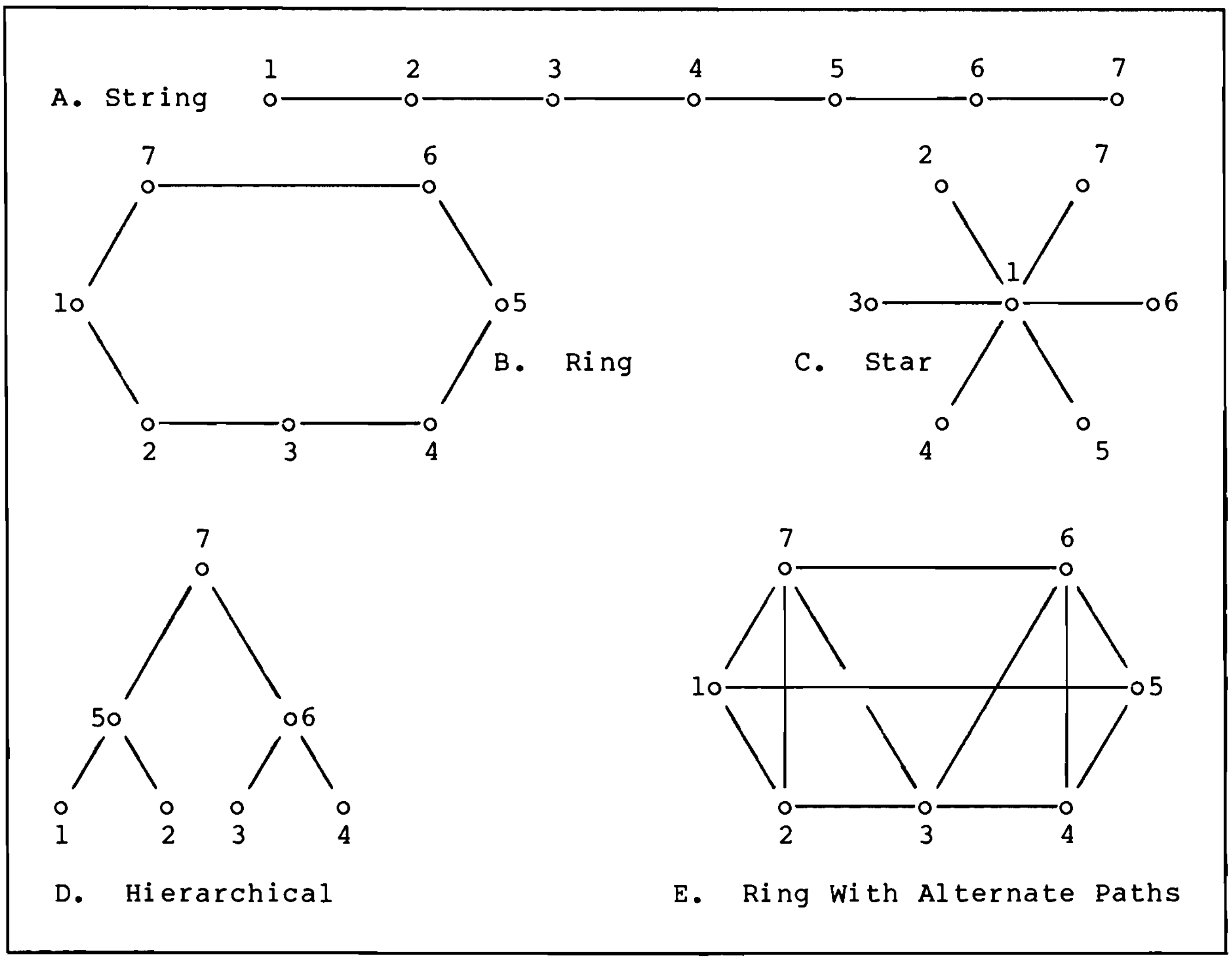


Figure 2-6. Network Configurations

In Figure 2-6 A, a failure anywhere except the end nodes effectively cuts the network into two separate, disconnected networks until the failure is repaired.

In Figure 2-6 B, any single failure can be compensated for by re-adjusting the NRV's to all nodes to avoid the bad link.

Without redundant links in Figure 2-6 C, there is no way to compensate for any failure of node 1, the center.

The hierarchical configuration in Figure 2-6 D, like the "star", is vulnerable to failures but does have the ability to allow some portions of the network to continue to function unlike the star configuration. No NRV reconfiguration can compensate for any failure, unless redundant links are present.

In Figure 2-6 E, suppose the link between Nodes 6 and 7 fails. The rerouting feature will do this automatically. If the automatic rerouting feature is not installed, one simply adjusts the NRVs so that these two nodes communicate with each other via node 3. Re-booting of Nodes 6 and 7 is required.

DSTEST

A procedure for testing the HP 3000-side communications hardware, called DSTEST, is described in the HP 3000 Distributed System Documentation. On the 1000 side, the slave program DSTES must be loaded and available either as a type 6 file or a temporary load residing in the system tracks.

Chapter 3
Remote I/O Mapping

Concept

Remote I/O Mapping is used to "map" or redirect I/O requests destined for an LU on an HP 1000 system to an LU on a different HP 1000 system in the DS/1000-IV network. It can be viewed as automatically converting interactive programs to remote use in a manner requiring no program modification. It was developed to accommodate terminal-less nodes in the DS/1000-IV network but can also be used for other applications subject to certain restrictions.

For example, assume we are at node 1 and a mag tape exists at LU 8 on node 3. A "map" can be established at node 1 which will, transparent to the user, cause all output sent to, say, LU 7 at node 1 to be re-directed and output on LU 8 at node 3 (see Figure 3-1).

```
 ----------------------------------------------------------------
|                                                                |
|          NODE 1          NODE 2          NODE 3                |
| EXEC or                                                        |
|  REIO    ------          ------          ------                |
|  call   |  LU  |        |      |        |  LU  |               |
| output  |      |        |      |        |      |    output     |
|   to  ----> 7 -----------------------------> 8 ----> appears   |
|  LU 7   |      |        |      |        |      |  on mag tape  |
|         |      |        |      |        |      |    at LU 8    |
|          ------          ------          ------                |
|                                                                |
 ----------------------------------------------------------------
```

Figure 3-1. Output to LU 7 at Node 1 is sent to LU 8 at Node 3.

Thus, we can run programs specifying our "mapped" LU (LU 7) as the output LU where, without user program modification, the output is re-directed to be output on some LU at a different node. The intervening node (node 2) is shown only to emphasize that any number of nodes may exist between the two nodes involved, including zero.

As another example, assume that we have a terminal-less node (node 1) in a network as shown in Figure 3-2. It is important to have LU 1 available as an interactive output device for error logging, etc. In our example, this is accomplished via Remote I/O Mapping by causing I/O requests destined for LU 1 to be re-directed to LU 10 on node 3. All error messages and other output sent to LU 1 on node 1 will instead appear on LU 10 at node 3. In a similar manner, any read requests made to LU 1 will be re-directed to LU 10 at node 3. The prompt (if any) will be displayed and the read performed (see Figure 3-2).

```
 -----------------------------------------------------------------
|                                                                 |
|                NODE 1       NODE 2        NODE 3                |
| program                                                         |
| or system     ------       ------        -------                |
|  prompt      |  LU  |     |      |      |   LU  |               |
| and read     |      |     |      |      |       |               |
| request -----> 1 ----------------------> 10 ---------           |
|           ---------------------------------   |       |         |
| response  | |      |     |      |      |    | |  read request   |
| received <  |      |     |      |      |    | |    appears,     |
|              ------       ------        ----|--   user types    |
|                                             |     response     |
|                                             |         |        |
|                                             |         |        |
|                                              ----------        |
|                                                                 |
 -----------------------------------------------------------------
```

Figure 3-2. Read Request on LU 1 Mapped to Node 3 LU 10.

In general, whenever an I/O request is made to a "mapped" LU on one system, an I/O request essentially identical to the request made on the source system is made to the destination LU on the destination system as specified by the "map". When the request on the destination system completes, status, driver type, transmission log, and any data read from the destination request are returned in a completion of the source request. Thus the I/O request on the source system is effectively mapped into a request on the destination system.

It is important to remember that this mapping is "one-way" in the sense that all transactions are initiated at the source node (the node where the "mapped" LU resides). That is, writes can be made to LU 1 (appearing on LU 10 at node 3) and read requests can be made to LU 1 (appearing and being satisfied at LU 10 at node 3), but both of these requests are invoked at the source node. It is still possible for LU 10 to function as a "regular" terminal at node 3. Output may appear from node 1 and input may be requested but nothing will be sent to node 1 from node 3 unless explicitly requested by node 1 via a read request.

Uses

Terminal-less nodes can use Remote I/O Mapping to send system console output to a terminal on another node. What's more, this terminal may receive output from and provide input to a number of terminal-less nodes, allowing it to act as the central control for a network of terminal-less nodes.

Resource sharing is facilitated by Remote I/O Mapping. Line printers, mag tape drives, and other unit-record devices can be shared with greater ease.

Output such as error logging from applications programs can be directed to a hard-copy device or perhaps a tape drive at a remote node without program modifications.

Remote I/O Mapping allows you to establish a remote interactive session at a remote node where, with a few differences, the terminal you are at will seem as though it were connected directly to the remote computer. This remote interactive session access will let you run and interact with programs on that node.

The procedures for setting up these uses will be discussed in the next section.

NOTE: Only I/O requests are transmitted to the remote node. Any access to system tables will receive local values and any non-I/O calls will be executed locally.

Mapping I/O To Remote Nodes

General Considerations

Equipment Tables (EQT's) play the key role in Remote I/O Mapping. A (system) LU is defined to be "mappable" if it points to an <EQT, subchannel> pair (or DVT on RTE-L, XL and A) which has been associated with the Remote I/O Mapping driver DVV00 at system generation.

Since we can point LUs to different <EQT, subchannels> (or DVT's) by using the LU command, we can map just about any unit-record type LU we want subject to Remote I/O Mapping and system "LU" command restrictions.

There may be several subchannels (and therefore several maps) associated with a mappable EQT. Refer to the "Network Manager's Manual, Vol. I," Chapter 2, for information on establishing mappable EQT's at gen time.

Throughout the rest of this chapter, we will refer to <EQT, subchannel> pairs, but it should be remembered that this is equivalent to a DVT in RTE-L, XL and A.

There are three ways to have an LU point to a mappable <EQT, subchannel>:

1. Associate the system LU with the mappable <EQT, subchannel> at gen time (this is the typical form, along with using the session "SL" command to point a session LU to this system LU).

2. Use the system "LU" command to point the system LU at the mappable <EQT, subchannel>.

3. On session systems the "SL" command may be used to map a session LU to a system LU which already points to a mappable <EQT, subchannel>.

Figure 3-3 shows some of the different possibilities of map set-ups. These are described below:

1. Several LUs may point to the same mappable <EQT, subchannel>. This is as one would expect, as RTE allows multiple LUs to point to the same <EQT, subchannel>. See (1) in Figure 3-3.

2. Several mappable <EQT, subchannel>'s may point to the same destination LU and node. The source LUs need not be at the same node. This may result in interleaved output at the destination LU, but there is an option to identify where each line of output is coming from as it appears on the screen (refer to IOMAP parameters). See (2) in Figure 3-3.

   NOTE

   The destination LU can not itself be mappable. This prevents <EQT, subchannel>'s from pointing back to themselves through other mappable <EQT, subchannel>'s, which would cause infinite loops of I/O map requests.

3. A mappable <EQT, subchannel> may point "nowhere" (the bit bucket). A request to an LU pointing to this <EQT, subchannel> will be completed with a transmission log of 0 and a status of 0. DVT5 at the RTE-L is such an example. See (3) in Figure 3-3.

4. A mappable <EQT, subchannel> may point to an LU on the same node. The destination LU can not be mappable. See (4) in Figure 3-3.

5. A mappable <EQT, subchannel> may have no LU pointing to it. This can happen if the LU had been moved elsewhere via the system "LU" command after the map had been set up. See (5) in Figure 3-3.

If Remote I/O Mapping is set up interactively from day to day, it is advised you use care and perhaps develop some tools to manage this resource (refer to Special Considerations in this chapter).

```
+----------------------------------------------------------------------+
|       Session Source Node                            Destination Node |
| +----------------------------------+                  +---------+    |
| |                  mappable        |                  |   LUs   |    |
| |   LUs      <EQT, subchannel>     |                  |         |    |
| |    1                             |        +---------|--> 1    |    |
| |    2                             |        |         |    2    |    |
| |    3                             |        |         |    3    |    |
| |    4 -----------> EQT 3,sub 0 ---|---------+         |    4    |    |
| |    5 ------+ 1                   |                  |    5    |    |
| |    6       |---> EQT 3,sub 1 ---|-----------  2 |---|--> 6    |    |
| |    7 ------+                  5 |              |---+  |    7    |    |
| |  > 8             EQT 5,sub 0 ---|-----------+        |    8    |    |
| |  | 9 -----------> EQT 6,sub 0 -  |                  |    9    |    |
| |  | 10                          4| |                  |    10   |    |
| |  | 11                           | |                  |    11   |    |
| |  +----------------------------+ |        +---------|--> 12   |    |
| |                                  |        |         |    13   |    |
| +----------------------------------+        |         +---------+    |
|  A/L-Series Source Node                     |        Destination Node |
| +----------------------------------+        |         +---------+    |
| |   LUs       Mappable DVT's       |        |         |  LUs    |    |
| |                                  |        |         |         |    |
| |    1 -----------> DVT4 ----------|---------+         |    1    |    |
| |    2                             |                  |    2    |    |
| |    3                             |                  |    3    |    |
| |    4     3                       |            +-----|--> 4    |    |
| |    5 -----------> DVT5           |            |     |    5    |    |
| |    6 -----------> DVT6 ----------|------------+     +---------+    |
| |                                  |                                 |
| +----------------------------------+                                 |
+----------------------------------------------------------------------+
```

Figure 3-3. Some Remote I/O Mapping Possibilities.

Using IOMAP

IOMAP is the program executed at the source node which sets up a map to an LU at a destination node. DS/1000-IV must be initialized between both nodes before running IOMAP. IOMAP is not interactive. That is, parameters are passed via the schedule parameters and parameters are returned to the "father" program via PRTN. The File Manager as the father program places the return parameters into system global variables. IOMAP accepts up to four parameters and returns four parameters. These are described in subsequent sections.

Running the program IOMAP under REMAT allows you to see the return parameters (in octal). If you run IOMAP under FMGR, the return parameters in the P-globals can be displayed using the "DP" command:

```
:DP,1P,2P,3P,4P
```

There are six basic tasks that the program IOMAP performs:

* establish/change a map
* establish a map using first unused mappable LU
* disable map
* request information on a mapped LU
* obtain error values
* request mapping information

These tasks are summarized in Table 3-1, along with some examples, and are explained in detail in the following sections.

Table 3-1. IOMAP Parameter Summary With Examples.

Task	Parameter: 1	2	3	4
Initialize I/O mapping	mappable LU	-1	ignored	ignored
Establish/change a map	source LU	dest LU	dest node	sec code
Establish a map using first unused mappable LU	-1	dest LU	dest node	sec code
Disable map	source LU	0	ignored	sec code
Information on map	source LU	-1	ignored	ignored
Obtain error values	-2	ignored	ignored	ignored
Mapping information	source LU	-2	ignored	ignored

Examples:

RU,IOMAP,75,1,3,DS	Map LU 75 to point to LU 1 on node 3
RU,IOMAP,1,40001B,3,DS	Map LU 1 to point to LU 1 on node 3 with the prompt option
RU,IOMAP,-1,100006B,3,DS	Map first available (unused) mappable LU to point to LU 6 on node 3 with the header flag option
RU,IOMAP,75,0,,DS	Unmap LU 75 (point it to the bit bucket)
RU,IOMAP,-2	Return last mapping error code detected

Establish/Change a Map

To establish (or change) a map on a specific mappable LU, use the following format:

```
RU,IOMAP,source LU,dest LU, dest node, security code
```

where: source LU — System LU on source system pointing to a mappable <EQT,subchannel> or DVT for which the map is to be set up (i.e., this must be a "mappable LU").

destination LU	Destination system LU of map. Bit 15 set enables the header flag. Bit 14 set enables the prompt flag and allows long timeout. Bit 13 set in conjunction with bit 14 also set disables the prompt flag but allows a long timeout.
destination node	Destination node number of the map (the node of the destination LU).
security code	Network Management security code.

RETURN PARAMETERS:

R1 = source LU if map successfully established
= -1 if security code specified was incorrect
= -2 if the source LU specified is not a mappable LU
= -3 if Remote I/O Mapping is not set up correctly
= -4 if the destination LU was mappable
= -5 if the destination node number is not known to the local node
= -6 if the destination LU is invalid or if attempts to obtain information about the destination LU failed.
= -7 if LUMAP is not present and active
= -8 if LUQUE is not present and active

R2 = destination LU of map as specified in IOMAP's schedule parameters

R3 = destination node number of map as specified in IOMAP's schedule parameters.

R4 = if IOMAP did not establish the map, it returns the specified security code. If IOMAP successfully established the map, it returns one of the following values:
= 0 normal completion
= -1 if the destination LU has either an associated infinite device timeout or a timeout larger than the user requested master timeout
= -2 if the destination LU is associated with the bit bucket
= -3 if the destination LU is not a unit-record device

The source LU is the mappable LU. The destination LU (dest LU) is the LU to which requests are re-directed, and the destination node (dest node) is the node number at which the destination LU is located. The destination LU must be associated with a unit-record device (such as a terminal, printer, or mag tape unit). Notice that the Network

Management security code is not necessarily the same as the Network User's security code used in the REMAT SW command, although the Network Manager may wish to make them the same.

Setting bit 15 (header flag) when specifying the destination LU will cause a message of the form shown below to be output on the destination LU preceding each record output to that LU:

MESSAGE FROM NODE # nnnnn PRGM ppppp AT DAY dddd, hh :mm :ss

where:

nnnnn = node number of message origination

ppppp = name of program sending message (or "(SYS I/O)" if the I/O request is system or class or buffered)

ddddd hh mm ss = day and time the request was made at the source node.

Setting bit 14 (prompt flag) causes a prompt of the form (nnnnn) to be written to the destination LU preceding each read request mapped from other LUs. nnnnn is the source node of the read request. The prompt flag is shown in Figure 3-6 later in the chapter.

These flags are useful when several LUs on several nodes are mapped to the same LU on one node. The user can then distinguish between the input and output requests from different nodes.

A map is changed in the same way it is originally set up. Run IOMAP specifying the source LU and the new destination LU (with or without header and prompt flags) and node.

WARNING: There is no protection against re-assigning a map which is in use. A recommended practice is to clear the map after use, and to use the feature described below to obtain each map.

Establish a Map Using First Unused Mappable LU

This format is used when you want to use a mappable LU which isn't already mapped (for example, when sharing a "pool" of these LUs among various users). IOMAP will search through the mappable LUs until it finds one which is not mapped and then use that LU as the source LU. The format is:

```
:RU,IOMAP,-1,dest LU,dest node,security code
```

where:

destination LU — Destination system LU of map. Bit 15 set enables the header flag. Bit 14 set enables the prompt flag and allows a long timeout. Bit 13 set in conjunction with bit 14 also set disables the prompt flag but allows a long timeout.

destination node — Destination node number of the map (the node of the destination LU).

security code — Network Management security code.

RETURN PARAMETERS:

R1 = source LU used for the map if map successfully established
= -1 if security code specified was incorrect
= -2 if no mappable LUs are available
= -3 if Remote I/O Mapping is not set up correctly or if IOMAP has not previously been run
= -4 if the destination LU was mappable
= -5 if the destination node number is not known to the local node
= -6 if the destination LU is invalid or if attempts to obtain information about the destination LU failed.
= -7 if LUMAP is not present and active
= -8 if LUQUE is not present and active

R2 = destination LU of map as specified in IOMAP's schedule parameters

R3 = destination node number of map as specified in IOMAP's schedule parameters

R4 = If IOMAP did not establish the map, it returns the specified security code. If IOMAP successfully established the map, it returns one of the following values:

= 0 - normal completion

= -1 - if the destination LU has either an associated infinite device timeout or a timeout larger than the user requested master timeout

= -2 - if the destination LU is associated with the bit bucket

= -3 - if the destination LU is not a unit-record device

This format cannot be used the first time IOMAP is run after boot-up. You must explicitly specify a mappable LU when running IOMAP the first time so that the Remote I/O Mapping subsystem will be initialized. This is easily done by running IOMAP in the WELCOM file, perhaps establishing a map or requesting information on a specific mappable LU.

Disable Map

To disable a map already established, the format is:

```
:RU,IOMAP,source LU,0,,security code
```

where:

source LU — System LU of map to be disabled.

security code — Network Management security code.

RETURN PARAMETERS:

R1 = source LU specified if map successfully disabled
= -1 if security code specified was incorrect
= -2 if the source LU specified is not a "mappable" LU
= -3 if Remote I/O Mapping is not set up correctly
= -7 if LUMAP is not present and active
= -8 if LUQUE is not present and active

R2 = 0 (the destination LU is the bit bucket)

R3 = 0

R4 = value of security code as supplied in IOMAP's schedule parameters

This will now make the LU "available" and a candidate for selection when IOMAP is looking for an unused mappable LU (source LU = -1).

Request Information on a Mapped LU

IOMAP can be used to determine which LUs in a system are mappable or to determine what map, if any, has been set up for a specific mappable LU. The format is:

```
:RU,IOMAP,source LU,-1
```

where

source LU — The mappable LU for which map information is to be displayed. If a non-mappable LU is specified, map information will be displayed regarding the first mappable LU greater than the LU specified.

RETURN PARAMETERS:

R1 = source LU as specified in IOMAP's schedule parameters. If a non-mappable LU was specified, the first mappable LU greater than the LU specified will appear in R1.
= if source LU was not a mappable LU, then R1 is the first mappable LU defined, greater than the source LU.
= -2 if the LU specified is not a "mappable" LU and there are no mappable LUs greater than the one specified
= -3 if Remote I/O Mapping is not set up correctly

R2 = destination LU of map for the LU appearing in R1.

R3 = destination node number of map for the LU appearing in R1.

R4 = 0

The destination node and security code parameters are ignored. The return parameters will show the current map for that LU as described previously.

Obtaining Mapping Information on an LU

IOMAP can be used to obtain mapping-related information on specific LUs. The format is:

```
:RU,IOMAP,LU,-2
```

where:

LU = The LU for which mapping-related information is to be displayed.

RETURN PARAMETERS:

```
            R1 = LU specified in the I/O mapping parameters.
               = -3 if I/O mapping is not initialized correctly.
    If R1 is mappable then:
            R2 = destination node
            R3 = destination LU
    If R1 is not mappable then:
            R2 = -1
            R3 = -1 if LU is an invalid LU number
               = -2 if LU is associated with the bit bucket
               = negative of the timeout of LUs EQT otherwise
```

Obtain Error Values

Any time Remote I/O Mapping detects an error, the error code is stored in the doubleword #LMPE (at the source node) and the reporting node number is stored in #LMPE+2. When an error occurs or if no map is set up for a called mappable <EQT,subchannel>, the request is completed with a transmission log of zero and a status of zero.

To obtain the error values use the following format:

```
:RU,IOMAP,-2
```

RETURN PARAMETERS:

R1 = first word (two characters) of the ASCII four-character error code (e.g., "DS" of DS04). For serious errors, R1 contains a numeric error code, in which case R2 and R3 are meaningless.
R2 = second word (two characters) of the ASCII four-character error code.
R3 = number of node reporting the error.

The errors in #LMPE are usually four character ASCII error codes, but if certain serious problems are detected, numeric codes are set in #LMPE, in which case the ASCII codes in #LMPE+1 and #LMPE+2 are meaningless. Table 3-2 shows these numeric error codes and their meanings.

Table 3-2. #LMPE Numeric Error Codes.

Code	Meaning
0	No error
1	Reserved for future use.
2	Reserved for future use.
3	Reserved for future use.
4	DS/1000-IV quiescent resource number is corrupt.
5	Source node is quiesced.
6	Class number set up for LUMAP has been corrupted.
7	Reserved for future use.
8	Program LUMAP is not present in system.
9	Destination node number is not in the NRV.
10	DS/1000-IV is not initialized.
11	Error on class number allocation.
12	No class numbers are available.

Note that #LMPE is unchanged unless an error occurs. That is, it always contains the last error encountered by the Remote I/O Mapping driver.

#LMPE error values are intended for use by the System Manager for troubleshooting information if Remote I/O Mapping is not working properly.

Remote Interactive Session Access

REMAT provides remote session access, but it is not interactive. Remote I/O Mapping provides the means to gain access to the File Managers on systems other than the local one. This "virtual terminal" capability allows you to establish a session at a remote node and interact with the File Manager almost as if the terminal were connected directly to the remote node.

The procedure described below shows how to set up a map to gain access to the File Manager and a truly interactive session at a remote node.

Establishing the Map

The first step is to run IOMAP at the node at which you wish an interactive session, mapping an LU at that node to your terminal's LU at your node.

Assume we are at node 1 terminal LU 1 and we wish to establish an interactive session at node 3. Using REMAT, we switch to node 3 and run IOMAP, mapping an LU to our terminal LU on our node. Refer to Figure 3-4. Note that this map could already exist, perhaps being established at boot time in the WELCOM file.

```
 ----------------------------------------------------------------
|                                                                |
|          NODE 1        NODE 2        NODE 3                    |
|                                                                |
|          ------        ------        ------                    |
|         | LU   |      |      |      | LU   |                   |
|   you   |      |      |      |      |      |  you wish         |
|   are   | 1 <-|------|------|------|- 75  |  to logon         |
|   here  |      |      |      |      |      |  here             |
|         |      |      |      |      |      |                   |
|          ------        ------        ------                    |
|                                                                |
|                                                                |
| :RU,REMAT                    Run REMAT at node 1              |
|                                                                |
| $SW,3,,DS                    Switch to node 3 (Network User's |
|                              security code = DS)              |
|                                                                |
| #RW,IOMAP,75,40001B,1,NM  <----------------------             |
|     000113  040001B  000001  047115  00000       |            |
|        ^        ^        ^       ^                 |            |
|        |        |        |       |          Map LU 75 to LU 1 |
|        |        |        |       |          at node 1, setting|
|        |        |        |       |          the "prompt" flag |
|        |        |        |       |          (network management|
|        |        |        |       |          security code = NM)|
|        |        |        |       |                            |
|        |        |        |    Octal ASCII coded "NM"          |
|        |        |        |                                    |
|        |        |     destination node of map               |
|        |        |                                             |
|        |     destination LU (in octal)                       |
|        |     with prompt option specified                    |
|        |                                                      |
|     source "mappable" LU                                      |
|     (octal representation of 75)                              |
|                                                                |
|        NOTE:  Return parameters are octal representation       |
|               of schedule parameters.  Map has been            |
|               successfully set up.                             |
|                                                                |
| #EX                                                            |
| :                                                              |
|                                                                |
 ----------------------------------------------------------------
```

Figure 3-4. Establish Map at Remote Node Pointing to your Terminal LU at your Node.

Obtain System Logon Prompt

The next step involves gaining the remote node's attention. This is done with the program SYSAT, described below.

SYSAT - System Attention Program

The program SYSAT can be used to get system attention or to break a program on a remote node. This is useful primarily in the case that the remote node has no terminals of its own but uses Remote I/O Mapping to communicate and when logging onto remote sessions interactively. In gaining system attention on a particular LU, SYSAT has the effect of someone striking a key on a terminal to get a logon (or break mode) prompt.

The format to get system attention on a remote node is:

```
:RU,SYSAT,attention LU,remote node number
```

This will cause the remote node to give its breakmode prompt or the logon prompt on the attention LU. The attention LU on the remote node must be a mappable LU (pointing to a mappable <EQT,subchannel>). On an RTE-L, the system prompt will always be given. On any other RTE, the system prompt will be given if the LU is 1. Otherwise, the program associated with Remote I/O Mapping, if any (specified in the interrupt table), will be scheduled and passed the EQT address. Typically, the program scheduled will be the system prompt program (PRMPT). At least one map must have been set up at the specified node for this request to work. Also, IOMAP must be present and active at the remote node or SYSAT returns a "NO SUCH PROGRAM" error ("NO PROGRAM ID" if the remote node is an RTE-A/L).

The format to set the break flag of a program in a remote node is:

```
:RU,SYSAT,program name,remote node number
```

or, if the local node is an RTE-MIII:

```
:RU,SYSAT,pr,gm,nm,remote node number
```

This will set the break flag of the specified program.

The parameter syntax for SYSAT is summarized in Figure 3-5.

```
In RTE-IVB, RTE-L:

        RU,SYSAT,attention LU,remote node number
or      RU,SYSAT,program name,remote node number


In RTE-MIII:

        RU,SYSAT,attention LU,remote node number
or      RU,SYSAT,pr,gm,nm,remote node number


Note that, in RTE-MIII, the program name must be
divided into three groups of two characters,
separated by commas.
```

Figure 3-5. SYSAT Parameters.

Logon and Establish a Session

If you are running under a File Manager at the local node, then you will want to keep this File Manager "out of the way" while interacting with the remote system. One way to keep your local File Manager from interfering with your remote session is to simply "OF" it. The procedure for OF'ing your file manager, obtaining system attention via SYSAT, and logging on is illustrated in Figure 3-6.

Notice that the password, if any, must be entered along with the account name (on the same line) in response to the logon prompt. Any valid account name can be used. USER.GENERAL is shown as an example. Also, RS is entered to restart your local file manager.

```
:OF,FMG01
FMG01 ABORTED
<CR>
S=01 COMMAND ? RU,SYSAT,75,3
MESSAGE DELIVERED

SYSTEM   3 LOGON: (    3) USER.GENERAL/PASSWORD

SESSION 75 ON.  4:14 PM  MON.,  18 AUG., 1980
PREVIOUS TOTAL SESSION TIME:  05 HRS.,  04 MON.,  58 SEC.

:(    3)       {now enter commands as if your terminal
   .            were directly connected to node 1}
   .
   .
:(    3)EX,SP
 $END FMGR
FMG75 REMOVED

SESSION   75 OFF  4:14 PM  MON., 18  AUG., 1980
CONNECT TIME:            00 HRS.,  00 MIN.,  34 SEC.
CPU USAGE:               00 HRS.,  00 MIN.,  00 SEC.,  350 MS.
CUMULATIVE CONNECT TIME: 05 HRS.,  05 MIN.,  32 SEC.
END OF SESSION
<CR>
S=01 COMMAND ?RS
:
```

Figure 3-6. Logon to Remote Node (user input underlined).

Obtaining a Breakmode Prompt

While logged on to a remote node, there may be times where you want the system's attention (breakmode prompt). However, the breakmode prompt you obtain will be that of the local system. To get the remote node's attention, obtain the local node's breakmode prompt and run SYSAT, specifying the LU mapped to this node as the attention LU (or, if running a program, you can also specify the program name). SYSAT will cause the remote node's breakmode prompt to appear.

Special Considerations

This section addresses the restrictions and other special considerations associated with Remote I/O Mapping.

Double-Buffered Calls (Z-Bit)

The double-buffered Z-bit option used in EXEC reads and writes cannot be used when performing an EXEC read or write to a mapped LU. It is for this reason that communication to some HPIB devices via a mappable LU is not supported.

Double-buffered reads from an RTE-L will always be translated into an interactive write/read.

Data Buffers Must Not Exceed 512 Words

Data buffers specified in I/O requests may not exceed 512 words. This should not be a problem since Remote I/O Mapping is intended for use with unit record devices.

Timeouts

When using Remote I/O Mapping to logon to a remote node, timeouts require special consideration. After logging on to a remote node, the remote File Manager issues a read request to the (remote) mapped LU, which is re-directed to the local terminal LU via a DEXEC call. This DEXEC read request is subject to the DS master request timeout value set at the remote node (default is 45 seconds). If you do not respond to the File Manager request within this master timeout, the following will happen:

remote node (logged on to):

- A zero length record completion will be recorded, causing the File Manager to write to the mapped LU "WAITING FOR INPUT," and then send another read request. These are turned into DEXEC calls, mapped to the destination node, and queued behind the original read request still pending on the terminal.

local node (where terminal is physically attached):

- While the read request times out at the remote node, it remains pending on the terminal you are at.

When you finally respond to the read request, the line that was typed is "thrown away" because that DEXEC reply is no longer expected at the remote node.

The remote File Manager message "WAITING FOR INPUT" appears on your terminal along with another prompt just as if you typed carriage return and nothing else.

If enough time elapses, several of these DS master timeouts may occur, resulting in several of the File Manager's read requests completing with zero length records. If enough read requests timeout in this way while under the File Manager, you will be logged off of the remote session.

Specifying the "prompt" flag (bit 14 on the destination LU) when establishing a map with IOMAP will cause the File Manager's DEXEC read requests to be performed with the interactive write/read option, allowing the request 20 minutes to complete. The DS master timeout (no prompt flag) and the interactive write/read timeout (prompt flag) apply to all read requests made to a mapped LU, not just the File Manager's. Bear in mind, also, that your local terminal timeout ("GOING, GOING, GONE") will function in the way you would normally expect it to, even though the session is a remote session.

When running a screen mode editor such as EDIT/1000, you should set bits 13 and 14 to obtain a long timeout but not a long prompt. This will prevent the prompt flag option from getting in the way of screen edits.

LUs Mapped To Terminals Use EQT Subchannel 0

Although any subchannel on a mappable EQT can be used by an LU, it is strongly recommended that each terminal LU (or interactive device) use EQT subchannel zero. This is because many programs check the subchannel of the LU that they are passed or call the routine IFTTY. A non-zero subchannel is interpreted by these programs to mean the device is not interactive. This does not apply to RTE-XL since it does not use subchannels.

Also, system prompts always go to the EQT without a subchannel specification and thus to subchannel zero -- there is no distinction made between subchannels here. Running SYSAT and specifying an LU will cause the prompt to be sent to whatever device is associated with subchannel zero of the EQT, regardless of the subchannel the specified LU was associated with.

LUs Mapped To Terminals Should Be Less Than 100

When logging on to a remote session via a mapped LU, the mapped LU must be less than 100. This is a session monitor restriction and is meant to prevent duplication of cloned program names.

EXEC 13 (I/O Status)

EXEC 13 requests obtain information (status and device type) about the device associated with a specified LU. The EXEC 13 can be performed on the local (mapped) LU and the information obtained since the Remote I/O Mapping software returns the appropriate information to the EQT at the node of the mapped LU. Since several different devices (LUs) may be mapped through the same EQT by specifying different subchannels, be aware that performing an EXEC 13 will give you the status word (EQT word 5) pertaining to the most recent operation. This means that if an EQT is used to map two LUs and you request status of one of these LUs, you may get the status of the other LU if it was the last one to perform an operation through that EQT.

Note that EQT word 4 is not provided. Any value returned for EQT word 4 is meaningless.

HP-IB and Disc LUs Not Mappable

Mappable LUs cannot be pointed to LUs which are connected to some HPIB devices or any disc LUs. Communication to some HPIB devices is through double-buffered (Z-bit) EXEC calls. The disc is not a unit record device and could easily exceed the 512 word buffer size limit.

Sharing Devices

LUs at remote nodes are not locked when data is mapped to them from local nodes. Care must be taken to ensure that the device use is coordinated between all users to avoid such things as interleaved output on remote lineprinters.

Error Processing

Programs performing input, output, and control requests to a mapped LU will have driver status returned in the A-register and the transmission log returned in the B-register upon successful completion. Unsuccessful completions may result from several causes:

1. A DS error. The A-register will return with bit 5 set (timeout) and the B-register will be set to zero. The actual DS error will be placed in #LMPE (accessible through IOMAP).

2. An EXEC error. If the EXEC error is detected at the local system (e.g., I001 - not enough parameters) the request is handled just like any other EXEC request with that error.

   If the error is detected at the remote node, the A-register on return will have bit 5 set, and the B-register will be zero. #LMPE will contain the error code. An error encountered at the remote node will cause return to the "good" return point when the no-abort bit is set.

3. A driver error (at the remote system). If the driver detects a condition which causes the remote system to print a message on its system console (driver caused EXEC errors of the form IONR, IOPE, IOTO, IOIT), then upon return to the program:

   - the A-register will have bit 5 set
   - the B-register (transmission log) will be zero
   - #LMPE will contain I007

   If no message is printed at the remote system console, the driver status is returned in the A-register as a normal completion and the transmission log will be set as in a normal completion.

Remote I/O Mapping Reserved LU

A "reserved LU" is specified at system generation time for Remote I/O Mapping. This LU is intended for use by the Remote I/O Mapping software, and is not to be used for user-established maps or other purposes.

Accessing The HP 3000

Note that when using Remote I/O Mapping to establish a remote session, you have access to all the programs at that session. This includes the program RMOTE when the node you are logged on to is a neighbor of an HP 3000 and has the DS link to the HP 3000.

Thus, you can run RMOTE, establish a 3000/MPE session, and have a "virtual terminal" connection to the HP 3000 from any terminal in the network via Remote I/O Mapping. However, you must remember that you are actually logged on to the HP 1000 that is the neighbor node to the HP 3000. This means that file transfers can be performed only between the HP 3000 and its neighboring HP 1000 node. To transfer a file from your local node to the HP 3000 (or vice versa), first use REMAT to transfer the file to the HP 3000's neighboring HP 1000. Then use Remote I/O Mapping to logon to the remote HP 1000, run RMOTE and perform the file transfer to the HP 3000.

Remote I/O Mapping cannot be used to map an LU on an HP 1000 to an HP 3000.

NOTE:

HP 3000 applications that reset terminal straps or make use of terminal block-mode capabilities may not function properly when run from RMOTE due to differences between the HP 3000 and HP 1000 I/O systems, drivers and interface cards.

Chapter 4
Remote VCP (DSVCP) and Forced Cold Loads

This section describes the operation of the DS/1000-IV A/L-Series VCP (virtual control panel) software, which allows the VCP of the 1000 A- or L-Series to communicate via a DS (distributed systems) link to a remote (master) computer, and thus allow a user at a remote site to "talk" to the VCP (e.g. access memory, instruct it to load programs, etc.). This software is designed to communicate with the A/L-Series VCP using the DS/1000-IV link level software and drivers. The master program also allows the user to run the DDL (Diagnostic Design Language) diagnostics over the DS link. Remote VCP operates only between adjacent nodes. DSVCP is also used for Forced Cold Loads to E- and F-Series machines.

Throughout this section, the word "master" will refer to the computer in control, which can be an M-, E-, F-, A- or L-Series computer. The "slave" refers to the computer that is being controlled remotely. This can be an E-, F-, A- or L-Series CPU, but the only command that can be issued to the E- or F-Series is the "\BREAK" function of DSVCP. This command to an E- or F-Series will cause an automatic reboot providing the CPU switches are set up for Remote Program Load. Due to hardware differences the M-Series computer cannot be controlled by the remote VCP capability.

The DS/1000-IV Remote Virtual Control Panel feature is implemented using two programs. The first is the VCP monitor, VCPMN. This program intercepts and displays A/L-Series VCP messages on the system console of the neighbor node considered the master of the A/L-Series Computer and also the node where VCPMN resides.

The second is the actual VCP master program, DSVCP, which allows the user to access the control panel of the slave computer, and consequently (in the A/L-Series) access and alter its registers (both CPU and I/O) and its memory. This program can access the boot loader programs in the VCP and cause various programs to be downloaded, via the DS link, or another A/L-Series loading device. The master program allows the master computer to do a memory dump from the slave A/L-Series to the master computer, and write the dump into a new file at the user's discretion.

Both the master program, DSVCP, and the monitor program, VCPMN, reside at the master computer.

NOTE

DSVCP does not work over HP 1000 Data Link. See FCL7 description in Data Link Chapter in Volume I, Network Managers Manual.

Required Hardware and Software

The following hardware and software are required to utilize the Remote VCP feature of DS/1000-IV:

1. HP 1000 A/L-Series Computer (slave) with a DS/1000-IV HDLC Interface Card installed. The HDLC card should be configured with select code 24 octal and BREAK feature enabled (U1S1 (switch 1) in the closed position. Remember only one card can have the Break flag enabled. Switch 5 on the CPU must be "UP" to cause VCP to use the HDLC card (Select Code 24) as the VCP channel. For automatic bootup off the neighbor node, the switch configuration on the CPU should be as follows:

Switch number	1	2	3	4	5	6	7	8
Position	open	open	closed	closed	*	DC	DC	DC

   *Open for L-Series computers, closed for A-Series computers.

   The file downloaded will be the first file encountered on a system disc with the name P00000.

   Consult the A/L-Series documentation for further information on these CPU switch settings. The slave computer may also be an E- or F-Series computer with HDLC card installed in any select code less than 37. The Forced Cold Load switch should be enabled on the HDLC card and RPL configured/enabled on the CPU board.

2. DS/1000-IV Master Node (either 1000 M/E/F/A/L-Series) with HDLC card installed, linked to the slave computer.

3. The required software is the VCP master program, DSVCP, the VCP slave monitor, VCPMN (only needed if slave is an A/L-Series), PROGL, and an initialized DS/1000-IV environment. VCPMN monitors the link to the A/L-Series waiting for front panel messages to arrive. When a message arrives, VCPMN displays it on the system console. The operator then runs DSVCP to respond to the A/L-Series front panel.

DSVCP Operation

The VCP master is started with the command string:

```
:RU,DSVCP,node number or -LU,[DU]
```

where:

NODE - is the positive node number and must appear as a neighbor node in the NRV, or negative LU number that has the slave A/L-Series (or E and F) connected on the other end. If the neighbor bit in the N is not set, DSMOD's CN command can be used to change this.

DU - This optional parameter is used when an immediate memory dump of the A/L-Series computer is desired. This version of the run string is used in response to the "NODE REQUESTING DUMP" prompt presented by VCPMN in response to an incoming VCP message. The dump request must be made by the A/L-Series, indicating it is expecting a dump request, in order to receive a valid dump. If you run DSVCP,node,DU when the A/L-Series is not soliciting the request, garbage records will be returned.

After scheduling DSVCP, it examines the Nodal Routing Vector to verify that the node is a neighbor to the master computer and that it is a legal node. If not, the master program responds with:

```
/DSVCP: ILLEGAL NODE

INPUT DS NODE (CR TERMINATES DSVCP)
```

and asks the user for the node again. If the user wishes to terminate the program at this point, a carriage return is sufficient. If the node is a rerouting node and has not been enabled, examination of the NRV entry will show 0 as the LU for access to this node. Use -LU instead of Node number in this case.

When DSVCP is successfully initiated the following message and DSVCP prompt will be displayed:

```
/DSVCP: NODE xxxxx LU yyyyy
/DSVCP:
```

indicating the node number and LU being used.

If the user inputs a negative LU number, the program tries to get the node number from the NRV. If the node number isn't found the program will still communicate with the LU itself, provided that it is the correct driver type 66 (DVA66). This search of the NRV is simply to obtain the node number for messages printed by the DSVCP.

During the time when DSVCP is controlling the line between the master computer and the slave computer, the link is taken out of DS mode and not available for normal DS traffic.

In normal operation, DSVCP prompts the user (/DSVCP:) for a command. DSVCP then determines if the command is to be sent to the A/L-Series, or if it is a command to be interpreted by DSVCP itself (i.e., if the first character is a backslash (\)) or if it should be sent directly to the A/L-Series (No backslash).

Normally, after issuing a command to the A/L-Series slave computer, DSVCP will enter a read mode waiting for some data to be returned. If the command specified a system reboot from the A/L-Series local disc (i.e. %BDC), no data will be returned. To return DSVCP to the prompt (/DSVCP:) mode the break bit must be set in the program's ID segment. In five seconds or less, DSVCP will return with a prompt. DSVCP should then be exited with a "\E" command to allow DS traffic to flow.

DSVCP Commands

There are four commands supported by DSVCP (\B, \E, \R, \W). They are described below. If a command is not recognized, the program responds:

```
/DSVCP: ILLEGAL COMMAND
```

An examples section can be found at the end of this chapter.

Break ("\B" or "\BREAK")

The BREAK command causes DSVCP to send a special message, called a "BREAK" frame, to the slave computer via the DS Driver. This halts execution of normal programs at the slave, and transfers control to the Virtual Control Panel at the slave. After the BREAK command is entered, DSVCP will respond with the P, A, B, M, and T registers at the time of the BREAK. VCP at the slave will consider any other break sent after the slave L-Series has gone into VCP mode as a PRESET (the same as VCP command "%P").

The Break command is the only command that can be used when the slave is an E or F series computer. When it is sent, the slave halts, and depending on the CPU switch settings, initiates a forced cold load. The Break feature is not available on M-Series computers.

On the first "BREAK" command the program will prompt:

PASSWORD?:

The user must type in the correct DS Network Management Security code (password), before the break frame will be sent. If the password is incorrect, a bad command is indicated.

Note:

1) This command halts the CPU, all real-time activity stops. This is why the password is required.

2) Activity is not halted until the correct password is entered.

End ("\E" or "\EXIT")

This command causes DSVCP to terminate, after returning the link to the DS mode. Using the "OF" command to terminate your session with DSVCP will leave the link in VCP mode, and thus, unusable by DS. Use (BR,DSVCP) to get a DSVCP prompt, then use the \E command to exit.

Wait ("\W" or "\WAIT")

This command causes the program to wait for input from the slave node. This command waits for either input from the HDLC card or a programmatic BREAK from the operator (BR,DSVCP). This command is useful when running the Diagnostic Design Language Interpreter (DDL) remotely via the DS link. When one diagnostic completes, another may be started and DSVCP given the \W command to wait for any completion data.

Read ("\R" or "\READ")

The \READ command is used to return DSVCP to the DS line read mode without sending anything to the slave. This is useful if the user has caused a break to the DSVCP master program, but later decides that no data should be sent or if DSVCP was aborted after someone sent a "BREAK" command. This command causes one read to take place on the DS LU. If the read times out (no message pending) the program requests input from the input console device, otherwise it prints the incoming message.

There are also cases where messages stack up on the HDLC card. One read request retrieves one message. Use the \R command to retrieve any additional messages and thus to resynchronize DSVCP with the slave computer.

Dump Routine

The DSVCP dump routine is used when VCPMN, the monitor program, has indicated that a dump is being requested by the slave A- or L-Series. This might be caused when a user at the A- or L-Series computer wishes to use the DS link to capture a memory image for later analysis (e.g. a "crash dump").

The user would break into front panel mode at the A- or L-Series locally, or via DSVCP remotely, and enter the command:

```
%WDS00xx
```

where "xx" is the select code (default is 24). This command results in the following message being printed by DSVCP at the neighbor node:

```
DUMP REQUEST, ENTER NEW FILENAME:
```

When the DUMP routine is initiated the program requests a filename from the user. The desired file is created (type 1 with 257 blocks) and the master starts sending a word count (128 words) to the slave A/L-Series. The slave A/L-Series responds with the number of words requested, which are written to the file. Each succeeding word count that is sent to the slave results in the next consecutive area in memory being read from the A/L-Series and stored in the file. This process continues 256 times (256 blocks * 128 words per block = 32K words of memory). (If a dump greater than 32K words is needed, use the following formula: number of words in memory divided by 128 equals file size in blocks (# words in memory / 128 = file size in blocks). The result of this formula should be used in the file size parameter of the namr typed in response to "DUMP REQUEST, ENTER NEW FILENAME:".) On the last record, the message:

    TYPE COMMENTS BELOW

is printed, giving you the opportunity to "label" the file. These comments are contained in the 257th block of the file as well as the time and date. This comment record is useful for documenting the conditions of the dump.

If an FMP error is encountered, a message:

    FMP ERROR -xxxxx ON DUMP FILE

and the program prompts for another file. If a (CR) only is entered the dump is aborted and an:

    !?

is printed. If a line timeout occurs, the message:

    LINE TIMEOUT - DUMP ABORTED

is printed. The next thing to do is enter the BREAK command and try again.

A/L-Series Register Access

To access the A/L-Series Registers, the user must first have issued the "\B" command then the user types the desired register name into the terminal, followed by a carriage return. The program will then respond with the contents of the register desired the same as with local front panel with a local terminal. All displays are in octal. If the user desires to change that register, the user should type the octal number desired into the terminal followed by a carriage return. DSVCP will respond with the register name and the new contents of that register.

It should be noted that the DSVCP will act exactly as if the user is communicating directly with an A/L-Series computer via a terminal. However, all commands must be terminated by a carriage return.

If the user desires to access a loader, the VCP loader command should be typed in the exact manner that is described in the VCP documentation when communicating with a terminal. This documentation is found in the apporiate computer reference manual.

Execution Commands (%R, %B, %E)

The DSVCP passes to the VCP slave the execution commands %R, %B, %E, %L or %W. After an execution command is given the program issues a "read" to the DS driver. If the read times out, DSVCP checks the "break bit" in its ID segment (IFBRK call). If the user indicates a break of DSVCP is desired, the master program again issues a prompt to the terminal for a command.

NOTE:

If the "break bit is not set, DSVCP may wait indefinitely for a VCP message to arrive. This will cause DS traffic between the master and the slave to also be blocked indefinitely. As a result, it is imperative that you break DSVCP and issue an "\E" command to exit. Do not abort DSVCP as it is the "\E" command that allows normal DS traffic to commence.

DDL Diagnostic Operation

For operating the DDL interface diagnostics via DS, the DDL program at the slave uses the VCP protocol when communicating over the DS link. DDL operates from the master terminal in the same manner as if the user were talking to the slave A/L-Series directly from a terminal. If the user desires to "BREAK" to the front panel, the command "\B" should be entered.

The following example of using DDL is provided to explain the remote operation of DDL:

Run DSVCP to L node 6.

```
:RU,DSVCP,6
/DSVCP:    NODE      6 AND LU 19
```

Send the break command to the A/L-Series.

```
/DSVCP:   \B
PASSWORD?:  DS      (Network Management Security Code)
```

When the break message is received from the A/L-Series, load DDL which, for this example, is assumed to be in file P00003 on the master node. In order to download a file, the file must have a file name P000XX where the X's are the file number in the range 0-17 octal. To initiate a download, only the file number can be passed to the DS Monitor PROGL which actually processes the download requests. PROGL must be a scheduled monitor.

The response should be "LC" which means Load Complete. If you don't get the LC indication, the troubleshooting chapter has several suggestions for checking out download problems.

```
P 011724 A 040000 B 177670 M 011723 T 103530 /DSVCP:   %LDS3
/DSVCP:
LC      (Load Complete message from VCP at the L-Series)
```

Now DDL is loaded from file P00003 and ready to be executed. The command %E will cause DDL to begin execution and prompt to the master node's system console.

```
/DSVCP: %E
```

Upon receiving the DDL prompt (along with the DSVCP prompt) you may begin entering a program.

```
DIAGNOSTIC DESIGN LANGUAGE
READY
>/DSVCP:   10 PRNT 20
>/DSVCP:   20 FMT ("THIS IS A TEST"/)
>/DSVCP:   100 STOP
```

List the program to check for errors.

```
>/DSVCP:   LI
   10 PRNT 20
   20 FMT  ("THIS IS A TEST"/)
  100 STOP
```

Now run the program.

```
>/DSVCP:   RU

THIS IS A TEST
LINE   100 STOP
```

Exit DSVCP.

```
>/DSVCP:   \E
/DSVCP: END
:  <------------- FMGR prompt resumes.
```

DSVCP Examples

Several examples are provided below showing the operation of DSVCP. Additional examples showing down-loading an L-Series, booting via the DS link and down-loading programs into a memory only L-Series are shown in the DS/1000-IV User's Manual.

When the DSVCP prompt is present the program is now indicating it is waiting for input. If the user desires to issue a "BREAK" to the slave CPU, the following is typed in.

```
\BREAK<cr> (or \B)
```

The slave L-Series responds with the contents of the registers as shown below:

```
P 012345 A 000000 B 077777 M 012344 T 002001
```

If the user desires to change the A-Register, an "A" would be typed only. The response would be:

```
A 000000 /DSVCP:
```

The user types the new octal contents, i.e., 125252 followed by a <CR>. The master responds with:

```
A 125252 /DSVCP:
```

Should the user wish to restart the program at the current location in the program counter, the following would be typed:

```
/DSVCP: %R<CR>
```

The program at the slave would then resume. Should the user want to cause a BREAK at the slave again, a key on the terminal should be struck in order to get system attention. The following should then be typed:

```
BR,DSVCP<CR>
```

The master program is now ready to accept a command or a character. If the user is executing DDL at the slave, the response initially looks like this:

```
DIAGNOSTIC DESIGN LANGUAGE
READY
>/DSVCP:
```

If the user desires to run a DDL program, the DDL command "RU<CR>" is typed after the prompt. The user can wait for the DDL program to respond, or the user can BREAK (BR,DSVCP) the master program in order to send a BREAK to the slave, enabling the slave VCP program.

Programmatic Forced Cold Loads

FCL66 is a utility to programmatically force cold load E-, F-, L-, or A-Series computers over a driver 66 HDLC link.

CALLING SEQUENCE

```
CALL FCL66 (NODE,IERR,SECUR[,TYPE[,FNUM]])
```

NODE — Node number or negative LU of the link to the system to be force cold loaded. If node number is specified, it must be described in the NRV as a neighbor and the direct link must be up. If the LU is specified, and if it is described in the NRV, it must be set in the NRV as a neighbor. For cases where rerouting has taken a node out of the NRV, the LU is still accepted if it is a driver 66 LU.

IERR — Returned error code (octal).

SECUR — DS network management security code to prevent inadvertent downloads.

TYPE — (Optional) CPU type of remote computer where:

```
          0 = M-, E- or F-Series
          1 = L-Series
          2 = A-Series

          The default is 0.

   FNUM   (Optional) L/A file number (octal). For L-Series, it
          must be between 0 and 17B. For A-Series, it must be
          between 0 and 77777B. The default is 0. (Not used
          for E- and F-Series, where the RPL descriptor block
          bit 0 determines whether P00000 or P00001 is used.)

 ERROR CODES IN OCTAL:

            -5 = RTE BREAK command was detected.
            -4 = Error on exit when clearing VCP
                 mode. (A and B registers contain
                 error return. However, if A is between
                 1 and 15 (inclusive), B is invalid and
                 A is returned from the driver.)
            -3 = Error return on Exec call. (A and B
                 registers contain error.)
            -2 = Parameter missing or error in
                 parameter.
            -1 = Illegal node number. Not a neighbor
                 nor a driver 66 LU.
             0 = No error.
             1 = Line failure.     \
             2 = Time-out.         |
             3 = Local busy.       |
             4 = Message aborted.  |
             5 = Remote busy.       > Returned from driver.
            10 = Not initialized.  |  For more information,
            11 = Wrong mode.       |  see the section DS DRIVER
            12 = Illegal request.  |  EQTs in Appendix C.
            13 = Card failure.     |
            17 = Bad interrupt.    /
```

NOTE

Even if IERR returns 0, the download may still fail. For example, if the P-file does not exist, or if PROGL is not in the system, the download will fail.

FCL66 verifies that the supplied node number or LU is legal and associated with driver 66. It then sets the link into front-panel-wait mode and sends a break to the remote computer. If the remote computer is an E- or F-Series, front panel wait is cleared and control is returned to the calling program. For L- and A-Series computers, FCL66 waits for a VCP message to come back from the remote computer and then sends a "%BDSnnnnn0024<cr><null>" command to the remote machine (nnnnn is the file number FNUM). Front panel wait is then cleared and control returned to the calling program. FCL66 waits for a period as long as the link's master time out for the VCP reply from the remote computer or for the link to come back up after VCP mode is entered.

To use FCL66, the switches on the remote interface must be set as described in the previous section, "Required Hardware and Software" in this chapter. Note that the HDLC card must be set to select code 24B for L- and A-Series computers.

Chapter 5
Internal Organization and Data Paths

Introduction

This section discusses the DS/1000-IV software organization. In the first section the functional relationships between major modules are discussed, aiding the software designer in determining the most efficient and reliable design possible. In subsequent sections, a more detailed description of each module's function and behavior is given, which the reader may find useful for troubleshooting purposes.

Data Flow Through DS/1000-IV, An Example

Refer to Figure 5-1 and let's follow a simple request through the system. We will discuss RTE-to-RTE communication paths in the first part of the chapter, then HP 1000-HP 3000 in the second part of the chapter.

A user program makes a "master request" subroutine call to one of the HP-supplied master subroutines. Some examples are:

```
DEXEC     DLGON     DLGNS
DMESS     DLGOF     DMESG     FCOPY
```

P-to-P routines:

```
POPEN, PREAD, PWRIT, PCONT, PCLOS
```

RFA routines:

```
DAPOS, DCLOS, DCONT, DCRET, DLOCF, DNAME, DOPEN, DPOSN,
DPURG, DREAD, DSTAT, DWIND, DWRIT
```

Their function is to build a "request buffer" containing the parameters necessary to describe the request. It is passed into the communications management software by calling subroutine "#MAST". Depending upon the request, data may be sent along with the request. The request buffers are shown in detail in Appendices A and B.

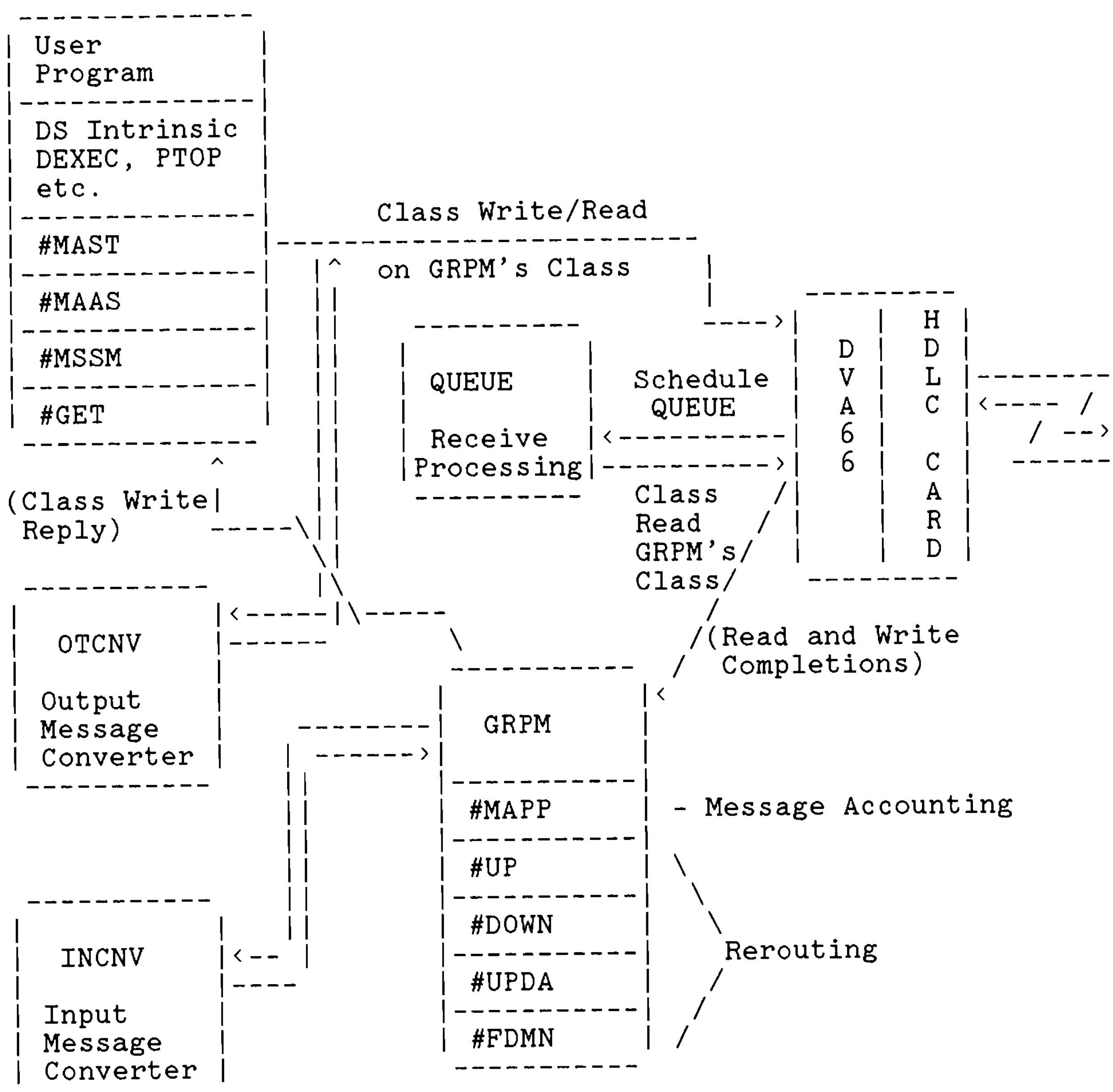


Figure 5-1. Internal Organization and Data Paths

```
                             ----------
                            |          |
       ---------  Schedule  |  QUEUE   |
      | H  |    |  QUEUE    |          |
      | D  |  D |---------->| Receive  |
      | L  |  V |<----------|Processing|
      | C  |  A |  Class      ----------
  --->|    |  6 |   Read
  ----| C  |  6 |    GRPM's Class
      | A  |    |                         Class
      | R  |    |\                     Write/Read  | Slave      |
      | D  |    | \                  GRPM's Class  | Monitor    |
      |    |    |<-----------------------------------|            |
       ---------   \     -------------------------->|   PTOPM    |
                    |   /          |    ^           |    or      |
                    V  /           V    |           |   EXECW    |
                --------         ---------          |   EXECM    |
               |         |      |         |         |   DLIST    |
               |  GRPM   |      |   RSM   |         |   OPERM    |
               |         |      |         |         |    etc     |
               |---------|      |         |         |            |
               | #MAPP   |      | Remote  |          ----------
               |---------|      |         |
               | #UP     |      | Session |
               |---------|      |         |
               | #DOWN   |      | Monitor |
               |---------|      |         |
               | #UPDA   |       ---------
               |---------|
               | #FDMN   |
                ---------
```

Figure 5-1 (Continued). Internal Organization and Data Paths

#MAST allocates a class number, to be used later for receiving the reply. (Note: sufficient class numbers must exist in the RTE system or else the master program will be suspended until one is free.)

#MAST then converts the nodal address to the appropriate DS LU by performing a table lookup on the Nodal Routing Vector (NRV). If the negative LU method of addressing is used, the NRV is searched for a neighbor node on this LU. If one is found the node number is used instead of the LU. If not found, a DS04 is returned to the caller.

Message Accounting Assigns Sequence Number

(The following processing is skipped if message accounting has not been generated into the local node or is not present at the destination node.) The message accounting routine #MAAS is called to assign a Message Accounting (MA) sequence number to the message. A sequence number will be assigned if the number of unacknowledged outstanding messages has not been exceeded.

Allocate a Transaction Control Block

A Transaction Control Block (TCB) entry is made allocated (taken from the free list of TCBs) and entered in the Master TCB list. The TCB is used to keep track of the request until a reply is received or the request times out. (More on this in the discussion of UPLIN below.) The number of TCBs is established by DINIT when the node is initialized. If none are available the master program will be suspended until one becomes free.

Assign Remote Session Identifiers To The Outgoing Message

At this point #MAST calls the Remote Session routine #MSSM to assign a Local and Remote Session ID to the message. (This processing skipped if there is no remote session anywhere in the network.) The remote session will not exist if:

1) There has been no prior logon via DLOGN.

   or

2) This program was not scheduled by a father with an existing remote session.

   or

3) This is this program's first message sent to the destination node.

   or

4) This program was not scheduled from a remote session.

When the first message goes out and comes back, a remote session will have been established and the session ID returned. This ID is saved in the Network Account Table (NAT) and is used when additional messages are generated to the same remote node. The NAT is contained in the module #MSSM appended to the user program. This table has room for 16 entries which enables a user program to have remote sessions active at up to 16 different remote nodes. (Only one remote session per node per program is allowed. Any number of programs may each have their own set of 16 sessions.)

When the message is sent out, MA waits for an "acknowledge" indication contained in message traffic returning from the destination node. MA will send up to 15 messages to a particular destination before requiring an acknowledgement. When the maximum number of messages have been sent and no acknowledgement messages returned, #MAAS will return a DS08(1) (node busy) error to any program trying to initiate additional traffic to that particular MA destination node. If all goes well #MAAS will update the MA tables in SAM and return to the calling routine, #MAST in this case.

#MAST checks the upgrade level of the destination node. If the destination node is a lower software upgrade level (i.e., DS/1000 91740) the message is sent not to the driver for transmission to the destination node, but to the message output converter, OTCNV. This is accomplished via a class write-read to LU 0, using OTCNV's class number instead of GRPM's class number. OTCNV converts the message "down" to the format expected by the destination node and sends the message off to the driver for transmission.

If no conversion is necessary, #MAST uses a Class I/O Write-read call specifying the transmission LU (zero if destined for local execution) to transmit the request. Two buffers are sent using the double-buffer feature of Class I/O. The data (if any) is in the first buffer and the DS/1000-IV request header buffer is in the second.

When the request is actually transmitted, either directly or after conversion, the General Request Preprocess Monitor's (GRPM) class number, #GRPM, is used. GRPM receives the status of the I/O transmission upon completion. If the destination CPU happens to be the same computer (local), then all processing can be done via Class I/O with no communication line activity involved. Refer to the RTE reference manuals for a description of Class I/O.

#MAST calls #GET to wait for the reply, or master time-out, utilizing a Class "Get" on the master requestor's class. Recall that the master class is allocated by #MAST for use with a particular message. The class number is stored in the TCB so when a reply arrives or a time-out occurs the reply data or error can be returned to the master via a class write-read. Note that when the reply arrives, the class number and the buffer in System Available Memory (SAM) will be deallocated and available for other uses.

Uplin and the Transaction Control Block

It is possible, at any given moment, for several programs in one computer to be waiting for a reply from another computer(s). There is no requirement that the replies arrive in the order the requests were sent. The TCB is the means by which the DS/1000-IV software can locate the proper receiver for each incoming reply.

In case a reply is not returned, the system protects the master caller from "infinite wait" by placing a timeout counter in the TCB. This counter is incremented by UPLIN, a program which runs every five seconds. If the counter goes to 377 octal before the reply arrives, UPLIN issues a class I/O write-read request on the master requestor's class number with a data length of zero. This is taken to indicate a timeout condition, DS Error DS05(0).

The TCB also contains a local sequence number (a unique number allocated to each newly-created TCB by #RSAX), which is used to solve the "abort" problem. That is, a program may be aborted by an impatient operator at any time. If a request is outstanding, a reply will eventually come back, which the system must be able to recognize and ignore.

It is also possible for this same program to be re-run, making a similar or even identical request before the first reply arrives. The local sequence number is incremented for each request, and is stored in the TCB and in the header of each request. The slave-side software echoes the last sequence number which, in incoming replies, is checked against the sequence number in the TCB. If a reply does not match any existing TCB it is ignored (the error message "TCB NOT FOUND, POSSIBLE TIMEOUT" is printed on the system console of the receiving side).

The TCB also contains the class number to be used for sending the reply back to the master and the master's I.D. segment address. See Appendix C for specific formats.

Thus, no matter what happens, system resources are never "lost". Class numbers and SAM buffers are recovered by whichever of the following occurs first: reply arrives, or if the master is kept waiting too long, a master time-out occurs, or the master program is aborted.

Class I/O Re-Threads

All transactions, except those which must be processed by the message converters, are sent from one processing stage to another by calling #RQUE to re-thread them onto another class, which links class buffer into the list of buffers waiting on another class. The buffer could be sent to the new class by issuing a class I/O write-read, but re-threading (changing the buffer linkage from one class list to another) is much more efficient. The software at each stage of processing would otherwise require maximum-size buffers, which, in the case of program-to-program communication, is 4096 words. Using the rethread method, the software modules require a buffer only large enough to contain the message header, excluding the data.

There is a top limit as to the number of requests queued onto a slave program's class. (This applies to a user's slave program in a PTOP mode as well as to a slave monitor such as EXECM.) This is to prevent a node from being "flooded" by requests (i.e. over-extensive usage of SAM for the class buffers) sent to only one program. The limit+1 is stored as a 1's complement of the value (default being -11, the actual queue limit defaults to 10) in #QLIM. When #RQUE detects the limit being exceeded while trying to add a new request onto the queue, it will reject the request resulting in a DS08(4) error at the originating node. This check is bypassed when the request is re-threaded to an EQT, such as in the case of a store-and-forward.

It is possible to override the default value of -11 in #QLIM. However, only the very sophisticated user should be concerned with that.

Failures on DVA65 Links Without Rerouting

If transmission of a request fails, GRPM analyzes the cause. If retries are possible, GRPM adds a delay interval (the value depending on the cause of the failure) to the current system time, sends the request to the system re-try module (RTRY) via a class I/O re-thread, and passes the absolute time to attempt the re-try in the class buffer header local appendage (see Appendix A). When RTRY receives the request (there may be others ahead of it), the absolute start-time for the retry is compared to the current system time. If the time has not yet arrived, it will suspend itself until then. (Of course, if the time has already passed, it does not wait). It then re-threads the class buffer to the output EQT on GRPM's class number. The RTE system will re-send the request as soon as the communication line is available. Because of the possibility of a wrap-around of the absolute time (only the lower 16 bits are passed) a maximum delay value is imposed; nominally this is 2 seconds. The user may change this by modifying #RDLY, which contains the delay value. (Nominally -200 for 2 seconds.)

If GRPM determines that all retries have been attempted, it returns either a DS01 or DS02 error, depending upon whether the last failure was a line parity (DS01) or line timeout (DS02). It sets a reply flag and returns the reply to the originating node. If this happens to be the local CPU, GRPM re-threads the class buffer to the master program's class number. If the message is a slave monitor reply, it is flushed and a message printed on the console.

The delay is necessary because any error which the driver's own retries cannot recover is probably caused by temporary conditions, such as a spurious line disturbance or a temporary lack of SAM in which to hold the buffer. These conditions will eventually disappear and the re-tries will eventually succeed, but re-trying before the conditions have had a chance to clear would squander them (the re-try count is fixed) and occupy the system needlessly.

Failures on DVA65/DVA66 Links With Rerouting

The rerouting software exists as a collection of subroutines appended to DINIT, DSMOD and GRPM. When rerouting is present in the system the GRPM error processing sequence changes. The transmission failure received by GRPM is processed differently depending on what type of link the error came from (DVA66/HDLC or DVA65/WASP).

When the error is from an HDLC link, the appropriate number of retries have already been attempted by the card. GRPM calls the appended rerouting modules to "down" the link and attempt to find a new path to the destination node. If all paths to the destination node are down a DS04(1) is returned to the user.

If a new path is found, the local NRV is updated to reflect the failed link and the new path. The rerouting program #SEND is scheduled to send out NRV update messages to all neighbor nodes. #SEND goes through the NRV looking for change flags in the NRV set by #DOWN (a rerouting routine appended to GRPM). When a changed entry is located, the node number and cost value are packaged and sent to all neighbor nodes. The neighbor nodes send any changes they make to their own NRVs on to their neighbor nodes and so on until the whole network has been updated with the new routing information. The algorithm converges rapidly because no new updates are sent upon reception of an update which does not cause a change to the receiving node's NRV.

For DVA65 links rerouting is supplied but when a failure occurs the failure is passed over to RTRY until the number of retries has been exceeded. At this time the DVA65 link will be declared down and the same rerouting processing described above takes place.

When an HDLC link is restored to service the driver gets an indication from the card that the link is functioning again. The driver passes this information on to GRPM. GRPM calls the rerouting "link up" processor to update the link vector and NRV, plus schedule #SEND to broadcast the change to all the neighbor nodes.

With DVA65 links there is no automatic "link up" indication. When the link has been restored, the DSMOD "/L" command must be used to reenable the link and generate the NRV update messages. At this point the operator must use the DSMOD "CN" command to restore the proper LU in the NRV. (This is for DVA65 links only.)

When HDLC links without rerouting have failures the message currently being transmitted is retried until the retry limit is reached then a failure returned to the driver. The link is marked down, but the failure is not returned immediatly to the master program. This is because the master request has already received a successful completion when the message was received by the card. This request will receive a DS05(0) timeout error after the master timeout period.

Any further traffic to this failed link will receive an error immediately (DS01, DS02, etc.), until the link is restored. When the HDLC link is restored, the HDLC card will be returned to service automatically and begin accepting messages. DVA65 cards require the DSMOD "/L" command.

Driver Processing

Driver 65 Processing

When a write request reaches the driver (DVA65), it first checks to see if the remote computer is sending a request. If so and the local node last transmitted, the driver rejects the request, sets the "local busy" flag in the EQT, and returns an error, otherwise the driver continues and the other side will reject its request. This results in a retry, with a delay of at least the "simultaneous request" timeout. Otherwise, the driver sends a Request Coming (RC) to signal the start of transmission and claim the line. The driver on the receiving end, upon detecting RC, will set its "request pending" bit and acknowledge receipt by sending Transmit Next Word (TNW). The sending driver then transmits the lengths of the data and request buffers, which the receiver echoes by way of acknowledgment. The receiving driver attempts to schedule QUEUE, passing it the incoming lengths. If QUEUE is busy, the driver sends a "STOP" control word, which causes the driver on the transmitting side to complete with an error indication, causing GRPM to delay the request for at least one second by sending it to RTRY.

When QUEUE is scheduled, it verifies the buffer lengths are within acceptable limits, and issues a Class I/O Read request on GRPM's class number using the lengths passed by the driver. If insufficient SAM exists, QUEUE will issue a "send STOP" control request, which the transmitting side treats as a Remote-busy.

Notice that the lack of SAM or inability to schedule QUEUE for long periods of time can cause the master program to receive the "remote busy" error (DS08). If, for example, QUEUE is disc-resident, it will be "busy" much longer than if it were made memory-resident. Since this module is so central to DS/1000-IV architecture, overall system performance will be severely degraded if this module is not generated into the system as a memory-resident program (or assigned its own partition).

If QUEUE can be scheduled and sufficient SAM exists to read the request and data buffers, the drivers on both ends perform a word-by-word transfer of the request and data buffers under microcode control. This feature avoids tying up a DMA channel during the transfer while still allowing more than two lines to be operated simultaneously. (Throughput is limited by line speed and the execution speed of the microcode data transfer instruction in the computer.) The microcode routines at both sides compute the error-check code words. When transmission is complete, the receiving microcode returns its block-check words to the transmitter, which verifies them against its own computation. If correct, the transmitter sends a TNW completing the transmission and returns to the software via a 10-ms time-out set in the EQT. If not correct, the microcode returns to the software part of the driver via a 10-ms timeout to handle the error. The transmitter then signals a retransmission by sending a Retransmit Last Message (RTLM). Up to seven retries are allowed at the driver level. If all retries fail, a "line error" condition is returned to GRPM by the drivers at both ends. At the receiving side, GRPM then prints the message, COMMUNICATIONS READ ERROR, via QCLM. At the transmitting side, GRPM will delay (via RTRY), and resubmit the message for retransmission.

DVA66 (HDLC) Processing

Driver 66 processing of the HDLC cards is quite different than Driver 65 processing for the WASP (Word-oriented Asyncronous Protocol) card.

When the message is queued to the driver from #MAST, DVA66 sends a message to the card to determine how much space is available on the card. The space on the card is managed in memory blocks called "frames". The frame size is user selectable and usually set to 1024 bytes for hardwired links and 128 bytes for modem links. When the card returns the amount of space available, the driver will break up the message into frames and send the indicated amount of frames to the card.

Appended to the front of the first frame are 2 words containing the request header length and data length. The interaction between card and driver is accomplished using OTA/LIA instructions when sending commands to and from the card, while the actual frame transfers occur using DCPC. The first 2 length words of a message are also OTA'ed to the card, the rest of the frame is a DCPC transfer. On the L-Series the complete transfer is via DMA.

Once the card receives one frame it can begin transmitting it to the interface at the other end of the link. If there is space on the card, additional frames of the message are transferred to the card. If the card is full, as the frame buffers become empty on the sending card, additional DCPC transfers can occur bringing additional portions of the message to the card. On hardwired links, where the frame size is 1024 bytes, the complete messages usually fit in one frame. A burst of traffic should go to the card at the rate of the DCPC channel, until the card memory is filled. Then the card will empty its buffers at the line rate.

The card has 7K bytes of transmit buffers and 7K bytes of receive buffers. At any one time, several messages may be contained entirely on the card.

At the other side of the link, once the first frame is received an interrupt is generated to notify the driver. The driver reads the first two words of the first frame to get the length words used to schedule QUEUE. The length data is then stored in the EQT and QUEUE is scheduled. QUEUE performs a class read back to the driver. If insufficient SAM exists to receive the message QUEUE will attempt to receive just the request header so that GRPM can return a DS08(0) remote busy indication to the origin node, otherwise QUEUE will issue a "SEND STOP" control request to the driver. This condition will eventually be treated as a "remote busy" condition by the sending side.

If all goes well, the message will be read from the card one frame at a time. The read takes place on GRPM's class number and the I/O completion is received by GRPM for processing when the complete message is transferred from the card.

Error Detection

With the 12771/DVA65 links, vertical and diagonal parity words are calculated by the microcode drivers at both ends. The hardware calculates the horizontal parity. The receiving driver sends its parity-check words to the driver on the transmitting side for verification. In conjunction with the horizontal-parity calculated by the hardware, this provides an error-check capability with an estimated probability of failing to catch an error, given that one occurred, of one chance in ten to the fourteenth power. Multiplying the probability that an error will be undetected, given that one occurred, by the probability of that error yields the probability of an undetected error. Many communication lines have error rates as low as one bit error per million bits and some lines (e.g., hardwire connections, leased lines, yielding an undetected error rate of one in ten to the twentieth bits transmitted.

On the microprocessor based cards, CRC-16 is used and is calculated for each frame by the Serial I/O Chip on the card. The actual check code is a 16 bit quantity and provides error detection of all bursts up to 16 bits in length. More than 99.99% of error bursts greater than 16 bits can be detected.

I/O Completion Processing

When the driver completes any operation (read or write), GRPM processes the status. If an error occurs, GRPM may retry, return an error code, or print a message depending on the circumstances. If this operation was a successful transmission, the class buffer is released and GRPM awaits the next I/O completion.

If the operation was an unsuccessful transmission, GRPM examines the cause of the error. In the case of DVA65, if all seven driver re-tries have failed because of transmission errors (block-check words did not verify), GRPM increments the line-error retry count. Four re-tries are allowed If a retry is allowed, GRPM calculates the time-of-day it wants the request re-sent and threads the buffer onto RTRY's I/O class. RTRY will re-send the request after an appropriate delay.

For DVA66 all retries are performed on the card. Only in the case of the "remote busy" error does GRPM route a DVA66 request to RTRY for additional processing. The number of retries is determined by the "REMOTE BUSY RETRIES" which can be altered with the DSMOD /T command. The default is 3.

When the rerouting option is genned in and the driver gets a link failure indication from the card the rerouting software appended to GRPM is activated as described earlier.

In the case of successful reception, the addressed node (destination node for requests, origination node for replies) is compared to the local node number. If not addressed to the local, the NRV is used to find the logical unit for transmission. A store-and-forward operation is initiated by threading the buffer on the transmission EQT using GRPM's class number. Note that the buffer did not have to be moved, only re-threaded, which saves time and CPU utilization. Later, when the transmission completes, GRPM will receive the status as a "write completion" and deal with it as described above.

If the message received was addressed to the local node, further processing depends on whether it is a request or a reply. If it is a request, a slave TCB is allocated and the class buffer is threaded onto the monitor's class number. If it is a reply, the master TCB is found by matching the transaction sequence number of the message (not the MA sequence number) with the TCB entry with the same sequence number, the class buffer is threaded onto the master's class number. GRPM then awaits the next I/O completion.

When Message Accounting is present in the system GRPM does additional processing on each I/O completion. On transmission the message accounting Post Processor Module (#MAPP) is called by GRPM to determine whether the message can be regenerated in case of failure. In master programs all messages can be regenerated. If the master is using the no-wait features of PTOP, no Message Accounting support is provided. In the slave monitor case or in the case of PTOP slave routines the message cannot be regenerated so the reply messages are requeued to a holding class (a queue in SAM) until an acknowledgement message arrives to indicate the reply was successfully received at its destination. At this time #MAPP deallocates the message from the hold class.

On transmit completions, messages that can be regenerated are deallocated from SAM.

On receive processing GRPM calls #MAPP to determine if the message is valid. The validity check involves checking to see if the message sequence number falls within a proper range or "window" of sequence numbers valid for this node. This window consists of 15 consecutive MA sequence numbers, beginning with the sequence number assigned to the message that has been waiting longest for an acknowledge. As acknowledgements come in the window of valid sequence numbers advances. The first received message may have been previously cancelled due to a time out at the origin node or routing changes that caused the cancel message to arrive prior to the actual message. The message may also have already been received in which case it is discarded.

The message may also be an MA internal message which is processed only by #MAPP and discarded. These include the message cancel requests from other MA nodes and no-response messages from the driver. The Message Accounting option is discussed in detail later in this chapter.

Slave-Side Processing

All slave monitors await requests by performing a Class Get request usually using the routine #GETR, which has expanded capabilities over #GET. When the request (and data, if any) is received, the buffers are moved from SAM into the monitor's local buffers and the class buffer (SAM area) is deallocated. For some monitors, e.g., PTOPM, where there is no reason to examine the data, the request header is received by the monitor and examined to determine what action is required. Generally in this case the message is simply requeued to a slave program's class. In any case, the class number is not deallocated. #GETR returns to the monitor which then performs the master's requested function. When the monitor has completed servicing the request, it builds a reply buffer, supplying the appropriate information and calls subroutine #SLAV to return the reply (and data, if any). #SLAV deallocates the slave TCB, converts the originator's nodal address to the appropriate output LU, and does a Class I/O write-read onto the output LU, using GRPM's class number. #SLAV then returns to the monitor which in turn calls #GET to await its next task, by suspending on a Class Get.

The handling of data transmission, store-and-forward, and error retries for replies is essentially the same as for the initial request handling by DVA66/DVA65/GRPM/RTRY. The fundamental difference is that the message is directed toward the originating nodal address rather than the destination nodal address. If an error occurs which is not retryable, or all retries have been exhausted, then the reply will be flushed. The node which does this prints an error "REPLY FLUSHED" on its system console. The master will receive a DS05 time-out error. If MA is generated in and active, the error qualification will be used to distinguish between requests and replies being lost.

When the reply (and data, if any) arrives at the originating node, GRPM locates the TCB for the master requestor and, finding the class number, rethreads the reply and data, if any, onto the master's class.

You will recall that we left #GET suspended on the master's class number which was allocated for this purpose. The master program resumes execution when the reply is threaded onto its class number. #GET checks that the reply (and data, if any) length is not larger than the allowable limit. If either one is larger, a DS03 error is returned. Otherwise, the data (if any) is moved into the caller's data buffer. The class number and TCB are deallocated. Control returns to the user's program along with any error information. In a DEXEC request, if the "no abort" bit is not set and an abortive error occurs, #TILT will be called to abort the master program.

If UPLIN has "timed out" the master request, it will write a zero-length buffer to the master's class number, which #MAST interprets and returns to the caller as a DS05 error. In any case, the class number and TCB are always deallocated when the master program receives control or is terminated.

A more detailed description of each of the communications management software modules follows.

DSN/X.25 Links

DSN/X.25 links use the DS customizing subroutine CSV66 attached to the DSN/X.25 driver DVX00.REL (for RTE-6/VM systems), or the DS customizing subroutine CXL66 attached to the DSN/X.25 driver DDX00.REL (for RTE-A systems)

A general description of DSN/X.25 is included in Chapter 1 of the the "DS/1000-IV Network Manager's Manual, Vol. I" (P/N 91750-90010). For a detailed description of DSN/X.25, refer to the the "DSN/X.25/1000 Advanced Guide" (P/N 91751-90002).

NOTE:

> If the DSN/X.25 connection consists of a Packet Switching Network (PSN) or a genuine modem link (not "hardwired" with baseband or short haul modems), circuits are NOT automatically reconnected if the network or modem(s) goes down and is later restarted. You must issue the line re-enable (/L) command in DSMOD to re-enable communication.

#MAST

#MAST performs the following functions:

1. Clears certain fields in the request header and enters the "maximum loop count" used to ensure messages cannot become caught in a "loop" due to manually entered incorrect routing information, or possible transient conditions during topology changes. This field is decremented by each node that processes the message. The message is flushed if the count becomes zero.

2. Returns DS00(1) to caller if the system is not initialized. Returns DS00(3) to caller if system is quiescent.

3. Allocates a class number for the request.

4. Locks the "RES" table access resource number, thereby guaranteeing that no other programs can modify the table. If there are no TCBs available, this RN is already locked, causing the calling program to be suspended until one is available.

5. Removes a TCB from the TCB pool, and links it into the master-request list, fills in the table entry information then clears the table-lock resource number (#RSAX performs this operation).

6. Converts destination node number to a logical unit number. If an LU is supplied, the NRV is searched and the LU converted to a nodal address of a neighbor node.

7. Assigns the message accounting sequence number to the message by calling #MAAS.

8. Assigns Remote and Local Session ID in request by calling #MSSM.

9. Checks destination node upgrade level to determine whether the request should be sent to the output converter OTCNV or directly to the driver.

10. Sends the request (with data, if any).

11. Calls #GET to await the reply (and data, if any).

12. When the reply arrives, if it was a "quiescent reject", puts calling program in the time-list, awaiting retry later, if the Remote Quiet Wait value is non-zero. Otherwise, an error is returned.

13. If the error flag in 7th reply word = 1, the master class number and TCB are de-allocated and an error return is made.

14. Deallocates the master class number and TCB. If the master program is aborted while waiting for a reply, the system will still receive the reply eventually, acknowledge it as usual, and link it to the TCB. The TCB will eventually be "timed out" by UPLIN. UPLIN will clear the TCB and release the class buffer and class number. If UPLIN detects that the program has aborted prior to the reply message arriving, UPLIN will deallocate the TCB. When the reply does arrive, if GRPM fails to find a matching TCB, it flushes the reply, and prints:

    TCB NOT FOUND, POSSIBLE TIME-OUT

    via QCLM.

15. Calls #MSSM to store returned Remote Session ID, if any, for use in additional requests from the master program.

16. If reply ok, gives return with received lengths in A and B registers.

#MAST error processing:

1. If sign bit (#15) of passed control word parameter is set, ASCII error codes are supplied to the caller in the A- and B- registers.

2. If the sign bit is not set, then the routine #TILT is called to print the error message, program name, reporting node number, and abort the calling program. #TILT is a subroutine internal to #MAST.

GRPM

GRPM is the general request pre-process module for RTE-to-RTE communications. It handles incoming messages (requests and replies), and outgoing transmissions.

Most of the rerouting processing is implemented by routines appended to and called by GRPM. In addition, a large portion of the message accounting (MA) feature is implemented by appending the MA Pre-and-Post Processor (#MAPP) to GRPM. The following paragraphs will explain how this works.

If either or both features are not required and the libraries which remove these features are generated in ($DSNRR and/or $DSNMA), calls to these routines are changed to "no-operation" instructions by routines contained in these "dummy" libraries.

Incoming Requests

GRPM awaits new input via a Class Get on its class number (allocated by DINIT when the node is initialized and kept in RES at entry point #GRPM). GRPM processes completion status of messages transmitted by #MAST, incoming messages received by QUEUE, and update messages for Rerouting and Message Accounting tables.

NRV update messages (Routing) and MA acknowledge requests are sent across the network on stream zero and are processed by the Rerouting and MA routines appended to GRPM.

An incoming request is rethreaded onto GRPM's class by QUEUE. If the request destination is not the local node, the line error retry counter in the stream word of the request is cleared, the hop count is incremented and the request (and data, if any) are re-threaded for output to the appropriate communication line logical unit on GRPM's class number.

If this is a new request to the local node, then:

a) The level flag is checked in the message to determine if message conversion is necessary. If needed, the message is rethreaded to the Input Message Converter's class (INCNV). If the message converters haven't been scheduled, an error is printed on the system console and the message is rejected with a DS07(1) error and sent back to the origin node. Replies are flushed. The message converters are scheduled automatically during DS initialization if any node in the NRV is level 0. When the conversion is complete, the message is written back on GRPM's class for further processing.

b) GRPM checks for stream zero messages (Message Accounting and Rerouting). If so, it is processed locally by the #MAPP or one of the Rerouting routines appended to GRPM.

c) If TCBs are not available, GRPM sets the reply and remote-busy flags in the stream-type word and returns the request to the originator by rethreading for output on GRPM's own class, queued to an EQT for transmission back to the sender.

d) If the system is going quiescent, or the required monitor is in unavailable-memory suspend (state 4), the "busy" flag is set in the request stream-type then the entire transaction is returned to the originator.

e) Otherwise, a slave TCB is allocated from the available pool.

f) GRPM checks the destination Session ID. If it's zero, no local session has been established for processing this request. Before dispatching the request to the appropriate monitor it is sent to the Remote Session Module (RSM). When a session is established, RSM will dispatch the request to the proper monitor. The monitor processing the request will execute under the session created by RSM.

If the incoming transaction is a reply and destined for the local node, GRPM checks the busy flag:

a) If it is set and the remote-busy retry count has not been exhausted, the flag is cleared and the buffer re-threaded onto RTRY's class number with a delay of one second. If the "busy" flag is set and no remote-busy retries remain, the master requestor receives a DS08 error.

b) If no error conditions exist, the list of master TCBs is searched to find the master program for this request. If found, the reply is re-threaded onto the master's class number. If the master TCB is not found the message in SAM is released, and a block is sent to QCLM indicating a communications error. The message:

   "TCB NOT FOUND, POSSIBLE TIMEOUT"

   will be printed on the local operator's console. GRPM then awaits the next incoming data block via a Class Get.

Outgoing Line Completions

Transmit completion processing is performed by GRPM simply to handle the error conditions that may occur and pass messages to the Transmission Retry Processor, RTRY, when necessary. When no errors exist, GRPM calls Message Accounting to perform post-processing on the transmitted message and then returns to the Class Get to wait for the next request.

The MA post-processing in the case of outbound slave reply messages consists of requeuing the message to a holding class. For requests, MA post-processing simply releases the SAM buffer.

GRPM processes completion status of all communication "write" operations (except system down-load messages which are handled by PROGL). If a write operation is successful, the message is deallocated from SAM. On remote-busy errors, GRPM checks the retry count in the stream word of the request. If all retries have been exhausted, the error is treated the same as a line error and a DS08 error code is returned. If another retry is possible, the absolute system time at which the retry should be attempted is computed and stored in the local appendage in the DS message header. The class buffer is then re-threaded onto RTRY's class.

For DVA65 links, parity or line timeout errors are retried up to four times by re-threading to RTRY'S class (the driver has issued 7 retries before completing with a parity-error status). If all 4 tries fail or a STOP RECEIVED (indicating some catastrophic error detected by the receiving driver) condition is detected, a DS01 or DS02 error is returned. All line errors have the error code and local node number stored in the error field in the request. If the message was a reply, the class buffer is simply cleared and a message is printed on the system console. If the message was a request and the originator is not the local node, the reply flag is set and the request is sent back to the originator. If the originator is local, the reply is re-threaded onto the master's class number.

On DVA66 (HDLC) links the retries are handled by the card. If a catastrophic failure occurs the card returns the failure status to the driver and "downs" the link. DVA66 will return the failure condition to GRPM. If this was a rerouting link, the rerouting module #DOWN will attempt to find an alternate route. If successful the NRV will be updated with the new LU and #SEND activated to send NRV update messages to all neighbor nodes.

If no alternate exists, an error is returned to the master program and the LU in the NRV for this destination node is set to zero. Future requests on this LU will be returned with DS04 errors.

Rerouting

As mentioned earlier, the rerouting feature consists of several subroutines appended to GRPM and routing tables in SAM which are used to dynamically determine optimum routing in the network at any given time.

The routing tables are the Cost Matrix, Link Vector and the Network Routing Vector (NRV).

Cost Matrix

The Cost Matrix is a 2-dimensional matrix holding the relative "cost" value to reach each destination over all possible rerouting links out of this node. Each entry in the table is 2 words. The first word is the "cost" value to get to a particular destination using a particular rerouting link. The cost value is summed to indicate the total cost to get to the destination; this is maintained by rerouting message traffic. The second word is the loop count (number of intervening nodes) to the destination node from this node. The entry point #CM in RES contains the base address of the Cost Matrix. The number of entries is equal to the number of Rerouting links in the node times the number of nodes.

Link Vector

The link vector is a table of 6-word entries, one entry per link. Each entry contains:

1) Link Status - up or down

2) LU of the link

3) System Time (2 words) - used to keep track of the number of times the link went down in a given 5-minute period.

4) Up/Down Counter - counts the number of link failures. When the number of temporary failures in a given 5-minute period exceeds the "max line down count" (alterable with the DSMOD /T command) the link is declared down and rerouting attempts to find other paths around the failed link.

5) Neighbor Node number - used by #SEND when sending NRV update messages to all neighbor nodes. Used by Rerouting to indicate the current best path (LU) to a given destination node.

These tables appear in Appendix C of this manual for a pictorial display of their format.

Rerouting Processing

As mentioned earlier, GRPM contains most of the software required to implement rerouting. GRPM contains a module to process rerouting link failures, link restorations, and a module to process routing update information from other nodes in the network.

Link Failure Procedure

When the driver completes any I/O operation to a DS/1000-IV communication link, GRPM processes the result. If the status is an unsuccessful transmission, GRPM examines the cause of the error and does necessary retries above the driver level. If the retries report a link/card failure, GRPM returns a DS01 or DS02 error to the user. With rerouting, instead of returning a link/card failure condition to the user, GRPM will enter a link failure procedure. In the link failure routine #DOWN, GRPM notes the downed link and issues the following message:

DS MSG: LU # nn JUST WENT DOWN

It then tries to find an alternate route to the destination if the link failure causes a change in the minimum-cost route to the destination. It also tries to broadcast the failure condition to the network by setting the change bit in the NRV so that the #SEND process can send out a Network Update Message to its neighbors (see #SEND Process).

Finally, the Message Accounting routines will regenerate the user message for retry if there is an alternate route, or return a DS04(1) error to the user if there is no path to the destination (the LU entry in the NRV is set to zero so that further messages to that destination will also result in a DS04(1) error).

If rerouting is not generated in or active on a particular link, link failures are processed by the retry processor, RTRY, and the NRV is not affected by rerouting. Retries are continued until a DS05 time-out error is returned to the master program.

"Up Link" Procedure

When the link level software (HDLC card and DVA66) discovers an operational link, it will send a link connection indication to GRPM (DVA66 only; for DVA65, see Compatibility Section below). GRPM will enter the "up link" procedure. Here, GRPM will note the return of the link by issuing the following message:

DS MSG: LU # nn JUST CAME UP

It will also send out an update message describing its knowledge of the network through the newly-connected link via #SEND. GRPM will then expect to receive an update message from the newly-connected neighbor showing its view of the network. These messages will then go to the network update message processor which will set up new, minimum cost routes.

Network Update Message Procedure

GRPM recognizes an incoming Network Update Message and will then enter the update procedure, #UPDA. This routine updates the cost to a destination referred by the message. If the minimum-cost route is changed due to the update, the new least-cost path will be reflected in the NRV and routing changes passed to the neighbors by setting the change bits in the NRV,then scheduling #SEND.

The #SEND Process

#SEND is scheduled (without wait) by the "link failure" procedure, #DOWN, or the update procedure, #UPDA, to send a Network Update Message to all the neighbors. It first checks to see if the send signal (#CMCT in RES) has been set by the link failure procedure or the update procedure. If the send signal is not set, #SEND exits. If the send signal is set, #SEND goes through the NRV looking for the change bit. When found, the node number and its cost value from the cost matrix are packaged to be sent to all its neighbors. When this process is completed, #SEND returns to test for the send signal.

Compatibility

Rerouting nodes are completely compatible with non-rerouting nodes; that is, traffic between them may exist. However, rerouting will only occur between nodes that have rerouting software. In the example below "R" indicates rerouting nodes, and "N" indicates non-rerouting nodes:

```
R---------R-------------N
|\        |             |
| \       |             |
|  \      |             |
|   \     |             |
|    \    |             |
|     \   |             |
|      \  |             |
|       \ |             |
|        \|             |
R---------RxxxxxxxxxxxxxN
```

Figure 5-2. Rerouting Topology Considerations

Automatic Rerouting can only occur within the R nodes. The extra link to N (denoted by xxx) does not provide any more reliability for the network because the software can only use a fixed path as long as the node on the other side of a link is a non-rerouting node. A manual switch of the LU using the DSMOD CN command can restore communications on the non-rerouting nodes.

For a link to be a rerouting link both nodes on either side must be Rerouting nodes.

The backward compatibility is achieved by setting the entry in the Cost Matrix (CM) to zero when a CPU#, LU pair is specified in the NRV initialization questions are asked by DINIT. The Network Update Message to change a "zero-cost" entry is ignored. Returned Network Update Messages (i.e., with DS06 indicating not a rerouting node or old node) are also ignored. Thus compatibility with DS/1000 nodes is achieved without modifying any software in the DS/1000 nodes.

However, while compatible in the network, there are some topology restrictions that make it inadvisable to leave an un-upgraded DS/1000 node, or one without rerouting, in critical locations in the network. There are cases where a "non-rerouting" node can cause some problems with transmission being possible in one direction but not the other after a topology change. For example (N = non-rerouting or DS/1000 nodes, R = rerouting nodes):

```
R1-----N-----R3--x--R5
|                   |
|                   |
|                   |
R2-----N-----R4-----R6
```

Figure 5-3. Rerouting Topology Considerations

If the link between rerouting nodes R3 and R5 fails, requests will be able to go from R5 to R1 but replies will not flow from R1 to R5. This is because routing updates generated by R5 indicating link R3-R5 is down are rejected as bad requests by "N" nodes, thereby preventing R1 and R2 from updating their tables. Since R2 and R1 can not receive routing messages from the rest of the network they effectively are non-rerouting nodes also. When the failure of R3-R5 occurs, R5 reroutes through R6-R4-N and gets its messages to R1. R1, having not received routing information, continues to use the R1-N-R3 path to get to R5.

Since higher-level software (e.g., Message Accounting) will eventually prevent requests from going to R1, the Network Manager is cautioned to set up a "mixed" network carefully.

It is recommended to set up your network such that non-rerouting nodes are "end" nodes. The example below would work nicely and shows many sub-networks which would also work.

```
N----R------R---------N---N
     |\    /|          \ /
     | \  / |           N
     |  \/  |
     |  /\  |
     | /  \ |
     |/    \|
     R------R
    /        \
   N          N--N
```

Figure 5-4. Rerouting Topology Considerations

Another issue in compatibility is related to the link level software. Although rerouting will support I/O drivers DVA65 or DVA66, for the old DVA65, the driver will not be able to generate an automatic link connection indication once the link is downed. This is true because DVA65 only supports the "unintelligent" communication cards. Thus, a user must re-enable the downed link when it is ready for operation again. This can be done by using DSMOD:

```
/DSMOD: OPERATION? /L

/DSMOD: ENABLE LU ? nn
```

The DSMOD "CN" command must also be used to restore the DVA65 LU from "0" to the appropriate LU. This function is performed automatically on DVA66/HDLC links.

Message Accounting

Message Accounting (MA) was developed to solve potential problems caused by the impossiblity of any link-level protocol to guarantee a message is delivered to the ultimate destination, and that it is delivered only once. MA functions independant of the driver or interface type. Such occurrances as power-failure or a node failing, with messages queued for re-transmission, illustrate this case. When messages are transmitted from the HP 1000 to the HDLC card it is then the card's responsibility to get the message to the next node in the Network. This is accomplished using the HDLC protocol which can assure delivery over the point-to-point link where the message becomes the responsibility of the next HP 1000 and HDLC card. Using the HDLC card, message accounting also allows DS to take advantage of the performance improvement achieved when I/O is considered complete when the DMA transfer to the card is complete. The transmission can be considered complete when the card has the message because MA has the capability of automatically retransmitting the messages if an acknowledgement is not received for a particular message. The additional performance is gained when a burst of traffic is processed by the HP 1000 and messages are transferred to several HDLC cards which then act as buffers for the CPU.

When the card has the message, the driver returns a successful completion and GRPM deallocates the message from SAM. In this situation, at any point along the path to the destination node there is the possibility that the message could be duplicated or lost with no regeneration of the message possible. Any link failure will cause the message to be lost and purged from the HDLC card. Rerouting will occur as a result of the failure but the messages present only on the card attached to the failed link are lost. Clearly a message regeneration capability is needed.

As was mentioned, Message Accounting functions independantly of the interface and driver type. When the HP 12771/12773 cards are used the MA function is performed the same. MA is not used on messages to and from the HP 3000.

Message accounting software has been developed to provide an "end-to-end" protocol where message regeneration capability is maintained. As opposed to the link-level protocol, the MA end-to-end protocol insures that the message gets from the origin node all the way to the slave monitor class queue at the destination node. For replies, MA insures the reply makes it back to GRPM at the origin node. No guarantee is made that the master program is still present in the system, MA just guarantees the reply will be received by the communications management level software at the origin node if a path between them still exists. At this point, if the master requestor has not been aborted, the message will be queued to the master and the transaction complete. If the master has been aborted the message is flushed and an error message is printed on the system console.

When a link fails, and rerouting is in the system but not MA, there is no record of which messages were received at the destination and which were not. Through the use of MA sequence numbers in the messages, MA causes the retransmission of lost messages and insures no duplicate messages will be passed on to the user. Lost message retransmission works as follows:

When a message has not been acknowledged within a certain amount of time, a message is sent out to verify that the MA channel is still connected. The current MA sequence numbers and flags are returned as an acknowledgement that the channel is still working. The sending side then resends any messages that the other side has not received as indicated by the MA variables it just received.

By assigning sequence numbers to outgoing messages and by the receiving end keeping track of which sequence numbers have already been received from each possible destination node, MA will flush any duplicate messages.

Message Accounting Terminology

The end-to-end connection will be referred to as a "channel". This channel is a logical connection and has no relationship to the physical link, link characteristics, or path (in fact, with rerouting the path may vary).

The following 3 variables are found in every message.

NS - Current MA send sequence number
NR - Current MA receive sequence number
NC - Current MA cancel flags

The following variables are kept in the MA table. There is a separate set for each MA destination node (each channel).

VS - Next send sequence number
VR - Next expected sequence number
VA - Last sequence number acknowledged
VF - Receive flags
VC - Cancel flags
VT1 - Acknowledgement Timer
VT2 - Idle Timer
VCC - Consecutive cancel counter

Message Accounting Table

The Message Accounting Table describing the data structure of these variables is found in Appendix C.

Message Accounting Initial Connection

Before any valid sequence numbers are assigned, the two nodes across the channel must be logically connected (i.e., their sequence numbers must be synchronized). Remember the two nodes may be anywhere in the network with any number of intervening nodes. MA is a logical channel between the end points.

Any traffic to a node whose connect state is "down" (i.e., logically disconnected) will cause the initialization handshake to be initiated to establish synchronization between the two nodes and their MA variables.

When #MAAS is called requesting an MA sequence number to a "down" node, it assigns -1, an invalid number, which causes the message to be dropped when it reaches its destination. The philosophy is that no messages should carry MA sequence numbers unless it can be assured that they are valid. What this does is initiate an initialization sequence between the two MA nodes and eventually the "dropped" message will be sent via MA retries.

The write completion of the above message will cause #MAPP to begin the initialization handshake. This handshake consists of an INIT message, answered by either an IR (Initialize Response), RR (Receiver Ready), or CAN (Cancel Outstanding Messages) message.

During a transition from a down link, once the connection has been completed, any non-delivered messages will be retransmitted via the MA retry mechanisms.

If the connect request is not successful, the state will change to "down". All requests will be returned to the master routine and all slave monitor replys will be cleared from the holding class (#MAHC). If a message is received on a channel thought to be 'down', an idle message is generated from the receiving side which causes the processing described above to take place.

Message Accounting Sequence Number Assignment

An MA sequence number is obtained by a call to #MAAS. #MAAS checks the number of unacknowledged messages for this channel. If the number exceeds 15 a DS08(1) error indication is returned.

Otherwise, the current values of VS (Next Send Sequence Number), VR (Next Receive) and VC (Cancel Flags) are set in the request/reply header. VS is then incremented and stored back in the MA table entry for this channel.

Message Accounting Receive Processing

Before GRPM passes incoming messages to the appropriate receiver, the messages are first filtered by #MAPP. Any invalid messages are dropped at this point by #MAPP.

Invalid messages are those with an NS number outside of the allowable window or those with an NS already received. This window arrangement is illustrated below:

```
               VR+W-1   (inclusive)
        NS --> |                    Receive Window
               VR       (inclusive)
```

NS = Sequence Number in received message.
VR = Next expected Sequence Number to receive.
W = The window value for incoming messages (constant 15).

If the message is acceptable, but NS is not equal to VR (ie., a message received out of order), then VR is not incremented. Instead the appropriate bit is set in the receive flags, VF.

Each bit in VF represents a sequence number (NS) relative to VR; bit 0 corresponds to VR, bit 1 to VR+1, bit 2 to VR+2, etc.

When an NS is received equal to VR, VR is incremented and VF and VC are shifted over. The amount of the increment/shift is dependent upon the bits set in VF. These bits in VF indicate messages that have already been received out of sequence. The least significant bit of VF is tested and if a 1, VR is incremented and VF is shifted. The increment/shift is done until bit 0 of VF equals zero. Zeros are shifted into the high order of VF and VC. This procedure updates VR catching all messages received out of order.

The exception to the above processing is the receipt of a cancel or no-response message. A "cancel" message can be thought of as multiple messages. Because of this, the checks for "already received" are ignored. The VR/VF/VC increment/shift amount equals (NS-VR+1). The cancel message cancels all messages in the range VR, the expected sequence number, to NS, the sequence number contained in the cancel message. Those messages not received are considered cancelled.

A "no-response" message also does not follow the normal receive processing. If a no-response message is received, the channel becomes logically disconnected. This message is received when MATIC has attempted the maximum number of cancel messages and receives no acknowledge from the destination node. MATIC generates the no response message to GRPM and the MA routines. #MAPP changes the state to "down". Timeouts (DS05) are sent to all waiting masters, and all replys for that particular channel are flushed from the MA holding queue (#MAHC). After any message is received, with the exception of idle messages (RR), the idle timer is set (VT2).

Message Accounting Message Acknowledgement

Incoming messages, regardless of their receive status, may carry acknowledgements for previous outgoing messages. The acknowledgement indicates that MA on the other side received a valid message.

Messages will be acknowledged when the receive sequence number from the message, NR, falls within the following window:

```
                    VS   (inclusive)
        NR -->      |                   Acknowledge Window
                    VA+1 (inclusive)
```

NR - Current MA receive sequence number
VS - Next send sequence number
VA - Last sequence number acknowledged

When a message is acknowledged, VA will advance by the amount (NR-VA). Note that several messages can be acknowledged with one received message. Each increment value of VA causes the cancel flags in the message (NC) to be checked for acknowledgement of a cancel message. The relationship between NR, VA, and NC is shown below:

```
NR-1      VA
|         |
v         v
+---------------+
15    NC        0
```

The cancel flags field (NC) incoming message provides the acknowledge that cancel messages have been received by the destination node which now allows the original message to be regenerated. Each bit in NC represents a relative position to NR; NR-1 corresponds to bit 15, NR-2 to bit 14, etc. Each bit set in NC between NR-1 and VA will cause a message retransmission attempt.

If the currently accessed bit in NC is not set, one of two things will happen. If the acknowledged message was a request, then the MA identifier in the TCB is cleared and the "request acknowledged" bit is set (bit 13 of TCB+1). This bit is only used in timeout processing to determine whether the request or the reply timed out. If the acknowledged message was a reply, then the reply will be flushed from the MA holding queue (#MAHC).

Message Accounting Message Timeout

The MA "watchdog" timer, MATIC, runs every second. Its sole purpose is to check the status of the two timers (VT1 & VT2) for every MA channel. The acknowledge timer VT1 indicates how long to wait for an acknowledge for the message just sent. The idle timer VT2 indicates how long to wait in the destination node for return traffic to be generated toward the origin node before special MA traffic is generated to carry the message acknowledgement parameters back to the origin. Both timers are checked to see if they are running (ie., non-zero). If so then each is decremented and checked for zero (a timeout).

If an acknowledgement is not received in time, VT1 will time out. If this happens, VCC is incremented indicating another timeout. If this value is less than or equal to the MA retry limit (#MARL) then a cancel message is sent across the channel. The NS for this message is VS-1 (ie., the same as the last message sent). Acknowledgement of this cancel message will cause one or more messages to be retransmitted.

If #MARL consecutive cancel messages have been sent without acknowledgement, MA will conclude that the logical connection to this channel has been severed. MATIC informs the MA module in GRPM (#MAPP) of this by sending a no response message (NR). This is a local class I/O message.

If VT2 has timed out, this indicates that an outbound message has not been sent in time in which to piggyback acknowledgements. Therefore, an idle message (RR) is sent to carry possible acknowledgement information (ie., the NR and NC fields). The NS for this message is VS-1 (ie, the same as the last message sent).

Message Accounting Message Retransmission

If a message does not receive an acknowledgement within the timeout interval (VT1), MA does NOT retransmit the message immediately. There are two reasons for this.

First, there are circumstances that could prevent a message from ever getting through. In this situation, the send sequence number NS used by this message would be held until the retry limit (#MARL) was exhausted. The acknowledgement window would thus eventually become frozen and prevent any further traffic. The cancel message allows NEW sequence numbers to be given to retransmitted messages, thus freeing up the movement of the sequence window.

Secondly, an acknowledgement timeout affects all messages to that channel. Immediate retransmission would cause a flurry of retransmissions including messages that might actually have been received. The cancel message allows selective retransmissions of only those messages that were lost.

When a cancel message is received those messages between NS and VR that have not been received are flagged as cancelled. NOTE: Should the data message arrive late (ie., after the cancel message has been received), it will be discarded as a duplicate since that particular NS has already been received. Each acknowledgement at the sending side checks for this cancel indication in NC. If the appropriate cancel bit is set a retransmission attempt will be made.

To retransmit a request, a message of length one(1) is sent to the appropriate master. The correct master is found by searching the TCBs for the correct sequence number. #MAST will then carry out the retransmission (including assigning a NEW MA sequence number).

To retransmit a reply, the holding queue is searched (#MAHC) and the appropriate slave reply is resent (again with a NEW MA sequence number).

Message Accounting Disconnect

If no acknowledgement for a message has been received after #MARL cancel attempts, MATIC sends a no-response (NR) message to GRPM. This is a local class I/O message.

When #MAPP receives this message it begins a cleanup cycle on this channel. An error message is formatted using the last reported error. If there were no errors reported then "DS05(3)" is used. This error message is sent to all masters waiting for replys from this channel. These masters are identified by searching through the TCBs.

The MA holding class (#MAHC) is also searched. Any slave transmissions awaiting acknowledgement (from this channel) are discarded.

The channel state is set to "down" and the line-down count is incremented. Any inbound messages (with the exception of an initialize request) will be discarded in this state. The channel will stay in this "down" state until either an INIT request is received or an attempt is made to send a message on this channel.

Message Accounting GRPM Interface

The cooperation of GRPM is essential for Message Accounting to function. There are two reasons for this. First, MA must act as a filter for message flow, discarding unwanted messages. GRPM must cooperate in this. Secondly, link level errors are determined (at least reported) at the GRPM level. Since there is no way for MA to determine (unfortunately) that a message is being returned with a link error, GRPM's cooperation is needed to flag these messages.

Before GRPM gives an inbound message to the appropriate user, it calls #MAPP to verify that the message is valid. All invalid messages and MA stream zero messages are discarded by #MAPP. Once the message is discarded, GRPM will discontinue any processing of this message and go on to the next message.

If GRPM detects a link failure on a request, it attempts to return this message to the master with the appropriate error code. In order to indicate to MA that a link error has occurred it also flags the NR field (with -1). If MA sees a message with the "link error flag" set it will record the error and "swallow" the message. Since MA's retry mechanism will eventually resend the original message, it is hoped that either the link problem has gone away (such as in a busy condition), or rerouting has found an alternate path. Only after #MARL attempts will MA inform the master (using the last recorded error).

RTRY

For DVA65 links RTRY performs all write-retries in the DS/1000-IV system. If a transmission fails, GRPM re-threads the class buffer onto RTRY's class number and stores the absolute time at which the retry is to occur in the local appendage of the class buffer header. When RTRY receives a new request, it checks to see if the absolute start time has already been reached. This could happen, for example, if it took a long time for a partition to become available. It could also happen if two or more requests arrived before RTRY was finished with another one, the second one having an established time for retry earlier then the first. If RTRY finds the retry time to be greater than the current time of day, it will suspend itself until then. When the retry time is reached, it re-threads the class buffer to the output EQT on GRPM's class number. If an error occurs with this operation, RTRY sends an error notification to QCLM's class number and de-allocates the class buffer.

A maximum value for the retransmission delay is stored in a DS entry point #RDLY in SSGA (part of RES). It is nominally set at 2 seconds (i.e., a value of -200). It may be modified by a user program, but should only be done by a sophisticated user.

For DVA66 links (HDLC) link-level retries are performed on the card.

QUEUE

QUEUE is the DS/1000-IV program scheduled by the communications driver when a new request arrives. This driver could be one of the DS drivers (DVA65, DVA66, DVR07), or the customized subroutines CSV66 or CXL66 for X.25 links. QUEUE's job is to provide a data buffer into which the data is read, via a Class I/O call. The request and data lengths are first checked for validity and the validity of the interrupt is checked. It must come from an initialized interface driven by DVA65 or DVA66 or it is considered a spurious interrupt and ignored (this condition may also occur if the RTE interrupt table was generated improperly).

If the request length is outside the allowable range (0 or 7 to 63 inclusive), QUEUE calls the driver to send a STOP. If there is no problem, QUEUE issues a Class I/O read using GRPM's (1000-1000) or QUEX's (1000-3000) class number (or PROGL's if the request is a CBL down-load or VCPMN's if it is a Remote Front Panel request.)

If an immediate reject of the request occurs because there is insufficient SAM, QUEUE calls the driver to send a STOP (both conditions result in a busy-reject being received on the master side). Otherwise, if the "read" is allowable and there are no active TCBs, QUEUE attempts to lock the "quiescent resource number". If the system is going into quiescence, QUEUE will be suspended here. Incoming data buffers will receive remote-busy rejects, because the driver cannot schedule QUEUE. If the RN is not locked, QUEUE terminates with the RN clear.

The following error conditions can occur:

1. Interrupt not from DVA65 or DVA66 -- it is ignored.
2. Interrupt not from initialized communication line -- clear the driver.
3. Interrupt from an EQT with no LU pointing to it -- it is ignored.
4. Lengths of data or request buffers out of range -- STOP sent.
5. GRPM's class is bad -- catastrophic error reported

```
DS ERROR: PROG=QUEUE  SEQ  =nnnnnn P=oooooo A=oooooo B=oooooo

where:
      nnnnnn = transaction sequence number
      oooooo = the named register contents
```

6. Not enough SAM to hold the request and data -- STOP is sent.

#SLAV

#SLAV is called by DS/1000-IV slave monitors to send a reply (and data, if any) back to the origin node. It performs the following steps:

1. Deallocates the slave TCB. If this fails, the error return is taken.

2. Checks the no reply bit in the request header. If set, just returns to caller.

3. Clears the message accounting send and receive sequence numbers and cancel flags.

4. Enters the maximum hop count into the message header.

5. Verifies that the reply length is in the range 13 to 37, inclusive, or, if not, returns a DS03 error.

6. Clear message accounting retry counter in message appendage.

7. Converts the nodal address of the origin node to an output LU. If this conversion fails, then a DS04 error is returned. This can only happen if the NRV for the slave side does not match that of the rest of the network, allowing requests to arrive from a node which cannot be accessed or if the link fails while the reply is being formulated.

8. Calls #MAAS to assign message accounting sequence number to reply.

9. Assigns software upgrade level of origin node and either sends reply to the message output converter, OTCNV, or to the driver using GRPM's class.

10. Returns to caller.

NOTE

RFAM ignores all errors returned by #SLAV. P-to-P slave routines convert them to -47.

Notice that transmission errors may occur in the reply, but this information is not available to the slave. Should all retry attempts fail, the reply will be flushed and the master program will eventually time out. When Message Accounting is present in the node the slave replies are maintained in a holding queue until the reply is successfully received by the origin node.

#GET

#GET is called by routines waiting to receive requests (and possibly data) on their class numbers. #GET performs the following functions:

1. Issues a "get" on the class number passed to it.
2. When the reply arrives, returns a DS03 if the request length exceeds the maximum allowed.
3. Moves the request into the user's buffer (up to the number of words specified in the call).
4. If the specified data length is exceeded, returns a DS03 error.
5. Deallocates the class buffer.
6. Returns the received request and data sizes in the A- and B-registers.

UPLIN

UPLIN protects the system users from waiting forever if one of the nodes in the network should fail or if transaction processing takes too long due to delays caused by insufficient swap tracks, SAM, or lost messages, etc. UPLIN also releases DS resources allocated to programs which aborted or terminated without releasing them, such as TCBs, PNL entries, class buffers, class numbers, etc. UPLIN is scheduled for execution every five seconds and performs the following functions:

1. Checks/waits for system quiescence. Prints message to operator when achieved.

2. Updates slave TCB timeout values. If a transaction has timed out, the TCB is returned to the available pool and if the monitor is OPERM, CNSLM or EXECW, the monitor is aborted. This allows other transactions to be processed, even if one is submitted which can never be satisfied, for whatever reason.

3. After processing each slave TCB list, UPLIN checks to see if the corresponding monitor is dormant. If so, it is rescheduled and passed its class number. Thus, the system can recover if an operator should abort one of the monitors, although the transaction it was processing may be lost (the master will receive a timeout error). If a slave monitor's ID segment no longer exists, the class buffers and all slave TCBs are released with a one-word message being sent to RPCNV for each slave TCB if it is an HP 3000 slave TCB. Note that the class # itself is not released. This allows you to be able to initialize DS and to specify slave monitors which are not yet present, and then to download or RP them. UPLIN automatically enables them at its next execution cycle.

4. Updates master TCB timeout field. If a TCB times out or if it is dormant or if the "bad" flag in the TCB is set, the master class number and the TCB are released. If the program is in any "wait" state (program status not equal to 1), a null request is written to the requestor's class number.

5. Scans the Process Number List and sends an HP 3000 "KILL" or HP 1000 logoff for any abandoned sessions (i.e., the program which signed on to the remote system did not send a "BYE" or logoff before terminating). The Process Number List entry is cleaned. The HP 1000 logoff is handled by calling #UPSM.

6. Scans the HP 3000 LU Table. For each HP 3000 LU that went down, UPLIN cleans up the HP 3000 tables.

   Deletes all associated Process Number List entries (PNL).

   Simulates a time-out on all associated master TCBs.

   Clears all associated Transaction Status Table (TST) entries and corresponding slave TCB entries.

   If the LU is an X.25 LU, releases the X.25 POOL LU and deletes the LU from the 3000 LU Table.

7. Call #UPSM to increment timeout counters in the Remote Session Pool (#POOL) entries.

8. Reschedules GRPM, RTRY, QCLM, LUMAP, RSM, OTCNV, INCNV, MATIC, QUEZ, QUEX, RPCNV, or RQCNV if they were scheduled by DINIT but now are dormant.

9. If PLOG is dormant or operator suspended but PLOG's class and SAM buffers or resource numbers are allocated, both are deallocated.

Remote Session Monitor

Overview

Remote Session Monitor provides the capability to interface to session monitor nodes within the session environment. All monitors attach and detach to specific sessions created either as default sessions or as specific sessions using calls to DLGON/DLGOF or DLGNS for non-session access to a node.

To implement Remote Session, DINIT asks questions necessary to allocate a SAM block for tables, to determine what account the default sessions will use, and a password for non-session access.

DSMOD provides commands for altering these parameters.

UPLIN takes care of logging off abandoned sessions with an appended Remote Session routine.

Each master program has part of Remote Session appended to provide remote and local session ID's for outgoing messages. If there are no session nodes anywhere in the network or a particular node will never access a session node, then remote session is not generated in and a dummy library ($DSNSM) is provided to satisfy externals.

Implementation

#DISM - DINIT Session Module

During initialization when DINIT is calculating the size of the SAM block that will be needed for all the DS tables - the routine #DISM is called.

It checks to see that Session Monitor is installed and active in the system, and if so, asks the question:

MAX # LOCAL SESSIONS FOR REMOTE NODES?

The response to this question is multiplied by 7 words per session and returned to DINIT as the size requirement for the Remote Session ID pool. DINIT allocates the additional SAM block and initializes a RES entry point #POOL with the base address of the Remote Session Identifier pool (table).

When the total amount of SAM has been calculated by other questions asked during DINIT initialization and SAM is allocated, #DISM is called again to fill in the ID pool. Using the number of pool entries indicated in the previous paragraph, #DISM searches for undefined LU's in the system starting with LU 253 and working down. The undefined LU's that are found are used as remote session identifiers for this node and are entered into the Remote Session Pool entries. If a system is generated where all 255 possible LU's are defined, Remote Session will not function.

In later calls from DINIT, #DISM will ask for and store away the default user name, and password for non-session access.

#RMSM - REMAT Session Module

Appended to REMAT, this remote session module handles the expanded SW command by extracting the optional account names for Node1 and Node2, and generating logon requests to the module RSM (discussed later) at the remote nodes.

#RMSM processes the attach (AT) and detach (DE) commands and provides special processing for the exit (EX) command. When the EXIT command is issued it is possible to have up to 16 sessions active, one at each of 16 different nodes. The EX processor generates a logoff request to each active remote session.

A network account table extension (NATX) is maintained by #RMSM and contains the account names for each of the 16 possible active accounts. When the SW command is issued without parameters, the account names at node1 and node2 are displayed from the NATX contained in #RMSM.

DLGON - Distributed LOGON/LOGOFF

This module is appended to each master routine that calls DLGON, DLGOF and DLGNS. This module generates requests to Remote Session Monitor, RSM, in the destination node to perform the logon/logoff. The messages go out on stream 7 which is normally used by OPERM. To distinguish the logon/logoff requests from regular OPERM requests an illegal operation code of "XX" is used. GRPM notes the absence of a destination session ID and passes the request to RSM rather than OPERM.

RSM checks the command code, finds the "XX" and processes the request as a logon/logoff request. OPERM never receives these special requests.

#NAT - Network Account Table

This module maintains the Network Account Table (NAT). The NAT is used to keep track of the 16 remote session ID's, each at a different node, that the program can have active at any one time. Each entry in the NAT consists of:

Word 1 - Node Number
Word 2 - Remote Session ID, Remote Sub-level and ownership flags
Word 3 - Address of user name in NATX. (Used only for REMAT)
Word 4 - EXECW Sequence Number
Word 5 - PNL entry pointer

Word 3 is used by REMAT/#RMSM to keep track of the 16 possible account names associated with the 16 different remote sessions REMAT may be attached to at any one time.

Word 4, the EXECW Sequence number, is used when the master program is memory-resident, scheduled from a remote node via EXECW, and has to insure that the NAT gets cleared between each execution of the program.

Word 5, the PNL entry pointer, is used to clean up PNL entries (see the following discussion of PNL).

#MSSM - Master Program Session Module

The main function of #MSSM is to provide remote and local session ID numbers when called by #MAST, which is trying to send out a message. The Remote session ID's are stored in the NAT if they exist. If no explicit logon has been performed, #MSSM searches for a father program that may have established a session at the destination node already. If a father is found that has an active session at the destination, that session ID is used. If no father is found, #MSSM checks if any other program scheduled from a session at this terminal has a session at the destination node and will use that session ID. Failing that, #MSSM checks if the program or an ancestor was scheduled from a session at the the destination node and will use that session ID. If no active session can be found the first message sent into the destination node will be routed through RSM where a default session will be created. The session ID is entered into the Request header and the request required to the appropriate monitor, when the reply returns #MAST calls #MSSM who enters the new remote session ID into the NAT. Any further requests generated by this master will use the already-established session ID and bypass RSM processing in the destination node.

RSM - Remote Session Monitor

Like any monitor, RSM hangs on a class get waiting for a request to process.

When a request arrives, RSM checks for stream 7 requests (OPERM) with a command code of "XX". If this is the case the ICODE value (0 = logoff, 1 = logon, 2 = non-session access) is picked up and the appropriate session routine is called.

For logoff, the session ID is passed via class write, to the session program LOGOF, using the class $LGOF. Logon is handled the same except using class $LGON. The session ID is returned in the reply header and stored in the NAT by the master program's #MSSM module. For the non-session access request, if the password supplied is correct, special session ID 254 is entered in the request/reply header and the reply sent back to the origin node (session ID 254 is reserved for this case). RSM then builds the Remote Session Pool entry of the newly-established session. See Appendix C in Volume II of the Network Manager' Manual for the format of a #POOL table entry. If the request was passed to RSM for logon processing because it had a session ID of zero, then the request is rethreaded on the appropriate monitor's class. RSM then returns to the class get to wait for another request.

#ATCH - Slave Monitor Attach Routine

All slave monitors call #ATCH to attach to the destination session ID found in the request header. #ATCH is a Remote Session routine which does the following pre-processing prior to calling the Session Routine ATACH:

- Ignores the call if this is not a Session Node (a dummy #ATCH is provided in non-session nodes ($DSNSM or $DSLSM)).

- If destination Session ID is 0 or 254, ignore the call. The request will then be processed outside the session environment.

- If the request is being handled in the local node (destination node = origin node) go attach to local SCB.

- Reset the timer word in #POOL entry. This timer, called the Idle Session Timer, keeps track of the time since the last access of the session, and is used to time-out a session that is not being accessed. The default timeout is 5 hours. This timeout value can be displayed with the "RS" command in DSINF and can be modified with the "\T" command in DSMOD. (The field is not used for any other purpose.)

- Attach to the session.

#SCSM - Slave Clone Session Module

This routine is called by EXECM and PTOPM to provide the slave cloning feature. #SCSM in a session node updates the associated entry in #POOL which is used by PTOP slave programs or programs scheduled by EXECW in directing replies back to the originating session if the master was running in a Session node. If the user requested a copy of the slave program, the program is duplicated and renamed.

If a slave is scheduled from a session node but runs in a non-session node, pseudo #POOL entries are still used. When the slave program does DEXEC/RFA etc., requests back to the master, as in the case of the remote Editor, for instance. #MSSM is still appended to the programs in the non-session nodes and will locate in the pool entry a session ID to use for any requests directed back to the master session.

DS/1000-IV MESSAGE HEADER

The DS/1000-IV message header contains the session ID word at offset #SID (see Appendix A for symbol offset definitions). The left and right bytes are "source session ID" and "destination session ID" , respectively, if any.

Session ID 253 represents the "permanent old node" default session. Session ID 254 represents special non-session access to Session Monitor node.

MESSAGE FLOW

To tie this all together, let's follow a PTOP POPEN request from one session node to another.

The master program issues a POPEN request with no prior session established. POPEN builds a message buffer and calls #MAST. #MAST makes preliminary entries in the message buffer and calls #MSSM to acquire Remote and local Session IDs for the message. #MSSM finds no ID present in the NAT (Network Account Table contained in #NAT) so it looks for a father that may have a session active at the remote node. If any father is found, a scan of the Process Number List is performed to find an entry for the father. If found, the session ID will be extracted and used by the "son" program to access the same session as the father. If no sessions can be found, the message goes out with no destination session ID.

When the message arrives in GRPM's queue at the remote node, if a destination Session ID is present in the message it is routed directly to PTOPM. In the case of this example where no session has been created, the message is routed to RSM where a new default session will be created for transactions from this master. The destination Session ID is placed in the request header and the request is relinked to PTOPM's class.

PTOPM attaches to the indicated session by calling #ATCH and schedules the slave. The slave "ACEPT" reply is returned to the driver on GRPM's class via #SLAV. The master in the origin node is waiting on a Class Get in the routine #GET.

When the slave reply arrives back at the master, #GET returns to #MAST. #MAST calls #MSSM to process the reply. #MSSM stores the Remote session ID in the NAT (Network Account Table) and creates a Process Number List(PNL) entry. The PNL entry will be used to generate a log-off when the master goes dormant, if the master doesn't generate its own. #MSSM returns to #MAST which returns to POPEN which returns to the master program.

When a PWRIT is issued, #MAST calls #MSSM which now has the remote and local Session IDs in the NAT. They are extracted from the NAT and entered into the message. The message is written to the driver and transmitted to the destination node. GRPM on the receiving side notes that a destination session exists, skips the rethread to RSM, and dispatches the PWRIT to PTOPM. PTOPM locates the slave's class and rethreads the PWRIT to the waiting slave. The slave generates its reply which is transmitted over the link and received by #GET in the master program. #GET returns to #MAST which calls #MSSM to make any changes to the NAT entries. There will be no changes on continuing message transfers so #MSSM returns through #MAST to the master. The master issues a PCLOS, the slave and the master exit, but the session is still there.

UPLIN executes every 5 seconds and, among other things, calls the routine #UPSM (UPLIN Session Module). #UPSM scans the PNL looking for entries whose associated program ID segment indicates a dormant state. UPLIN finds that the master program has gone dormant failing to log off its remote session and generates a no-reply logoff request to RSM at the remote node and clears the PNL entry. UPLIN then continues its normal processing.

PTOPM – Program-To-Program Communications

PTOPM is the monitor which receives all program-to-program communications messages on the slave side. It services POPEN, PWRIT, PREAD, PCONT, PCLOS, FINIS, and requests from the REMAT commands SL (slave-list) and SO (slave-off). The last four are processed entirely by PTOPM. The first four are passed to the slave program, where the slave routines GET, ACEPT, and REJCT process them.

POPEN Processing

For POPEN, PTOPM checks to see if the slave program exists. If not, then error -41 is returned, and PTOPM gets the next request.

#SCSM is called to perform optional cloning and remote session functions. If the slave program exists, it will be scheduled whether or not it is already in PTOPM's table. This ensures that a POPEN call will always operate without error regardless whether the slave program has previously been aborted, or terminated without a PCLOS or a FINIS call.

PTOPM maintains a table of the slave programs currently "open". A maximum of 20 are allowed. If this table is full and a new POPEN request arrives for a new slave, error -42 is returned. If the program named already exists in this list, then that entry and class number will be used, and the request is threaded onto the program's class number and PTOPM gets the next request. Otherwise, a new entry is made and a new class number obtained. If there are no available class numbers or SAM is not available, error code -42 is returned to the master. If both exist, the program is scheduled and passed the newly-obtained class number. If an error occurs here, error code -48 is returned. Otherwise, the request is threaded onto the class number. PTOPM goes back to get the next request.

Certain programming conventions must be followed if a slave program can have more than one master (allowed in RTE-to-RTE communication only). Since it is likely that each of the masters will be communicating with the slave independently of the others, the slave should maintain a table containing information relevent to the context of each master. The slave then allocates a new table entry for each POPEN it receives. The tag field should be used by the masters to identify themselves (e.g., TAG(1) = subscript of context table entry). Master programs must not close slaves explicitly, since no counter of the number of masters talking to each slave is maintained. Instead, it is suggested that PCONT, possibly with a special function code in one of the TAG words, be used by the master to signal the end of its interest in the slave, which then clears the entry in its context table. When all entries are clear, it may either wait for further requests by returning to The GET routine or call FINIS and then terminate. The first alternative may be taken for example, in the case where the slave must maintain other data tables, which must not be "forgotten". If the slave cannot accept a POPEN because its table is not large enough, it must reject the call. Access to the slave by the other masters is unaffected.

PREAD, PWRIT, and PCONT Processing

PREAD, PWRIT, PCONT are all processed similarly. PTOPM's internal table is searched for the slave program, utilizing the ID segment address from the master's PCB. Of course, if this has been modified or for any reason the program is not found in the table (as could happen, for instance, if an "SO" command was issued to terminate the slave or if PTOPM was re-loaded on-line), error code -44 is given. The class number from the PCB is compared to PTOPM's internal table entries and error -103 is returned if there is no match. Otherwise, the request is threaded onto the slave program's class number. As with POPEN, if errors occur here, error -48 is returned.

If a slave program has too many requests queued onto its class, (see CLASS I/O RE-THREADS in this chapter), the new request is returned to the master with error -58. If the slave program is dormant and not in the time list, the request is returned with error -45.

PREAD Processing

PREAD sends a message to the slave containing the data buffer length, which is picked up by the GET subroutine and returned to the caller. When the ACEPT routine is called by the user, the user's data is passed back to the master program by building a communications message which is rethreaded to GRPM's class number.

PNRPY Processing

When the 'no-reply' option is active, any PWRIT, PCONT, or PCLOS call has the 'no-reply' bit set in its message header.

On the master side, a TCB is allocated with timeout equal to the ITTO parameter in the PNRPY call. When the TCB times out, UPLIN deallocates the TCB, but does not generate a zero-length record or report an error as is usually done on TCB timeouts.

On the slave side, the slave must still process the master calls as usual by doing an ACEPT or REJCT. #SLAV, which is appended to ACEPT and REJCT, deallocates the slave TCB and checks the 'no-reply' bit. When this "no-reply bit" is set the reply message is not sent.

PCLOS Processing

PCLOS aborts the slave and PTOPM generates a reply to the master request. Resources allocated currently by the slave are lost to the system. PCLOS should not be used if the slave has other masters or maintains data tables which should not be destroyed.

GET Processing

The class number specified in the parameter ICLAS is used to get the master request and the slave program is provided with the function code of the master request in IFUNC. On PWRIT, GET moves the request buffer into the user space if the buffer is specified in the GET call.

ACEPT Processing

ACEPT implies that the master request is valid. On PWRIT, the request data is moved into the user space unless it has been previously obtained in the GET call. On PREAD, the read data is included with the reply and SAM is deallocated. On all master calls (PREAD, PWRIT, PCONT, POPEN), the reply bit for accept is set and TAG is included. The driver is called through #SLAV and clean-up is done for the return to the GET call.

REJCT Processing

REJCT implies that the master request is invalid. A 'no data' reply is created, TAG is included, and the reply bit is set for reject. The driver is called through #SLAV and clean-up is done for the return to the GET call.

FINIS Processing

If a slave program calls FINIS, the request is serviced by PTOPM. If the slave has other requests it has not yet received, these are cleared out. The class number is released. The program's entry in the table of active programs is cleared out. Note that the slave is not terminated by the FINIS call.

SLAVE LIST Processing

The Slave List (SL) is a table used to move the names of all slave programs into the reply buffer, which is sent back to REMAT. This is the only request to PTOPM which generates a non-zero data length reply. Note: the order of listing programs has absolutely no relationship to the order in which they were "opened".

SLAVE OFF Processing

If the Slave Off (SO) request is to "clear all slaves", then the internal table is used to generate PCLOS processing for each program in the list. Otherwise, a PCLOS is only generated for the named program.

RFAM - Remote File Access Monitor

RFAM (the Remote File Access Monitor) receives all requests built by the RFA master routines (e.g., DOPEN, DCLOS, DWRIT, DREAD). It makes FMP subroutine calls (e.g., OPEN, CLOSE, WRITF, READF) on behalf of the master programs. It requires storage for one 144-word Data Control Block (DCB) for each file opened by any master program. There are two versions, one which contains storage for one DCB and one which can handle many more, storing DCBs in the remaining unused memory in its partition and using the disc as an overflow storage area if more DCBs are required.

The single-DCB version operates identically to the multiple-DCB version, except that it never has to determine where the DCB for a file resides.

The multiple-DCB version contains sufficient memory space for two DCBs inside its program space. If generated as a memory-resident program, a maximum of two open files is possible. You can obtain a performance improvement if the number of files you require RFAM to handle can be stored in the unused memory of its partition. RFAM will calculate this for you and print the answer if you reply to the question "INPUT # FILES?" asked by DINIT with a -1.

When each request arrives, a copy of the four-word pseudo-DCB stored in the user's program is used to locate the DCB RFAM stores. If the DCB is not currently in memory (multiple-DCB version only), it is fetched from its location in the disc overflow area. (A least-recently-used swapping algorithm is implemented to minimize disc accesses). Both versions of RFAM handle this DCB identically in the following respects:

- DOPEN, DPURG, DNAME, and DCRET are executed by RFAM without any information from the four-word pseudo-DCB. The master subroutine copies information from RFAM's reply into the user's four-word pseudo-DCB. If you have a file "open" and then call any of these routines with the same pseudo-DCB, RFAM will maintain the DCB for the original file and allocate another DCB entry for you. Even though RFAM still has both files "open", you have lost the ability to access the first file. Therefore, the user must explicitly close all files when no longer required. REMAT provides an "FL" command which you can use to clear out files which have not been closed. Notice that the analogous FMP subroutine calls are slightly different in this respect. They will automatically close any "open" file before executing OPEN, PURGE, RNAME, or CREAT calls.

- All other master RFA routines send a copy of the pseudo-DCB to RFAM which uses the information to locate the DCB, but the reply does not cause the pseudo-DCB to be modified.

- DPURG and DNAME do not use or modify the pseudo-DCB.

- Extended file calls are implemented by providing alternate entry points (DXCRE, DXCLO, DXREA, DXAPO, DXPOS, DXLOC) in the corresponding file access routine. The double word parameters are used where necessary and the function codes used by RFAM at the destination node are 14 through 20 rather than 0 through 12 used by the regular file calls.

DEXEC

This utility subroutine is included in user programs which request remote EXEC functions. When the destination node is not local, it builds a request buffer (and optionally data) for transmission to the remote computer. When the request arrives, it is routed by GRPM to either EXECM (or EXECW, in the case of a remote schedule-with-wait), as described in the first part of this chapter. In the case where the destination is local, however, the EXEC call is made by DEXEC to improve efficiency. In this case, however, no buffer-length check is made. Programs which make use of this fact will run only when the destination node is local if their buffer lengths exceed 512 words. The special "write-read" DEXEC call (ICODE = 1, bit number 11 of the control word is set) is implemented by building a request buffer and sending it to EXECM.

DVA66 – 1000/1000 HDLC – 1000/3000 Bisync Driver

CAUTION

HEWLETT-PACKARD DOES NOT SUPPORT DIRECT CALLS BY USER SOFTWARE TO ANY DS/1000-IV COMMUNICATION DRIVER. SEVERE DISRUPTION OF NETWORK COMMUNICATION MAY RESULT.

DVA66 is the driver for the HDLC PSI cards for 1000-1000 communications and the BISYNC PSI cards for 1000-3000 communications. In the A/L-Series ID.66 drives the HDLC PSI cards and BISYNC PSI cards.

DVA66 – HDLC "Link Connect"Processing

On boot-up, DVA66 is entered when DINIT gets to the "ENABLE LU's" section. DINIT sends a connect request to the driver. The driver disables interrupts from the card. The card, responding to the disabled interrupts request, notes that there has never been communication between the card and driver and therefore needs initialization information from the driver. The card then sends to the driver an indication that it has just "powered up". The driver then realizes the card needs initialization information from the driver and sends a "power-up acknowledge" command to the card followed by a copy of the EQT time-out value.

Now the driver goes back to the original request which was the connect request. The "connect" command is issued to the card followed by a buffer "frame" size request. The card returns the frame size, based on a switch setting on the card. A switch indicates either 128-byte frames or 1024-byte frames. The driver stores the frame size in the EQT. The driver then enables unsolicited interrupts on the card and returns a completion to DINIT.

When the connect command is issued to the card it begins sending SARM (Set Asynchronous Receive Mode) frames to the other end of the link. If the other end is initialized it will be waiting for the SARM and respond with a UA (Unnumbered Acknowledge) frame. Once both cards on the link have exchanged SARM/UA's, the link is considered connected. The cards generate a connect interrupt to their respective drivers.

The driver formulates a zero-length message and schedules QUEUE to pass the message back to GRPM. GRPM detects the zero-length message and calls the rerouting routines to process the newly-available link. This completes link connection processing.

DVA66 – HDLC Retry Processing

The retry processing is performed by the HDLC card. The retry limit is determined by the formula:

Line timeout / T1 timeout = Retry count

The timer T1 defines the period of time the sending card will wait for acknowledgement before retransmitting the frame. The default values of T1 are used when the line timeout is set to zero. The values are shown below:

Line speed	T1 (large frames)	T1 (small frames)	Retrys
230K bps	.15 sec	.10 sec	10
56K bps	.50 sec	.15 sec	10
19.2K bps	1.50 sec	.20 sec	10
9600 bps	3.00 sec	.40 sec	10
4800 bps	6.00 sec	.70 sec	10
2400 bps	12.00 sec	1.40 sec	10
1200 bps	24.00 sec	2.80 sec	10
300 bps	96.00 sec	11.20 sec	10

The maximum number of retries used by the HDLC card is 10. The card determines the timeout value based on the setting of switches on the card indicating the line speed. When the line timeout value in the EQT is non-zero the card performs the above calculation to make the determination of retries. When ever the result of the calculation is less than 1 or equal to zero the default of 10 retries is used. A sample calculation where the EQT timeout is 200 milliseconds and the T1 timeout is 100 milliseconds (.10 sec) the calculation would be:

200 / 100 = 2 retries

Control over the retry count is via the EQT timeout value.

NOTE

When using modems that supply the clock, the line speed must be set to a value equal to or less than the modem's speed. For situations where there are long transmission delays (for example, satellite communication) the speed that the card is set to may need to be slower than the modem's so that the T1 time is long enough to compensate for the transmission delay.

DVA66 – HDLC "Link Down"Processing

When the HDLC card exceeds the maximum retry limit on a frame the link is declared down. The driver is interrupted and informed of the failure.

A message with length of one is returned to GRPM to inform rerouting of the down link. When the link failure occurs the card resets itself, discarding all other pending frames. Once rerouting has changed the path to the destination, these messages will be regenerated by the message accounting routines if the feature is generated in.

DVA66 – Read and Write Processing

When a message is queued to the driver, the driver examines size of the message to determine how many frames will be required on the card to transmit the message. The driver then asks the card for the necessary number of frames. The driver then sends the message to the card one frame at a time. If the complete message won't fit on the card the driver sends as many as the card has room for, sets a time out and suspends until the card has room for more frames.

The first frame of the message includes a "Start of Message" flag and two length words indicating the lengths of the request header and data portion of the message.

When the first frame is received by the card on the receiving end the card interrupts its driver and indicates it has received a frame with the start of message bit set. The receiving driver then reads the two length words from the frame and schedules QUEUE to provide a class read buffer to receive the message. QUEUE issues a class read request using either GRPM's class or PROGL's if this is a cold-load request or VCPMN's if this is a remote front panel request. QUEX's class is used on a BISYNC link to an HP 3000. The driver begins reading the frames off the card as they become available. When the complete message is put back together again the read completion status is returned to GRPM.

Coverage of the HDLC protocol may be found in the hardware manuals describing the HDLC cards.

DVA66-DSLIN Processing

If you run DSLIN in primary mode (initiate communication with the HP 3000), DSLIN sends a initialization request to the HP 3000. When the HP 3000 sends an initialization reply back, the reply is processed by QUEX. QUEX then sets the "BUFFER SIZE" field in the HP 3000 LU Table and sets the CR (Continuation Record) field to the appropriate value (0 if the HP 3000 does not support Continuation Records, 1 if it does).

If you run DSLIN in secondary mode (wait for the HP 3000 to initiate communication), the HP 1000 waits for the DSLINE command to be issued at the HP 3000. DSLINE causes an initialization request to be sent to the HP 1000. QUEX receives the request and sends an initialization reply to the HP 3000. QUEX then updates the HP 3000 LU Table as above.

If you run DSLIN in primary mode, the line reverts to secondary mode after the last user terminates 1000-3000 communication. To close the line, you must run DSLIN and specify the close parameter.

For more information on DSLIN operation, refer to the DS/1000-IV User's Manual, Chapter 3.

DVA65 – 1000/1000 Communications Interface Driver

****
* CAUTION *
****

HEWLETT-PACKARD DOES NOT SUPPORT DIRECT CALLS BY USER SOFTWARE TO ANY DS/1000-IV COMMUNICATION DRIVER. SEVERE DISRUPTION OF NETWORK COMMUNICATION MAY RESULT.

DVA65 is the HP 1000-to-HP 1000 communications interface driver for the 12771 hardwire-serial and the 12773 modem interface cards. It establishes the communications line link-level protocol for communication to DS/1000 nodes (91740). DVA65 and the HP 12771/12773 cards can also be used between DS/1000-IV nodes if desired however, the recommended method is to use DVA66 and the intelligent HDLC interface cards. If the driver at the other end of the line is at the same time also initiating a request by sending a Request Coming (RC), a "simultaneous request" is said to occur. The driver detects this when response to its own RC is also an RC. This situation is resolved by having DVA65 always post the last successful communications operation for a link in the EQT. When a simultaneous request is detected, the driver on the side which last received any message gains priority and is allowed to send its message, resending the RC to start over. The side which last sent data returns a "simultaneous request" error status to GRPM, which sets up a retry after a delay.

When the receiving computer successfully recognizes an RC, it returns a Transmit Next Word (TNW). The transmitting side then sends the data length which the receiver echoes. If an error occurs, the transmitting side goes back to sending the RC, up to seven times. The length echoed by the receiving driver is checked, and if it verifies, the transmitting side sends the request length, which is also echoed by the receiver and checked by the transmitter. The receiving driver now schedules QUEUE and passes the two lengths in the scheduling parameters. If QUEUE is busy, the receiving driver sends a STOP, which is treated as a "remote busy" on the initiating side. When QUEUE is scheduled to read the request, it issues a Class I/O request to the driver. This request causes buffers in SAM to be allocated and passed to the driver. (Since the buffers are in SAM, which is in the system map, the microcoded interrupt handling section never has to be concerned with setting up and enabling the user map for its data transfer operations.) The receiving driver now echoes the request length and includes a flag in bit 15 to indicate if the receiver runs in open or closed loop (0/1). If the length echo is valid, the originating CPU sends a TNW and initiates its microcode to do the block transfer. When the receiver gets the TNW, it responds with a

TNW and also initiates its microcode to do the block read.

Data transmission is handled by the microcoded portion of the driver and is described below.

The sending node begins transmitting when it receives the other side's TNW. After completion of the data transfer, the receiver sends the calculated vertical and diagonal parity words. The transmitting side compares these to those it calculates. If they both verify, then the transmitting side so indicates by sending a TNW. Otherwise, it sends a Re-transmit Last Message or "STOP" if all retries have been exhausted.

Transfers may occur in either open- or closed-loop. Open-loop means that the transmitting side simply sends the data. Closed-loop means that the receiving side is expected to respond to each word as it is received. Open-loop is normally used with the modem interface (12773). The transmitting-side interface card is placed in "transmit mode" and data is sent without a word-by-word response from the receiver. Closed-loop operation is normally used with the hardwired interface. In this mode, both I/O interface cards are placed in "receive mode". Each word sent is returned by the receiving computer to initiate transfer of the next data word.

In either mode, the block-check code calculation and exchange is the same. When the last data word is sent, the transmitting card is placed in "receive mode". Upon receiving the last data word, the receiving side driver replies with the 1's complement of the vertical parity word. The transmitting side stores this value, responding with TNW. The receiving side responds with the 1's complement of the diagonal-parity word it calculated. The transmitting side now compares both words with the values it calculated. If they are correct, the transmitting side acknowledges with a TNW. If the receiving side driver has detected a parity, data-overrun, or other error at any time during reception, the vertical and diagonal-parity words are not complemented, so the transmitting side detects this condition as a parity error.

If a message block-check failure (parity) occurs and all retries have been attempted, the transmitter sends a STOP to indicate the message is aborted. If a transmission retry is possible, the transmitter sends a Retransmit Last Message (RLM) and waits until the receiver has sent a TNW. The RLM and STOP words are normally sent only when the final TNW is expected, so they are handled at the end of block transfer. If, for some reason, one side is in the middle of a block transfer when the other side sends one of these control words, it will time-out and return the timeout error as its completion status.

Table 5-2 shows the codes for the special protocol words and Figure 5-5 shows the line protocol. Note that these code words have meaning only at certain points in the protocol and that data transmission is transparent.

Table 5-2. DVA65 Control Word Codes

TNW	170360
RLW	7417
RLM	170377
STOP	7760
RC	170017

```
 ---------------------------------------------------------------------
| Transmitting side                           Receiving side          |
|                                                                     |
| RC (Request Coming)       ----------->                              |
|                          <-----------        TNW                     |
| <data length>             ----------->                              |
|                          <-----------        <data length> is echoed |
| <request length>          ----------->                              |
|                                                                     |
|             <delay while QUEUE is scheduled>                        |
|                                                                     |
|                          <-----------        <request length> is echoed |
|                                              (bit 15 = 1 if closed loop |
|                                              is used) or STOP if QUEUE |
|                                              cannot be scheduled     |
| TNW or STOP               ----------->                              |
|                                                                     |
|------- (R) ------------------------------------ (R) ---------------|
|                                                                     |
|                          <-----------        TNW                     |
| data                      ----------->                              |
|                          <-----------        1's complement of the   |
|                                              cumulating vertical parity |
|                                              word                    |
| transfer (repeated        ----------->                              |
| as many times as         <-----------        1's complement of the   |
| necessary)                                   cumulating vertical parity |
|                                              word                    |
| last data word            ----------->                              |
|                          <-----------        1's complement of the   |
|                                              cumulating vertical parity |
|                                              word                    |
| TNW                       ----------->                              |
|                          <-----------        1's complement of diagonal |
|                                              parity word             |
| TNW (sent if              ----------->                              |
| transfer is good)                                                   |
| If transfer must be                                                 |
| repeated:                                                           |
| RLM                       ----------->                              |
|                          <-----------        TNW                     |
|                                                                     |
|        (both sides resume at the state labeled (R)                  |
|                                                                     |
| Any "hit" or detected error in the protocol causes both sides to    |
| go back to Request Coming.                                          |
 ---------------------------------------------------------------------
```

Figure 5-5. DVA65 Line Protocol

For hardwire serial interfaces only, which are installed in select codes numbered higher than the privileged-interrupt board, all protocol (TNW, RC, etc.) exchanges are done by the driver continuator section. In all other cases, the I/O exchanges are done by the microcode, but a delay is incurred of one time-out, approximately ten milliseconds on the average, in order to transfer control to the continuator section. Throughput is usually improved by installing the serial interfaces in lower-priority I/O slots than the privileged I/O board (i.e., use the driver in a non-privileged mode). Since the portion of the driver which sends data is microcoded, DVA65 can operate in either mode. Modem interfaces may be used non-privileged also, but you must use an odd numbered subchannel assignment and they will run at half-speed.

DVA65 maintains counts of the number of transmissions it has sent (exclusive of retries) and the number of re-transmissions in the EQT extension (see Appendix C for details). These counters allow line statistics to be calculated. They are simply used as 16-bit counters and allowed to roll over without indication. That is, if bit 15 is set, then the value indicated is not negative, but rather 32,768 plus the value in the low 15 bits. The following FORTRAN example illustrates how to convert a value (here, IWORD) from the EQT extension and convert it into a count (here, COUNT):

```
COUNT = IAND(IWORD, 77777B)
IF ( IWORD .LT. 0) COUNT = COUNT + 32768.0
```

The EQT extension address is kept in the EQT. The EQT can be found from the Device Reference Table, given the LU. See the RTE Programmer's Reference Manual for details on the EQT and DRT tables. See Appendix C for information specific to DVA65's use of the EQT and extension. Communication between an RTE and an HP 3000 is similar in many ways to RTE-to-RTE communication. A block diagram of the software organization is shown in Figure 5-6. Master-side communication is shown on the left. Slave-side communication paths are shown on the right. This is a broadly drawn diagram; more detail is provided later for the "communications access software" block and the master and slave sides. The block in the center, "Communications Access Software", represents modules which perform I/O to the 3000, including the driver.

```
<-------- MASTER SIDE --------> <---------- SLAVE SIDE ---------->
                                                   ---------
      Request to DS/3000              DS/3000 Reply  |  Reply  |
     ---------------------           ---------------|Converter|
    |                    |        |                  |  RPCNV  |
    |                    |        |                   ---------
    |                    V        V                       ^
    |                 -------------------                 | DS/1000
    |                | Communications    |                |  Reply
 -------             | Access Software   |          -------------
|Master |            |                   |         |   DS/1000   |
|Program|            | Modules:          |         |    Slave    |
|       |            | QUEX, QUEZ, QUEUE,|         |   Monitor   |
|       |            | QCLM and          |          -------------
 -------             | DVA66 (BISYNC) or |                ^
    ^                | CSV66/CXL66 (X.25)|                |
    |                | or DVG67 (HSI)    |                |
    |                 -------------------                 | DS/1000
    |                    |         |                      |  Request
    |                    |         |                   ---------
    | Reply from DS/3000 |         |                  | Request |
     ---------------------          ----------------->|Converter|
                                   Request in         |  RQCNV  |
                                   DS/3000 format      ---------
```

Figure 5-6. Overview of DS/1000-3000 Communications

HP 1000 to HP 3000 communications can take place over MODEM, Direct Connect, or Hardwire Serial Interface (HSI) connections using Bisync or X.25 protocols. Programmable Serial Interface (PSI) cards are used for MODEM and Direct Connect Bisync as well as X.25. Table 5-3 shows which PSI cards are used for the various connections. The 12889A interface is used for the HSI link. HSI and PSI links cannot be active in the same 1000 at the same time.

Table 5-3. PSI Communication Interfaces

	M/E/F IO	A/L-Series IO
Bisync MODEM	12793B	12073A
Bisync Direct Connect	12834A	12082A
X.25 MODEM	12250A	12075A

The PSI and HSI links use different versions of the communications access software. For HSI links, the HSI versions of QUEX and QUEZ must be active and the driver DVG67 must be associated with the LU connected to the 12889 card. There can only be one HSI link to an HP 3000 active from an HP 1000.

For PSI links, the PSI versions of QUEX, QUEZ and QUEUE must be available. Bisync links to 3000s use driver DVA66 (the same driver used for 1000 to 1000 HDLC communication). X.25 links use the customizing subroutine CSV66 attached to the X.25 driver DVX00.REL (RTE-6/VM systems), or the customizing subroutine CXL66 attached to the X.25 driver DDX00.REL (RTE-A systems), just as in 1000 to 1000 X.25 communication. Multiple Bisync links can co-exist with X.25 links.

Master programs write requests to QUEX's class (HSI) or directly to the communications LU (PSI), then wait for a reply on their own I/O class. When the reply comes back from the 3000, QUEUE receives it and rethreads it to QUEX's class number (PSI) or QUEZ informs QUEX that there is an incoming message (HSI). The reply is then matched to the proper TCB using the sequence number and back to the master program using the class number from the TCB. All requests and replies are in DS/3000 format.

When a request for a slave monitor arrives from the 3000, it is in DS/3000 format. QUEUE passes the request to the request converter (RQCNV) which converts it to DS/1000-IV format and passes it to the proper slave monitor. When the monitor completes its processing, it checks the "3000 bit" in the stream word and, when it is set, passes the reply to RPCNV, the reply converter. (Otherwise the reply goes to GRPM, as explained earlier in this chapter.) RPCNV converts the reply to DS/3000 format and sends it to the 3000.

DS/3000 Messages

All communication messages which go to and come from a 3000 are in DS/3000 format. A complete list of DS/3000 messages are included in Appendix B. Each message has a class and stream associated with it; most of these are equivalent to a DS/1000-IV stream.

DS/3000's message class 0 is a special case. Stream 20 on class 0, the initialization request, establishes the buffer size to be used over the line. Stream 21, the termination request, indicates that the sending computer no longer has any master users on the link. If the receiving computer also has no master users, communications between the two is terminated.

Because various resources must be allocated at initialization of the link, both computers must agree on the buffer size to be used. On the 1000, the size is set by choosing one of three libraries ($D3KRB, $D3KBB, or $D3KMB). These libraries allow 304, 1072, or 4096 words. The 3000 gets its buffer size from its I/O configuration or from a parameter in the DSLINE system command. The value must be a multiple of 16 which is less than 4096.

The side sending the initialization request puts its buffer size divided by 16 in the fourth word of the initialization request. If

Otherwise it puts its buffer size divided by 16 in the reply and the first system must not exceed it.

DS/3000 messages sent over Bisync and HSI lines can be larger than the communications buffer, but for larger messages continuation records must be used. Over X.25, the X.25 software takes care of segmenting the messages, so it does not need to be done by DS/1000-IV or DS/3000 software. (On X.25 links, the choice of buffer library controls only the total amount of data which can be sent to a 1000 slave program.) Figure 5-7 shows an example of a PREAD request which generates 350 words of data and is transmitted over a Bisync line configured for 304 words.

```
 ------------------------------------------------
|                                                |
|   Master Program              Slave Program    |
|                                                |
|    ------------                                |
|   | PREAD      |                              |
|   | Request    | ------->                     |
|   |            |                              |
|   |            |                              |
|    ------------                                |
|                              ------------      |
|                             | ACCEPT     |     |
|                    <------- | Reply      |     |
|                             | with con-  |     |
|                             | tinuation  |     |
|                             | bit set    |     |
|                             |            |     |
|                             | 271 words  |     |
|                             | of data    |     |
|                              ------------      |
|                                                |
|    ------------                                |
|   | Req. Header| ------->                      |
|   | w/cont. bit|                               |
|    ------------                                |
|                              ------------      |
|                             | ACCEPT     |     |
|                    <------- | Reply      |     |
|                             |            |     |
|                             | 79 words   |     |
|                             | of data    |     |
|                              ------------      |
|                                                |
 ------------------------------------------------
```

Figure 5-7. Continuation Data Buffer

In this example, the master sends a PREAD request containing DS/3000 header and appendage. The slave sends an ACCEPT reply with header, appendage, and data. Because the entire message cannot fit in the 304-word line buffer, the continuation bit in the stream word of the header is set. When the master receives the reply, it knows to signal the slave for more data by sending just an 8-word header ACCEPT request with the continuation bit set. After the slave receives this continuation request, it sends the rest of the data in an ACCEPT reply. The master knows the transaction is complete when the continuation bit is not set.

DS/3000 Bisync Protocol

DS/3000 uses a subset of IBM's Binary Synchronous Communication (Bisync or BSC) for its communications protocol. For a complete description of Bisync see the reference in the Bibliography.

Figure 5-8 shows a typical DS/3000 exchange. At the beginning, the link between the computers is idle (no transmissions in progress from either side) and the computer on the left has a message to send. That computer bids for the line by writing the Enquiry (ENQ) character. The computer on the right signals that it is ready to receive by responding with an Acknowledgement (ACK0). From that point, the computers exchange messages in "write conversational" mode: the computers send text to one another without intervening acknowledgement characters. (Each block contains the standard "low level" Bisync protocol characters -- STX and ETX -- and the CRC parity check word.)

```
                            [Line is idle]

  Bid for line      ENQ      ------->

                                  <-------     ACK0    Acknowledge bid

  Send message      Text     ------->

                                  <-------     Text    Reply with
                                  .                     message
                                  .
                                  .
                    Text     ------->

                                  <-------     Text

  End of            EOT      ------->
   transmission
                            [Line is idle]
```

Figure 5-8. DS/3000 BISYNC Exchange

If a computer does not have a DS/3000 message to send, its text consists of a null message: one word with all bits on. When a computer has nothing to send after receiving a null message in response to a null message, it terminates the text exchange by sending an End of Transmission (EOT) character. (The side sending the EOT does not have to be the same side that sent the ENQ.)

DVG67 and the HSI Link

This section describes the modules which go in the "communications access software" block in Figure 5-6 for the hardwired serial interface (HSI) link to DS/3000. Included are QUEX, QUEZ, and DVG67. The relationship between these three modules is shown in Figure 5-9.

```
 ---------------------------------------------
|                    DS/3000 Requests and       |
|                     Replies (to 3000)         |
|                           |                   |
|                           V                   |
|                      ----------               |
|        Zero-length   |  QUEX's  |             |
|        ------------->|I/O  Class|             |
|        |              ----------              |
|        |                   |                  |
|    --------                V                  |
|   |  QUEZ  |          ---------------         |
|    --------          |               |        |
|        ^             |     QUEX      |        |
|        :             |               |        |
|        :              ---------------         |
|   Unlock:                 ^        |          |
|    R. N.:      I/O        |        |          |
|        :       requests|          | DS/3000  |
|        :                V          | Requests |
|        ----------------            | and Replies |
|       |                |           | (from 3000) |
|       |     DVG67      |           |          |
|       |                |           |          |
|        ----------------            |          |
|                ^                   |          |
|                |                   |          |
|                V                   |          |
|           -----------              |          |
| (To HP 3000) |12889 Board|          |          |
|           -----------              |          |
|                                    V          |
 ---------------------------------------------
```

Figure 5-9. Communications Access Software for HSI

DVG67 I/O Requests

CAUTION

> HEWLETT-PACKARD DOES NOT SUPPORT DIRECT CALLS BY USER SOFTWARE TO DVG67. SEVERE DISRUPTION OF NETWORK COMMUNICATION MAY RESULT IF THIS WARNING IS NOT HEEDED.

DVG67 is a "state driven" driver; an event causes the driver to go into a certain state, and that state determines what the driver's next action will be.

When DS is first started, DVG67 is said to be in the "uninitialized state." After initialization and after a Line Close, DVG67 is in the "unopened state."

When a computer is receiving data blocks, it is said to be in the "receive state." This is a broad term because a station must also send messages -- such as acknowledgements -- while in the receive state. Similarly, when a station is sending data blocks it is said to be in the "send state," even though it is also receiving messages.

When QUEX has made a Write Reset request from the send state (this causes an automatic Read Inquiry request) and DVG67 ACTUALLY RECEIVES an ENQ, the station is said to be in the "Restricted Read State."

When DVG67 has received the DLE EOT message, indicating that the remote station is disconnecting, the station is said to be in the "Disconnect State."

When the Line has been opened but the station is not in the send or receive state, the station is said to be in the "Control State."

In each of the states that a station can be in -- uninitialized, unopened, control, send, disconnect, and receive -- only certain control, read, and write requests are valid. Requests incompatible with the line state complete immediately with a completion status of 2. The following gives a list of the requests that are allowed in each state:

Uninitialized

Control -- Clear
Control -- Initialize

Unopened

Control -- Clear
Control -- Initialize
Control -- Line Open

Control

Control -- all DVG67 functions except Line Open
Read Inquiry
Read Initial
Write Inquiry
Write Reset
Write Disconnect

Receive

Control -- all DVG67 functions except Line Open
Write Conversational
Write Reset
Write Disconnect

Send

Control -- all DVG67 functions except Line Open
Write Conversational
Write Reset
Write Disconnect

Restricted Read

Control -- all DVG67 functions except Line Open
Write Reset
Write Disconnect

Disconnect

Control -- Line Close

These functions are shown in Table 5-4.

Table 5-4. DVG67 Functions

FUNCTION	MEANING
Read Inquiry	a. Signals DVG67 to unlock the indicated Resource Number when an Inquiry is received. b. Requires no buffer.
Read Initial	a. Reads the remote station's line bid (Inquiry). b. Sends positive acknowledgement to the remote station. c. Reads a block of text into the specified buffer.
Write Inquiry	a. Sends a line bid (Inquiry) to the remote station to gain use of the line. b. Receives the acknowledgement from the remote station.
Write Conversational	a. Sends a block of text (contained in the specified write buffer) to the remote station. b. Receives a block of text from the remote station and puts it into the specified read buffer. c. Gives QUEX a transmission log that reflects the number of text characters received. However, if an acknowledgement is returned instead of a block, it is reported to QUEX as a zero character count.

NOTE

For Write Conversational, the buffer described in the EXEC call will be a combination of the write buffer and the read buffer. Each of these buffers will follow this standard buffer format:

Let the length of the write buffer = X and let the length of the read buffer = Y. The length of the combined buffer will by X + Y + 1. The first word of the combined buffer will contain -X, the negative length of the write buffer. The write buffer will come next, in word 2 through word X + 1. The read buffer follows the write buffer, in word X + 2 through word X + Y + 1. The length of the read buffer will be the negative length specified in the EXEC write request.

Table 5-4. DVG67 Functions (continued).

```
                    The buffer used for a Write Conversational
                    request:
 Word Number
 -----------        ------------------------------------------
      1             -X (the negative length of write buffer)
                    ------------------------------------------
      2                                                     ^
      .             standard write buffer                   |
      .                                                  X Words
      .                                                     |
    X + 1                                                   v
                    ------------------------------------------
    X + 2                                                   ^
      .             standard read buffer                    |
      .                                                  Y Words
      .                                                     |
  X + Y + 1                                                 v
                    ------------------------------------------
```

Write Reset	a. Sends an EOT (End-of-Transmission) to the remote station to relinquish use of the line. b. In some situations, receives a message from the remote that can be EOT, DLE EOT, or ENQ. c. Requires no buffer.
Write Disconnect	a. Informs the remote station that the local station is plannning to disconnect the line by sending the DLE EOT message. b. Requires no buffer.
Clear	a. Disables the interrupts on the synchronous interface board and marks device as unavailable.
Initialize	a. Builds board-dependent control words with fundamental device parameters for the driver.

Table 5-4. DVG67 Functions (continued)

FUNCTION	MEANING
Line Open	a. Readies the line for read/write operations by establishing the initial state of: - interface board - communications line b. Gives the default value of 7 to the number of retries and the default value of 60 seconds to the "Long Timeout."
Line Close	a. Disconnects the line and disables it for further read/write operations. Hence, line close is the inverse of line open. b. Clears "Data Terminal Ready."

A complete message exchange is defined as the sending and receiving of those messages necessary to:

a. Put the station into the send state or receive state.

b. Do as much sending or receiving as is necessary while in the send or receive state.

c. Get the station out of the send or receive state.

The Read/Write requests each handle the sending and/or receiving of one or more messages. Hence, a complete message exchange can be specified by a particular sequence of Read/Write requests.

DVG67 supports the contention mode of BSC (as opposed to the supervised mode, when polling and addressing are done on multipoint lines) and the sequence of requests for typical reads and writes will be described with this in mind.

When QUEX has a message to send to the 3000, it must bid for use of the line and simultaneously put the station into the send state. It does this by making the Write Inquiry request. Next it is ready to send data blocks, using a Write Conversational request to send each block.

QUEX operates in the conversational mode, in which each station responds to the receipt of one block with the sending of another block. Note that when text is received, the station changes from the send state to the receive state.

Example 1. Send one block of data. (Figure 5-10 shows the corresponding message exchange. The RTE is on the left, the MPE is on the right.) Three write requests specify the following functions (the remote station is assumed to have no data to send):

Write Inquiry
Write Conversational
Write Conversational

Remember, the "Null Message" is a one-word text block that is sent when a station has no DS message to send. When a station receives a null message in response to a null message, and has nothing to send, it sends an EOT.

```
 -------------------------------------------------------------------
|                                                                   |
| Station in Send State                   Station In Receive State  |
| ---------------------                   ------------------------  |
|                                                                   |
| Write Inquiry    ENQ ------->                                     |
|                          <------- ACK0       Read Initial         |
|                                                                   |
| Write            Text ------->                                    |
|  Conversational                                                   |
|                          <------- Null       Write                |
|                                   Message     Conversational      |
|                                                                   |
| Write            Null ------->                                    |
|  Conversational  Message                                          |
|                          <------- EOT        Write Reset          |
|                                                                   |
 -------------------------------------------------------------------
```

Figure 5-10. DVG67 Message Exchange with One Block Sent

Example 2. QUEX sends two blocks of data and receives one block of data in the conversational mode (Figure 5-11 shows the corresponding message exchange). It makes five write requests, specifying the following functions:

Write Inquiry
Write Conversational
Write Conversational
Write Conversational
Write Reset

```
 -------------------------------------------------------------------
|                                                                   |
|  Station in Send State                Station in Receive State   |
|  ---------------------                ------------------------   |
|                                                                   |
|  Write Inquiry     ENQ -------->                                  |
|                              <------- ACK0     Read Initial        |
|                                                                   |
|  Write             Text -------->                                 |
|   Conversational                                                  |
|                              <------- Null     Write               |
|                                       Message   Conversational    |
|                                                                   |
|  Write             Text -------->                                 |
|   Conversational                                                  |
|                              <------- Text     Write               |
|                                                 Conversational    |
|                                                                   |
|  Write             Null -------->                                 |
|   Conversational  Message                                         |
|                              <------- Null     Write               |
|                                       Message   Conversational    |
|                                                                   |
|  Write Reset       EOT  -------->                                 |
|                                                                   |
 -------------------------------------------------------------------
```

Figure 5-11. DVG67 Message Exchange with Write Conversational

When the Write Reset request follows a Write Conversational request (hence, DVG67 is in the send state), DVG67 will automatically perform a Read after sending the EOT. If nothing is received within three seconds, DVG67 will declare the Write Reset request complete, with normal completion status. However, if an ENQ or EOT (disconnect) message is received, DVG67 will complete the request and indicate the message received in the completion status. This procedure gives the remote station a chance to send or say it does not want to send.

On the other hand, the Write Reset request made in the Control State or the Receive State is not followed by an automatic receive. When the system is sending and DVG67 detects an error completion, it will put the station into the control state and make a Write Reset request to reset the condition of the remote station before attempting to send another message.

When the HP 1000 is the receiving side instead of the sending side, QUEX must receive the HP 3000's line bid, put the station into the receive state, and acknowledge the ENQ.

To do this QUEX schedules QUEZ without wait. QUEZ makes a Read Inquiry request to DVG67, then waits on a resource number. The Read Inquiry signals DVG67 to unlock that resource number when an ENQ arrives from the 3000. The unlock reschedules QUEZ, who writes a zero-length message to QUEX's I/O class and terminates.

When QUEX receives the zero-length message from QUEZ, it makes a Read Initial call to send the ACK0 acknowledgement and read text from the 3000.

Thereafter, QUEX makes Write Conversational requests in order to receive additional data blocks. QUEX makes a Write Reset request to send the end-of-transmission (EOT) message. When this occurs, the remote station is given the "EOT received" completion status (and no data) and the station is put back into the control state.

Example 3. QUEX receives one block of data. (Figure 5-10 shows the corresponding message exchange. The RTE is on the right, the MPE is on the left.) QUEX makes a read request, specifying the following functions:

Read Initial
Write Conversational
Write Reset

Once in the receive state, QUEX goes into the conversational mode to send data blocks as responses by making a Write Conversational request.

Example 4. QUEX receives two blocks of data and sends one block of data in response. (Figure 5-11 shows the corresponding message exchange.) It makes one read and three write requests, specifying the following functions:

Read Initial
Write Conversational
Write Conversational
Write Conversational

In all these examples, the text blocks sent between the computers could contain several DS/3000 messages. After a block of text arrives, QUEX has the responsibility to separate and deliver the individual messages.

All communication requests are made without wait. The RTE node is always the primary station, and the HP 3000 is the secondary.

QUEX and QUEZ for the HSI Link

In the HSI link, QUEX has all the responsibility for sending and receiving messages from the HP 3000. It also must know the current state of the communications line and direct the driver to send ENQs, EOTs, and null messages at the appropriate time.

Figure 5-12 shows the initialization sequence performed when QUEX is first scheduled by DINIT, when QUEX receives a "disconnect" from the HP 3000, when QUEX is rescheduled by UPLIN, and when a "DS/3000 COMMUNICATION LINK *DOWN*" message is printed.

```
If DS is initialized for PSI link to 3000, print error message
and suspend (go to program state 6).

Initialize DVG67 with Clear and Line Open calls

Open line with Write Inquiry. Receive ACK0.

Send DS/3000 Initialization request.

Get DS/3000 Initialization reply. Post buffer size in 3000 LU
Table and clear sign bit in QUEX's class number.

Display "*UP*" message.

Wait for messages on I/O class. (Could be a 1000 user or QUEZ's
zero-length message to signal ENQ received from 3000.)
```

Figure 5-12. QUEX's Initialization for HSI Link

When using the HSI (hardwire) link the RTE side is always the first system to send the initialization request (message class 0, stream 20) and is designated as the primary. The MPE is the secondary and must back off when ENQ contention occurs. When QUEX sends the initialization request, it sets the "primary" bit in the SLC EQT extension. Thus, the driver will not return an ENQ contention error when that situation occurs.

QUEX must receive a reply to the initialization request before it will release any outgoing messages. This blocking action is accomplished by setting the buffer length to -1 in the 3000 LU Table. The master module which write to QUEX's class, D3KMS, detects this and returns a "line disconnected" (-56) error. When the initialization reply arrives, QUEX clears the bit in #QXCL, prints the "* UP *" message on the system console, and is open for business.

Whenever the line is idle, QUEX schedules QUEZ (to check for an ENQ arriving from the 3000) then waits on its own I/O class (to receive a master request, a slave reply, or QUEZ's zero-length signal).

When master requests and slave replies arrive on its class, QUEX gathers the outgoing messages into one large buffer (up to the maximum communications buffer size) and sends it to the HP 3000 in a Write Conversational request. (QUEX will first bid for the line with a Write Inquiry, if necessary.) If any DS/3000 messages arrive from the HP 3000, they are "de-blocked" and dispatched to waiting masters and/or the slave monitor request converter (RQCNV). QUEX remains in this loop as long as data remains to be sent in either direction. After an EOT is sent or received, it schedules QUEZ and goes back to its I/O class get.

If the HP 3000 has data to send while the line is idle, QUEX is brought out of its class get suspend by the zero-length message from QUEZ. QUEX reads the buffer with a Read Initial request, then goes to its de-block/dispatch/Write Conversational loop to exchange further messages with the 3000.

The HSI version of QUEX sends and receives DS/3000 messages with data passed in the "Z" buffer. A dummy "local appendage" with the LU of the HSI link is added to all incoming messages before they are dispatched to a master or request converter. If logging is enabled, the word preceding the message is the 3000 LU for outbound messages and the negative LU for inbound messages.

DVA66, CSV66, and the PSI Links

This section describes the modules which go in the "communications access software" block in Figure 5-6 for the programmable serial interface (PSI) links to DS/3000. Included are the monitors QUEX and QUEZ, and the driver DVA66. Also included are the customizing subroutines for X.25 links, CSV66 (for RTE-6/VM) and CXL66 (for RTE-A). The relationship between these modules is shown in Figure 5-13.

```
 ------------------------------------------------------------------
|                                                                  |
|                    -------     --------------                    |
|                   | RPCNV |   |              |                   |
|                   |   or  |   |     QUEX     |                   |
|                   | D3KMS |   |              |                   |
|                    -------     --------------                    |
|                       |            ^         |                    |
|                       |            |         |                    |
|                       |            |         | DS/3000            |
|                       |         -------      | Requests           |
|             DS/3000   |        | QUEUE |     | (from 3000)        |
|             Requests  |         -------      | dispatched         |
|             & Replies |            ^         | to RQCNV or        |
|             (to 3000) |            |         | replies (from      |
|                       v            |         | 3000) to 1000      |
|   --------         ------------              | masters            |
|  |        |   I/O   |            |          |                    |
|  |  QUEZ  |<--------| DVA66 or   |          |                    |
|  |        |  Status | CSV66 or   |          |                    |
|   --------  (writes)| CXL66      |          |                    |
|                      ------------            |                    |
|                           |                  |                    |
|                      (To HP 3000)            v                    |
|                                                                  |
 ------------------------------------------------------------------
```

Figure 5-13. Communications Access Software for PSI

DVA66 I/O Requests for BISYNC

CAUTION

HEWLETT-PACKARD DOES NOT SUPPORT DIRECT CALLS BY USER SOFTWARE TO DVA66. SEVERE DISRUPTION OF NETWORK COMMUNICATION MAY RESULT IF THIS WARNING IS NOT HEEDED.

The following functions are used by the PSI versions of QUEX and QUEZ to communicate with the BISYNC firmware on the board:

Initialize

Tells the card what block size will be used in transactions with the card (304, 1072, or 4096 words). The card may not be able to handle certain block sizes and can return a block size parameter to override the requested block size.

The remote and local ID sequences are sent to the card. These are used to validate connect requests either initiated from the 1000 or 3000.

Once the initialize parameters are sent to the card, DSLIN will issue the message "3000 LINK READY FOR CONNECTION" then proceed to issue the primary connect command.

Primary Connect

The primary connect request tells the card to connect as a primary station and initiate the connect sequence with the remote 3000.

Secondary Connect

This command tells the card to wait to receive a connect sequence initiated by the 3000. This command places the 1000 in secondary station mode.

Get Parameters

This command retrieves the board type, firmware revision date code, dip switch settings, diagnostic hood indicator, data block size etc. The block size is used by QUEX to determine if the "asked for" size is larger than the block size the card will accept. In this case the maximum block size is defined by the card.

Read

Issued by QUEUE to retrieve incoming HP 3000 messages.

Write

The write command is used to send a block of data to the card.

The 1000-3000 low-level communications is quite similar to the 1000-1000 modules; the driver schedules QUEUE, then QUEUE performs a class I/O read on QUEX's (rather than GRPM's) class.

For incoming messages, the card does not know where the break between the header/appendage and the data occurs. QUEUE performs a class I/O read of two buffers onto QUEX's class. (The "Z" buffer at this point is only large enough to hold the "local appendage".) The driver uses the word counts in the HP 3000 message header to modify the class I/O header in SAM to make it look like a "Z" buffer was passed containing the data and local appendage.

QUEX and QUEZ for the PSI Link

The logic for the versions of QUEX and QUEZ designed for PSI links is less complex than the HSI modules. The complicated interactions necessary to implement the protocols is handled by either the BISYNC firmware on the PSI card or the X.25 software.

Because there can be any number of PSI links, QUEX cannot initialize lines. When DINIT is run to initialize DS, all BISYNC links in the 3000 LU Table are brought up in secondary mode (waiting for call). The program DSLIN performs re-initialization as primary or secondary for a given BISYNC line. (Copies of DSLIN may be run by different users on different LUs.) Refer to the DS/1000-IV User's Manual, Chapter 3 for information on running DSLIN. No DINIT or DSLIN initialization is needed for X.25 lines connected to a 3000. These LUs will be placed in the 3000 LU Table as they are allocated, and will be deleted from the table by UPLIN when they are deallocated.

Messages bound for a 3000 over PSI links are not combined with other DS/3000 messages into a single communications buffer. All writes to a 3000 are made directly from D3KMS (for masters) or RPCNV (for slaves) to the communications LU via class I/O. QUEX does no processing for outgoing messages.

QUEUE reads incoming messages, passes them to QUEX via its class number and QUEX dispatches them to the proper master or RQCNV. The dispatching is done via a class rethread.

The PSI QUEZ waits on its I/O class number and checks the I/O status for outbound messages. (It also removes the class buffers left over from the WRITE/READ.) If the error bit is set in the I/O status, QUEZ sets the LU "down" in the 3000 LU Table, passes an error message to QCLM, and reinitializes the card to be in secondary mode.

Because QUEZ needs to know which LU the status word is associated with, a word containing the LU number is sent at the beginning of each outbound message. (This word is not used for HSI links.) The driver removes this word (and decrements the length) before the message is written to the 3000.

If logging is enabled, this LU word is written to the log device along with the message. On inbound messages, the word preceding the logged message is set to the negative of the LU.

X.25 Link

1K MASTER/3K SLAVE: An HP 1000 master program initiates communication with an HP 3000 by calling HELLO. The HELLO subroutine calls the DS/1000-IV library $D3X25 which contains a reference to the DSN/X.25 routine ALTAD. (ALTAD is contained in the DSN/X.25 library X25LB.REL.) ALTAD allocates a switched virtual circuit (SVC), and $D3X25 places the POOL LU associated with the SVC in the 3000 LU Table via a call to D$PUT. $D3X25 does a "connect" on the SVC and sends an HP 3000 initialization message to the HP 1000. The POOL LU placed in the 3000 LU Table is then marked "up" by the routine D$UP!.

If an SVC goes down before if is marked up in the 3000 LU Table by D$UP!, HELLO calls RELSX to release the circuit. RELSX releases the POOL LU by calling the DSN/X.25 routine RPOOL contained in the DSN/X.25 library X25LB.REL.

The HP 1000 terminates communication by calling BYE. The BYE routine causes an HP 3000 BYE to be issued at the HP 3000 and marks the X.25 LU "down" in the 3000 LU Table. UPLIN checks the 3000 LU Table for "down" X.25 LUs, deletes them, and calls RELSX. RELSX calls the DSN/X.25 routine RPOOL, which returns the X.25 SVC to the pool.

NOTE:

> If the DSN/X.25 connection consists of a Packet Switching Network (PSN) or a genuine modem link (not "hardwired" with baseband or short haul modems), SVCs are not automatically

reconnected if the network or modem(s) goes down and is then restarted. You must use the DI command in DSMOD to disable the LU associated with the circuit and return it to the POOL. Do NOT use the DSMOD /L command.

3K MASTER/1K SLAVE: When a line open request arrives from an HP 3000, DSN/X.25 allocates a POOL LU on behalf of the HP 1000 slave and routes the request to the DS/1000-IV request converter RQCNV. RQCNV checks the HP 1000's Transaction Status Table (TST) for an entry (i.e., a new request from the HP 3000). If there is no TST entry, RQCNV checks the 3000 LU Table for an X.25 LU. If there is an X.25 LU in the table, it is known that the HP 3000 request is not a new X.25 request and RQCNV will not allocate an LU. If there is not an X.25 LU in the table, RQCNV calls D$PUT which places the POOL LU into the 3000 LU Table. $D3X25 does a "connect" on the SVC associated with the LU and marks the LU "up" via a call to the routine D$UP!.

When it is finished, the HP 3000 disconnects the X.25 SVC and the DS/1000-IV monitor QUEZ receives a link down indication at the HP 1000. QUEZ checks the 3000 LU Table for an X.25 LU and, if it finds one, marks it "down." UPLIN checks the table for a "down" LU, deletes the entry from the table, and calls RELSX. RELSX calls the DSN/X.25 routine RPOOL which returns the X.25 SVC to the POOL.

NOTE:

Because UPLIN only runs every 5 seconds, there may be a delay between the time an X.25 SVC is disconnected and the time the LU is removed from the 3000 LU Table. If there are a limited number of POOL LUs, this delay may temporarily prevent a program from obtaining a POOL LU.

DS/1000-3000 Master Side Communication

Figure 5-14 is a block diagram of the software which delivers DS/3000 messages from master users at the 1000 to the 3000.

HELLO, BYE, and Master Subroutines

MPE requires each of its users to be operating under a specific Session Main Process (SMP). Remote users from the 1000 establish their session through the subroutine HELLO, either by calling it directly from their own program or by using the HELLO command in RMOTE.

If the HP 3000 LU parameter in HELLO is an integer less than 256, it is treated as the LU number of a BISYNC or HSI connection. Otherwise it is treated as the ASCII encoding of an X.25 address. This X.25

ASCII string may contain leading ASCII characters and up to 15 digits. The total length of the string cannot exceed 28 characters. It is terminated by any non-digit.

When an X.25 address is passed, HELLO converts the digits to Binary Coded Decimal format and calls the ALTAD subroutine (provided in the DSN/X.25 library X25LB.REL) to allocate a switched virtual circuit (SVC) POOL LU connected to the X.25 address. HELLO uses routines in the DS library $D3X25 to allocate a POOL LU, put it in the 3000 LU Table, and mark it "up." After the LU is placed in the 3000 LU table and marked "up," HELLO opens the line with wait by making a Primary Connect call. Processing continues with the normal HELLO request.

If the account and (optionally) passwords are correctly provided, MPE assigns an SMP number, which HELLO returns as an integer parameter.

It is possible for one program to establish a session, then pass the SMP number to a son program for its use. The father program must schedule its son with wait because two programs cannot access the same MPE session simultaneously. The son program gains access to the session by calling PRCNM.

Specifically, RMOTE will pass the negative of its SMP number as the fifth parameter when scheduling sons through the RU, ON, or RW commands.

DS/1000 software will kill the session if the program which issues the HELLO goes dormant before calling BYE to log off.

Once the session is established, a program can make PTOP or DS/3000 RFA subroutines calls. In PTOP calls, the 3000 is addressed by the negative of its logical unit number. No address is necessary for RFA calls.

1000 users terminate their 3000 session by calling the subroutine BYE. After closing the session, BYE sends a DS/3000 termination request if no other 1000 programs are using the line. (For X.25, that will always be the case.) BYE does not wait for a reply to the termination, but lets QUEX close the line when the reply is received. For X.25 connections, BYE marks the POOL LU "down" in the 3000 LU table. UPLIN deletes the "down" LU from the Table and the DSN/X.25 routine RPOOL returns the X.25 SVC to the pool.

If the other side is making use of the line, it rejects the request and the line stays up. Otherwise the line is disconnected (a DLE EOT is sent).

When termination is successful on a BISYNC line, QUEX sets the buffer size in the 3000 LU table to -1 to indicate the line is down.

If the PTOP POPEN call is made using an X.25 pool LU, the PTOP software can make use of the routine LU3K to find out which LU was assigned to the master program by the X.25 software. See the User's manual for the calling sequence.

```
 ------------------------------------------------------------------
|   -------               MASTER SIDE                              |
|  | RMOTE |                                                       |
|  |-------|                                                       |
|  | HELLO |                                                       |
|  |-------|                                                       |
|  | D3KMS |----------------------------------------------         |
|   -------       --------                                |        |
|      ^         | Master |                               |        |
|      |         | RFA    |                               |        |
|      |         | User   |                               |        |
|      |         |--------|                               |        |
|      |         | HELLO  |                               |        |
|      |         |--------|                               |        |
|      |         | FOPEN, |                               |        |
|      |         | etc.   |                               |        |
|      |         |--------|                               |        |
|      |         | D3KMS  |---------------------------    |        |
|      |          --------       --------             |   |        |
|      |             ^          | Master |            |   |        |
|      |             |          | PTOP   |            |   |        |
|      |             |          | User   |            |   |        |
|      |             |          |--------|            |   |        |
|      |             |          | HELLO  |            |   |        |
|      |             |          |--------|            |   |        |
|      |             |          | POPEN, |            |   |        |
|      |             |          | etc.   |            |   |        |
|      |             |          |--------|            |   |        |
|      |             |          | D3KMS  |--------    |   |        |
|      |             |           --------         |   |   |        |
|      |             |              ^             |   |   |        |
|      |             |              |             |   |   |        |
|      |             |              |             |   |   |        |
|      |             |              |             v   v   v        |
|      |             |              |             ---------------  |
|      |             |              |      -----|Communications | |
|      |             |              |     |  ---|Access Software| |
|      |             |              |     | |  -|(QUEX, QUEZ,   | |
|      |             |              |     | | | |QUEUE & Driver)| |
|   --------      --------       --------  | | |  ---------------  |
|  | Master |    | Master |     | Master | | | |         ^         |
|  | Class  |    | Class  |     | Class  | | | |         |         |
|   --------      --------       --------  | | |         v         |
|      ^             ^              ^      | | |    ---------      |
|      |             |               ------  | |    | HP 3000 |    |
|      |              -----------------------  |     ---------     |
|       ---------------------------------------                     |
 ------------------------------------------------------------------
```

Figure 5-14. DS/1000-3000 Master Side Modules

D3KMS Subroutine

The master subroutine for HP 3000 master requests is D3KMS. It is similar to #MAST. It obtains a class number for the reply and sends the request by writing the request (and data, if any) to the 3000. (See Appendix B in Volume II for request and reply buffer formats). D3KMS then makes a Class I/O Get call, which suspends the user program until the reply arrives or the request times out.

D3KMS performs the following functions not found in #MAST:

- Handles continuation requests/replies for $STDLIST/$STDIN requests from the HP 3000. (Continuation is not needed for X.25 links.) All data to and from the HP 3000 is blocked into records whose length depends upon the 3000 buffer library chosen during generation (either 304, 1072 or 4096 words) and the buffer length recorded in the 3000 LU Table. (See the example in Figure 5-7.) At link initialization time, a bit is set in the initialization request and reply to indicate each side can buffer continuation records for these messages. Buffering is done at the end of the partition in the case of RMOTE. When D3KMS is appended to the user programs, the standard 134-word buffer within D3KMS is used to collect these messages.

- Processes intervening $STDLIST/$STDIN requests from the HP 3000. These could be the result of an operator command (see Figure 5-15) or a PTOP slave program's I/O request.

- Checks for the RTE "BReak" command during $STDLIST/$STDIN data transfers, and sends a "BREAK" or "CONTROL-Y" request to the HP 3000, depending on what the user specifies.

It contains code for the following:

- The ICC function for interrogating HP 3000 condition codes.

- The PRCNM subroutine, which allows master users to access the session opened by a father program's HELLO.

- Subroutines for managing the appendage area (storing and receiving parameters) to be used by user-callable subroutines (PTOP and RFA).

- Global data words used by HELLO, D3KMS, and other master subroutines.

WARNING

Hewlett-Packard does not support the use of continuation records by user software for prints to $STDLIST or reads from $STDIN. No guarantee can be made that D3KMS has enough space to buffer these messages.

D3KMS checks the process number list each time it has a message to send. If the link has gone down (and UPLIN has flushed all the entries for that 3000) it will not find its SMP number in the list and will report a DS06 (illegal request) error. If the link goes down while D3KMS has an outstanding request, UPLIN will send it a zero-length record and D3KMS reports a DS05 (timeout) error.

D3KMS writes BISYNC and X.25 messages to the communications LU with I/O completion status sent to QUEZ's class. (HSI messages are written to QUEX's I/O class.) PSI messages have a word containing the 3000 LU number added to the beginning of the message; HSI messages do not. For both HSI and PSI, header and appendage are written in the regular data buffer while the data area is transferred in the "Z" buffer.

The example in Figure 5-14 shows how RMOTE and D3KMS work together. Suppose an operator runs RMOTE, switches to remote, signs onto an HP 3000, and sends a LISTF command. Figure 5-15 shows the exchange that takes place.

```
-----------------------------------------------------------------------
| HP 1000                                  HP 3000                    |
| --------------------------               ------------------------   |
| User types command (LISTF)                                          |
| RMOTE processes it,                                                 |
| and calls D3KMS to send                                             |
| the request to HP 3000.                                             |
|                                                                     |
| ---> D3KMS command request-------->                                 |
|                          <--------       $STDLIST Request           |
|                                                                     |
| (picked up by D3KMS                                                 |
| and printed on terminal.                                            |
| Sends reply)                                                        |
|                                                                     |
| $STDLIST reply           -------->                                  |
|                          <--------       $STDLIST request           |
|                                                                     |
| (picked up by D3KMS                                                 |
| and printed on terminal.                                            |
| Sends reply)                                                        |
|                                                                     |
| $STDLIST reply           -------->                                  |
|                                                                     |
| This sequence repeats                                               |
| until the LISTF output                                              |
| has been completely sent.                                           |
| Then the reply comes back:                                          |
|                                                                     |
|                          <--------      LISTF command reply         |
| (Picked up and processed                                            |
| by D3KMS. Because this is                                           |
| the reply to the original                                           |
| request, control returns                                            |
| to RMOTE.)                                                          |
|                                                                     |
| <--- D3KMS                                                          |
|                                                                     |
| RMOTE   now executes again.                                         |
-----------------------------------------------------------------------
```

Figure 5-15. RMOTE and D3KMS Operation

DS/1000-3000 Slave Side Communication

Figure 5-16 is a block diagram of the software which accepts DS/3000 messages from users at the 3000, converts them to DS/1000 format, presents them to the proper slave monitor, and returns the reply to the 3000.

When QUEX receives a request buffer it sends it to the Request Converter's (RQCNV) I/O Class number. RQCNV changes the request buffer from the format used by DS/3000 to that used by DS/1000-IV. It determines the proper monitor and writes the buffer onto that monitor's I/O Class number.

The same monitors are used for requests made from a HP 3000 as for requests from RTEs. Their operation is as described previously. When they complete a transaction, they call #SLAV. #SLAV's operation is similar to that described for RTE-to-RTE communication, except it threads the buffer to the Reply Converter's (RPCNV) I/O Class number for conversion to HP 3000 format. RPCNV reformats the reply buffer and sends it to the HP 3000.

CNSLM, the console monitor, prints message directed to the 1000 from the MPE operating system for TELL, WARN, and aborted session messages.

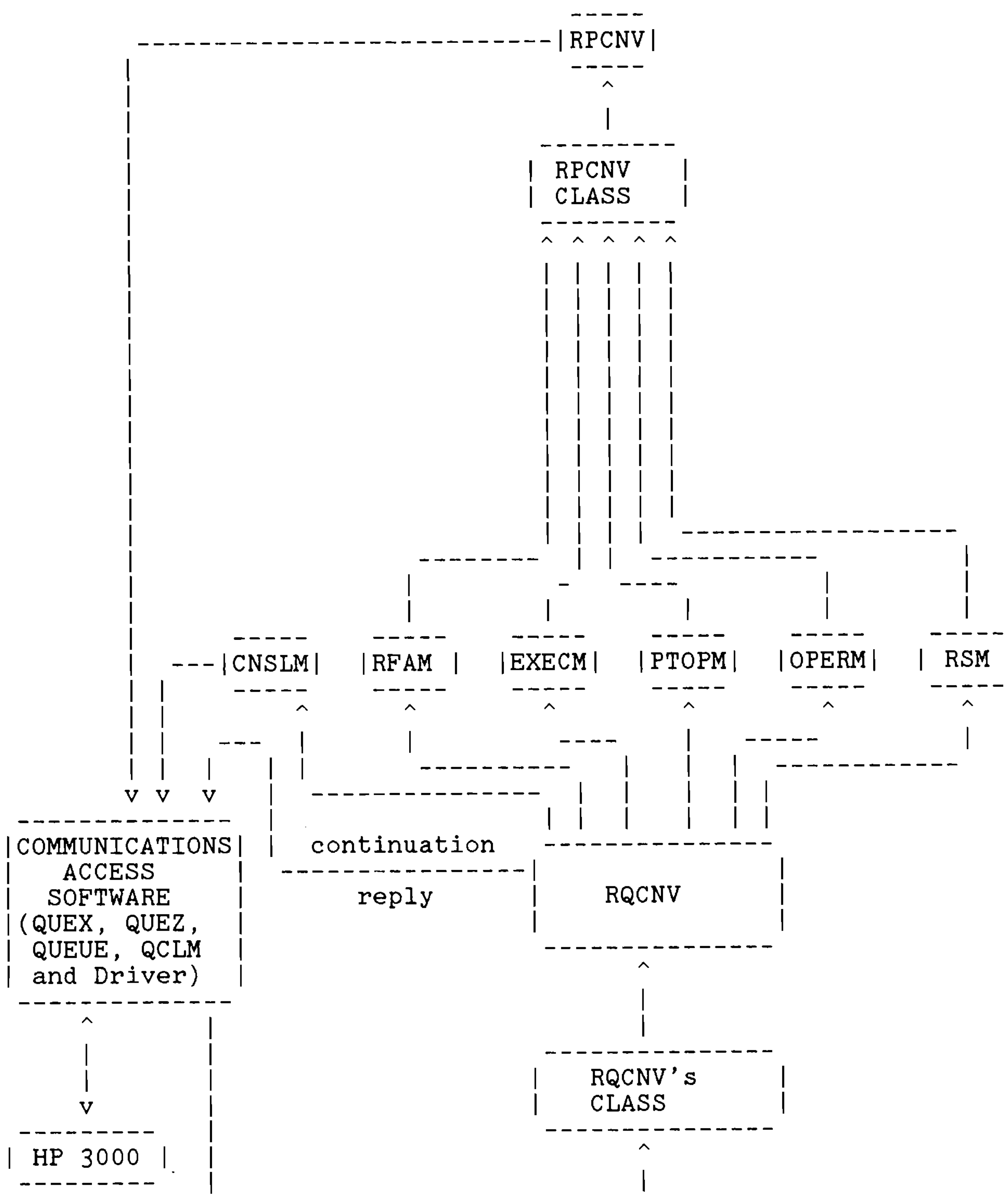


Figure 5-16. DS/1000-3000 Slave Side Modules

RQCNV and RPCNV – Request and Reply Converters

When the request converter (RQCNV) receives a DS/3000 request, it reformats the parameters into a DS/1000-IV request, allocates a slave TCB, builds a Transaction Status Table entry, and writes the request to the proper slave monitor (CNSLM, RFAM, EXECM, PTOPM, OPERM, or RSM).

The procedure is more complicated when a write request is received with its continuation bit set. RQCNV must save the data in the first request (by using an I/O class to hold it temporarily) and send a continuation reply to the 3000. When the continuation request is received from the 3000, its data is also written to the holding class. When the final continuation request (indicated by no continuation bit set) is received, RQCNV collects all the data off the holding class, deallocates it, and sends the data and its DS/1000 request to the monitor.

After the monitor has completed processing a request, it calls #SLAV. #SLAV tests the "HP 3000 Master" bit in the stream-type word. If set, indicating that the request came from an HP 3000, it sends the reply to RPCNV (the reply converter).

The reply converter takes DS/1000-IV replies, converts them to DS/3000 replies, and sends them to the 3000. RPCNV uses the information saved in the Transaction Status Table to build the DS/3000 header and to determine which HP 3000 sent the request. (RPCNV deallocates the TST entry when it is done.)

The procedure gets complicated when a read reply is too large to fit in a single communications buffer. When this happens, RPCNV fits as much data as possible in the reply, sets the continuation bit, sends the reply to the 3000, and writes the rest of the data onto a holding class. When the 3000 sends a continuation request, RQCNV passes it through to RPCNV and the next part of the data is sent. When all the data has been sent, the holding class and TST entry are deallocated.

RQCNV and RPCNV can only accommodate data that will fit in the buffer provided by the 3000 buffer library ($D3KRB: 512 words, $D3KBB: 1024 words, or $D3KMB: 4096 words). This limit is independent of the 3000 communications line buffer provided by the same libraries. For example, if the line buffer is 304 words and the RQCNV/RPCNV data buffer is 512 words, RQCNV can accept continuation buffers only if they total less less than 513 words.

It is possible to load the converters with a different library than that used for QUEX. Under no circumstances should the converters be loaded with a smaller library; they must be able to accept the first buffer sent from the 3000.

CNSLM

CNSLM receives system (as opposed to user) requests from the HP 3000 for $STDLIST (i.e., standard list output). It directs I/O requests from MPE TELL or WARN commands to LU 1 and the log-on LU for the process. Logoff messages for sessions aborted by MPE are printed on the system console.

It recognizes stream types 20 (write) and 23 (control) of class 20, rejecting all others. Processing depends on stream type as shown in Table 5-5.

Table 5-5. CNSLM Processing

STREAM TYPE	DESCRIPTION	PROCESSING
20	$STDLIST directed to terminal	If "from process number" equals zero, the message is printed on LU 1 and the LU specified by the "to process number." No reply is sent. Otherwise, an output call is used to send the message to the system console and a reply is sent.
23	FCONTROL to $STDLIST	A reply is sent to the 3000.

1000 — 3000 Clean-Up Information

If an HP 1000 program logs on to a 3000 but terminates without issuing a BYE, resources need to be cleaned up on both sides. UPLIN removes the local PNL entry and terminates the 3000 Session Main Process by issuing a KILL JOB request. When a reply to the KILL is received from the 3000, QUEX scans the PNL and sends a DS/3000 termination request if no other entries exist for the indicated link.

QUEX acts on the 3000's reply. Rejects are ignored. For PSI, accepts cause QUEX to disconnect the given line and reinitialize it in secondary mode.

For X.25 links, the disconnect from QUEX causes the associated switched virtual circuit to be returned to the POOL. (The secondary mode initialization is rejected by DSN/X.25 customizing subroutines CSV66 and CXL66.) QUEZ checks the 3000 LU Table for POOL LUs and, if it finds one, marks it "down" so UPLIN can clean up. When UPLIN finds a "down" LU in the Table, it deletes it and calls RELSX. RELSX calls the DSN/X.25 routine RPOOL which returns the SVC to the POOL.

NOTE:

Because UPLIN only runs every 5 seconds, there may be a delay between the time an X.25 SVC is disconnected and the time the POOL LU is removed from the HP 3000 LU Table. If there are a limited number of POOL LUs available, this delay may temporarily prevent a program from obtaining a POOL LU.

Appendix A
DS/1000-IV Message Formats

Request Header

All user-initiated DS/1000-IV requests and commands are passed through the system as a single class-I/O buffer, contained in two parts (the "double-buffer (Z)" bit is set when the class-I/O request is made). The first buffer contains the user data, if any. The second buffer contains the DS/1000-IV "request header".

All DS headers will have the following information:

```
              REQUEST                                          REPLY
      +-------------------------+                   +-------------------------+
    0 | Stream   Word           |        F          | Stream   Word           |
    1 | Transaction Seq Number  |        I          | Transaction Seq Number  |
    2 | Source Node #           |        X          | Source Node #           |
    3 | Destination Node #      |        E          | Destination Node #      |
    4 | ECOD1                   |        D          | ECOD1                   |
    5 | ECOD2                   |                   | ECOD2                   |
    6 | Reporting Node #        |      HEADER       |A| Reporting Node #      |
    7 |NXF|sub|  Qual| Level    |                   |XXF|sub|  Qual| Level    |
      |   |lev|      |          |                   |   |lev|      |          |
    8 | MA "send" seq. #        |        A          | MA "send" Seq. #        |
    9 | MA "recv" seq. #        |        R          | MA "recv" Seq. #        |
   10 | MA "cancel" flags       |        E          | MA "cancel" flags       |
   11 | "Loop counter"          |        A          | "Loop counter"          |
   12 |Src Sessn | Dst Sessn    |                   |Src Sessn | Dst Sessn    |
      +-------------------------+                   +-------------------------+
      | Request-specific        |                   | Reply-specific          |
      |  parameters             |                   |   parameters            |
      +-------------------------+                   +-------------------------+
      | RTRY delay value        |                   | Reserved                |
      | MA retry|S||C| LU       |                   | MA retry|S||C|  LU      |
      +-------------------------+                   +-------------------------+
      15      12 11 8 7      0                      15      12 11 8 7
```

There are three main areas of the header. The first 13 words are common to all messages, and are called the Fixed Header Area. The second area is in a format specific to the individual request, and is of variable length. The third area, called the local appendage, contains information which the local system needs to handle the request, but which has no meaning in any other node. This third area is not transmitted. #MAST and the communication drivers provide this area.

Stream Word

The Stream word (word 1 of the message) has the following format:

```
+-------+-------+----+---+-------------+-----------+---------------+
| 3000  | REPLY | RB | FL| REMOTE BUSY | LINE ERR  | STREAM NUMBER |
|MASTER| FLAG   |    |   | RETRY COUNT | RETRY CNT |               |
+-------+-------+----+---+---+--+--+---+-----+-----+-+--+--+--+--+-+
   15      14     13   12  11 10  9  8    7      6   5  4  3  2  1 0
```

Where:

3000 Master : this bit is set if the request is from an HP 3000 or is a reply to an HP 3000 request.

Reply : this bit is set for a reply

RB : this bit is set if the request or reply has been rejected for any of the "remote busy" conditions: no SAM, no TCB, QUEUE busy, remote is quiescent.

FL : Format level:
- 0 Message format is Level 0
- 1 Message format is Level 1 or above (actual level number can be obtained from a field later in the header)

Remote busy count : each time a message is rejected because of any of the "remote-busy" conditions, this count is incremented. The value placed here initially is the complement of the value specified in reply to DSMOD question "REMOTE BUSY RETRIES[1 to 10]?", and incremented each time the request is rejected for the "remote-busy" conditions. When the count reaches 1111 (base 2), GRPM abandons re-try attempts.

Line error count : This field is used by DVA65 only. Initially zero, this count is incremented each time the driver reports an irrecoverable parity error on transmission. The driver makes seven retries, and this field can count up to three, so 21 re-tries are provided automatically. This field is used by DVA65 only.

Stream Field : indicates the type of request as shown below.

STREAM FIELD	TYPE OF REQUEST
0	MA or RR control message, handled by GRPM
1	DLIST (Directory List/Cartridge List)
2	$SDTLIST (CNSLM)
3	DEXEC (EXECW) (Sched with wait)
4	PTOP (Program-to-Program Communications)
5	DEXEC (EXECM) (Remote Exec Requests)
6	RFA (Remote File Access)
7	Operator Request (OPERM)
9 *	PROGL (System Cold Load requests)
10	RDBAM (Remote Database Access)
11 *	APLDX (APLDR for memory-only RTE-L)

* There is no request/reply information provided in this appendix for PROGL or APLDX. QUEUE sends download requests directly to PROGL. Therefore these requests are never logged by PLOG and cannot be examined.

Transaction Sequence Number

The "transaction sequence number" is the number assigned by #RSAX to identify the unique request. It provides the unique identifier which allows matching of the returning reply to the waiting master requestor.

Source and Destination Node Number

The "source" and "destination" node numbers identify the origination node and destination node. GRPM determines which node number to use when forwarding messages by the "reply" bit in the stream word; if zero, the message is outbound, so the destination node is used. If set, then this is a reply, so the "source" node number is used. The destination node number field will always be non-negative; if the original master call specifies a negative LU, #MAST converts it to a node number.

ECOD1, ECOD2 and Reporting Node Number

Error information is returned in the reply. Note that the error may have occurred "on the way over", hence, depending upon the error and the node reporting it, the request may never have arrived at the destination node. Even if the "reporting node number" is the same as the "destination node number", the request was delivered, but it may or may not have been executed, again, depending upon the error reported.

The storage for these three error codes is sometimes used for other purposes than to flag an error. The manner in which an error is indicated is determined by the conventions established between the master routine and its counterpart slave monitor. However, if an error is reported, it is always reported in ECOD1 & ECOD2, with the reporting node number placed in the third word. Most errors have qualifier codes defined, and the qualifier code is placed elsewhere in the message.

The first error code word contained in the header is ASCII, if an error occurred. The second codeword may be ASCII or numeric: the sign bit of the third word is set (A) if ASCII, clear if numeric. The first word indicates the major area: RF (RFA), FM (a File-Manager-returned error), RS (remote-session), etc. Thus, the user can distinguish an error code which was returned by a remote subsystem, such as FMP, from a DS-returned error code, without ambiguity.

It is up to the master routines to convert these errors to a form expected by the user program, e.g., to convert ASCII errors to numeric form if the only means to return an error indication to the caller is thru an integer parameter.

These error words are accessible through the subroutine DSERR. (NOTE: ECOD1 and ECOD2 are not used in HP 1000 to HP 3000 messages. DSERR calls issued under these circumstances may return erroneous results.)

Qualifier and Message Level

The "miscellaneous flags/message level" word has the following format:

```
 15  14  13  12  11  10   9  8   7  6  5  4   3  2  1  0
+----------------------------------------------------------+
|N | R | F | sub-level          |error qualifier|  level    |
+----------------------------------------------------------+
```

qual = error qualifier code (meaningful only if an error is reported)
level = message format level
reserved field = 1 bit
sub-level = the upgrade level within "level"
F (bit 13) Flag in in replies to indicate that the sub-level is valid.
R (bit 14) Reserved
N (bit 15) is set to indicate a no-reply request.

The level number of the local node is obtained by #MAST (requests) or #SLAV (replies) from the #LEVL entry point in 'RES'. The level number in the incoming message is compared with the level number of the local node to determine if message format conversion is necessary. The conversion is only necessary when a message is being sent to a node of a lower upgrade level or a message is being received from a lower upgrade level node. The sub-level number is used when minor changes in protocols (not requiring a message converter) are made. Slaves that have sub-level 0 echo the sub-level field of the master request. Slaves that have sub-level 1 and greater set F to distinguish themselves from slaves with sub-level 0 and insert their sub-level number. Thus, a slave reply with sub-level 0 may have 1 in its sub-level field, but this is invalid as indicated by the F-bit being cleared.

Message Accounting Variables

The "MA 'send'", "MA 'recv'" and "MA 'cancel'" sequence numbers are used by the Message Accounting software. See Chapter 5 for more details on message accounting.

Loop Counter

The "Loop counter" initially contains the two's complement of the "Loop counter" as set at node initialization time, and is incremented as it passes through each node.

Session ID Word

The "Session ID" word contains the Session identifier word for the local session and the remote session. See discussion of Remote session in Chapter 5 of this manual.

Request Specific Parameters

The formats of the individual requests and their replies are described later in this Appendix.

Local Appendage

The Local Appendage contains information needed to handle the requests in the local node but is meaningless to the remote node and is not transmitted.

The RTRY delay value contains a 16-bit delay value, calculated by GRPM and passed in this location to RTRY. Used only for this purpose.

S(bit 11) = MA assignment bit, indicates that MA at the local node has assigned valid sequence numbers. This bit is cleared by the driver on all incoming messages.

The MA retry portion of the last word is contained in bits 15 thru 12.

C (bit 8) = 0 if message is coming from communications link driver (i.e., the I/O status contained in the A-register upon return from the class-I/O "get" is a valid status. If C = 1, then the message has been re-queued, and the I/O status is to be ignored.

The LU portion (low 8 bits) of the same word is set by the driver (when C=0) to indicate the LU from which the message has been received.

The remainder of this document shows the specific formats for the headers of each request type, and all internal tables. There are two diagrams for each call and command. The side marked "REQUEST" is built by the master requestor subroutines and #MAST. The side marked "REPLY" is built by the slave-monitor and #SLAV. #MAST sends the request, #SLAV sends the reply.

Formats for specific types of message subject to change without notice.

Message Accounting Messages

Rerouting and Message Accouting share stream 0 when sending their messages through the network. The following message format is used:

```
      +--------------------------------+
 0    |                    stream # 0  |
 1    |                                |
 .    |  Fixed Header Area             |
 .    |                                |
12    |                                |
      +--------------------------------+
13    | Subfield Identifier  (0 or -1) |  Request specific area
14    | Type                           |
      +--------------------------------+
```

Subfield Identifier:

0 = re-routing message

-1 = Message Accounting message

The Subfield Identifier is used by GRPM to distinguish between Rerouting and Message Accounting messages which both come in on Stream 0.

For re-routing messages, the data buffer contains re-routing information such as node #, cost value, and hop count.

For MA messages, the word after the subfield identifier contains MA message type. Rerouting does not use this field. Fields already existing in the Fixed Header Area contain any data that MA may need to transfer to other nodes in the network, e.g. initial MA sequence numbers.

The MA message types (word 14) are:

```
RR   =  1  (Receiver Ready)
INIT =  2  (Initialize Request)
IR   =  4  (Initialize Response)
CAN  =  8  (Cancel Outstanding Message)
NR   = 16  (No Response)
```

Rerouting Update Message Format

The following messages are exchanged between neighbors to pass routing updates. They have the following format:

```
    +-------------------------------+
    |  NODE NUMBER                  |
    |  COST VALUE                   |        B
    |  HOP COUNT                    |     D  U
    -                               -     A  F
    -                               -     T  F
    |  (NODE NUMBER, COST, and      |     A  E
    |   HOP COUNT are repeated for  |        R
    |   as many values as need to   |
    |   be sent, maximum is 128     |
    |   entries per message)        |
    +-------------------------------+
   0|  STREAM #  =  0               |
   1|  Reserved                     |
   2|  SOURCE NODE NUMBER           |
   3|  DESTINATION NODE    = -LU    |
   4|  Not Used                     |     H  B
   5|  Not Used                     |     E  U
   6|  Not Used                     |     A  F
   7|  Not Used                     |     D  F
   8|  Not Used                     |     E  E
   9|  Not Used                     |     R  R
  10|  Not Used                     |
  11|  Not Used                     |
  12|  Not Used                     |
  13|  Zero (0)                     |
    +-------------------------------+
    |  Not Used                     |   APPENDAGE
    |  Last LU                      |    BUFFER
    +-------------------------------+
```

DS/1000-IV DLIST

```
           Request                                      Reply
     +-----------------------+               +-------------------------+
   0 | Stream # 1            |             0 | |1|                    1|
     |                       |               |                         |
     |  Fixed Header Area    |               |   Fixed Header Area     |
     |                       |               |                         |
     |                       |               |                         |
     |                       |               |                         |
     |                       |               |                         |
     |                       |            12 |                         |
  12 |                       |            13 | STATUS (#0:DONE)        |
  13 | MAX LINE LENGTH       |            14 | ACTUAL LINE LENGTH      |
  14 | NEW RQ FLAG=0         |            15 | BROUT=CO-ROUTINE ADR    |
  15 | \                     |            16 | \                       |
  16 |  > NAME FILTER        |            17 |  > NAME FILTER          |
  17 | /                     |            18 | /                       |
  18 |(MASTER SECURITY CODE) |            19 | (MASTER SECURITY CODE)  |
  19 | (ICR))                |            20 | (ICR)                   |
  20 | (TYPE)                |            21 | (TYPE)                  |
     +-----------------------+            22 | CURRENT DISC LU         |
     .                       .            23 | CURRENT DISC TRACK      |
     .                       .            24 | CURRENT SECTOR          |
     .                       .            25 | DISPLACEMENT IN BUFFER  |
     .                       .            26 | #SECTORS/TRACK          |
     .                       .            28 | DISP.IN DIRECT LU       |
     .                       .            29 | #DIRECTORY TRACKS       |
     .-----------------------.               +-------------------------+
```

CURRENT DISC LU thru #DIRECTORY TRACKS area of reply is also stored in request message, after first time. In this manner, DLIST is able to continue listings from multiple requesters.

DS/1000-IV Program-To-Program Calls

P-TO-P "FCODE" word:

```
+-----+---------------+-+---------------------+
|  FF | C |               | |       FCODE        |
+-----+-----------------+-+-------------------+
 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0
```

```
FF: Bits 15 and 14 = 0 in requests
                   = 1 in accepted replies
                   = 2 in rejected replies

    Bit  13        = 0 do not clone slave program
                     1 clone slave program (if in session monitor node)

    Bits 12-8        reserved for future use.

FCODE: Bits 6-0    = 1 for POPEN
                   = 2 for PREAD
                   = 3 for PWRIT
                   = 4 for PCONT
                   = 5 for PCLOS  (bit 7 is set to indicate FINIS)
                   = 6 for Slave off
                   = 7 for Slave list
```

PCB format:

```
    word 1: Slave program's ID segment address
         2: Slave program's class number
         3: Reserved for future use
         4: Slave node
```

POPEN

```
              Request
    +-----------------------+
  0 | Stream # 4            |
    |                       |
    |  Fixed Header Area    |
    |                       |
    |                       |
    |                       |
    |                       |
 12 |                       |
 13 |FF|C|    | | FCODE=1   |
 14 |  P        N \         |
 15 |  A        M  > NAME   |
 16 |  E        b /         |
 17 | ITAG (1)              |
 18 | ITAG (2)              |
 19 | ITAG (3)              |
    |    .                  |
    |    .                  |
    |    .                  |
    |    .                  |
    |                       |
    |                       |
 35 | ITAG (19)             |
 36 | ITAG (20)             |
    +-----------------------+
```

```
               Reply
    +-------------------------+
  0 | 40004 (8)               |
    |                         |
    |   Fixed Header Area     |
    |                         |
    |                         |
    |                         |
    |                         |
 12 |                         |
 13 |FF|        | | FCODE=1   |
 14 | SLAVE PROGRAM ID        |
 15 | SLAVE I/O CLASS         |
 16 |                         |
 17 | ITAG (1)                |
 18 | ITAG (2)                |
 19 | ITAG (3)                |
    |    .                    |
    |    .                    |
    |    .                    |
    |    .                    |
    |                         |
    |                         |
 35 | ITAG (19)               |
 36 | ITAG (20)               |
    +---  --------------------+
```

PREAD

```
              Request                                   Reply
     +-----------------------+             +-------------------------+
   0 | Stream # 4            |           0 | 40004 (8)               |
     |                       |             |                         |
     |  Fixed Header Area    |             |  Fixed Header Area      |
     |                       |             |                         |
     |                       |             |                         |
     |                       |             |                         |
  12 |                       |          12 |                         |
  13 |FF|      | | FCODE=2   |          13 |FF|     | | FCODE=2     |
  14 | PROGRAM ID            |          14 | SLAVE PROGRAM ID        |
  15 | I/O CLASS             |          15 | SLAVE I/O CLASS         |
  16 | IL (>0)               |          16 |                         |
  17 | ITAG (1)              |          17 | ITAG (1)                |
  18 | ITAG (2)              |          18 | ITAG (2)                |
  19 | ITAG (3)              |          19 | ITAG (3)                |
     |   .                   |             |    .                    |
     |   .                   |             |    .                    |
     |   .                   |             |    .                    |
     |   .                   |             |    .                    |
     |                       |             |                         |
  35 | ITAG (19)             |          35 | ITAG (19)               |
  36 | ITAG (20)             |          36 | ITAG (20)               |
     +-----------------------+             +-------------------------+
```

PWRIT

```
              Request
     +-----------------------+
   0 | Stream # 4            |
     |                       |
     |  Fixed Header Area    |
     |                       |
     |                       |
     |                       |
  12 |                       |
  13 |FF|         | | FCODE=3|
  14 | PROGRAM ID            |
  15 | I/O CLASS             |
  16 | IL (>0)               |
  17 | ITAG (1)              |
  18 | ITAG (2)              |
  19 | ITAG (3)              |
     |   .                   |
     |   .                   |
     |   .                   |
     |   .                   |
     |                       |
     |                       |
  35 | ITAG (19)             |
  36 | ITAG (20)             |
     +-----------------------+
```

```
              Reply
     +-------------------------+
   0 | 40004 (8)               |
     |                         |
     |  Fixed Header Area      |
     |                         |
     |                         |
     |                         |
  12 |                         |
  13 |FF|        | | FCODE = 3 |
  14 | SLAVE PROGRAM ID        |
  15 | SLAVE I/O CLASS         |
  16 |                         |
  17 | ITAG (1)                |
  18 | ITAG (2)                |
  19 | ITAG (3)                |
     |   .                     |
     |   .                     |
     |   .                     |
     |   .                     |
     |                         |
     |                         |
  35 | ITAG (19)               |
  36 | ITAG (20)               |
     +------------------------
```

PCONT

Request

```
    +------------------------+
 0  | Stream # 4             |
    |                        |
    |  Fixed Header Area     |
    |                        |
    |                        |
    |                        |
    |                        |
12  |                        |
13  |FF|       | | FCODE=4   |
14  | PROGRAM ID             |
15  | I/O CLASS              |
16  |                        |
16  | ITAG (1)               |
17  | ITAG (2)               |
18  | ITAG (3)               |
    |     .                  |
    |     .                  |
    |     .                  |
    |     .                  |
    |     .                  |
    |                        |
    |                        |
    |                        |
    |                        |
    |                        |
    |                        |
35  | ITAG (19)              |
36  | ITAG (20)              |
    +------------------------+
```

Reply

```
    +--------------------------+
 0  | 40004 (8)                |
    |                          |
    |  Fixed Header Area       |
    |                          |
    |                          |
    |                          |
    |                          |
12  |                          |
13  |FF|       | | FCODE=4     |
14  | SLAVE PROGRAM ID         |
15  | SLAVE PROGRAM I O CLASS  |
16  |                          |
17  | ITAG (1)                 |
18  | ITAG (2)                 |
19  | ITAG (3)                 |
    |     .                    |
    |     .                    |
    |     .                    |
    |     .                    |
    |     .                    |
    |                          |
    |                          |
    |                          |
    |                          |
    |                          |
    |                          |
35  | ITAG (19)                |
36  | ITAG (20)                |
    +--------------------------+
```

PCLOS

```
         Request                                   Reply
   +-----------------------+               +-------------------------+
 0 | Stream # 4            |             0 | 40004 (8)               |
   |                       |               |                         |
   |  Fixed Header Area    |               |  Fixed Header Area      |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
12 |                       |            12 |                         |
13 |FF|     | | FCODE=5    |            13 |FF|          | | FCODE=5 |
14 | PROGRAM ID            |            14 | SLAVE PROGRAM ID        |
15 | I/O CLASS             |            15 | SLAVE PROGRAM I/O CLASS|
   |                       |               |                         |
   +-----------------------+               +-------------------------+
```

FINIS

```
         Request                                   Reply
   +-----------------------+               +-------------------------+
 0 | Stream # 4            |             0 | 40004 (8)               |
   |                       |               |                         |
   |  Fixed Header Area    |               |  Fixed Header Area      |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
12 |                       |            12 |                         |
13 |FF|     |1| FCODE=5    |            13 |FF|          | | FCODE=5 |
14 | PROGRAM ID            |            14 | SLAVE PROGRAM ID        |
15 | I/O CLASS             |            15 | SLAVE PROGRAM I/O CLASS|
   |                       |               |                         |
   +-----------------------+               +-------------------------+
```

Bit 7 in the FCODE word is set to indicate FINIS.

SLAVE OFF (REMAT)

```
           Request                                        Reply
   +-----------------------+                 +-------------------------+
 0 | Stream # 4            |               0 | 40004 (8)               |
   |                       |                 |                         |
   |  Fixed Header Area    |                 |  Fixed Header Area      |
   |                       |                 |                         |
   |                       |                 |                         |
   |                       |                 |                         |
12 |                       |              12 |                         |
13 |FF|      | | FCODE=6  |              13 |FF|      | | FCODE=6    |
14 |   P        R\ PROGRAM|                 |                         |
15 |   G        N > NAME   |                 |                         |
16 |   M        b/         |                 |                         |
   +-----------------------+                 +-------------------------+
```

SLAVE LIST (REMAT)

```
           Request                                        Reply
   +-----------------------+                 +-------------------------+
 0 | Stream # 4            |               0 | 40004 (8)               |
   |                       |                 |                         |
   |  Fixed Header Area    |                 |  Fixed Header Area      |
   |                       |                 |                         |
   |                       |                 |                         |
   |                       |                 |                         |
12 |                       |              12 |                         |
13 |FF|      | | FCODE=7  |              13 |FF|      | | FCODE=7    |
   |                       |                 |                         |
   |                       |                 |                         |
   |                       |                 |                         |
   +-----------------------+                 +-------------------------+
```

DS/1000-IV DEXEC

Normal Completion

ECOD1 and ECOD2 contain status information from the A and B Registers. If the A bit (bit 15 in Reporting Node Number word) equals zero, the A and B registers contain the information returned in them, respectively, by the operating system upon completion of the system request. This information varies according to the specific request, and is documented in the approriate RTE programmer's manual.

Error Condition Completion

ECOD1 and ECOD2 contain 4-character ASCII message (as do A and B Registers) If the "A" bit=1 in Reporting Node Number word the rest of Reporting Node Number word contains the Error Node Number.

The following ICODE's are handled by EXECM:

1 READ
2 WRITE
3 I/O CONTROL
6 TERMINATE PROGRAM SCHEDULED BY EXECM
10 SCHEDULE PROGRAM WITHOUT WAIT
11 TIME Request
12 EXECUTION TIME (OFFSET/ABSOLUTE)
13 I/O STATUS
25 PARTITION STATUS
99 PROGRAM STATUS

The following ICODE's are handled by EXECW:

6 TERMINATE PROGRAM SCHEDULED BY EXECW
9 PROGRAM SCHEDULE WITH WAIT
23 PROGRAM QUEUE WITH WAIT
24 PROGRAM QUEUE WITHOUT WAIT

READ

```
              Request                                    Reply
     +----------------------+            +-------------------------+
   0 | Stream # 5           |          0 | 40005 (8)               |
     |                      |            |                         |
     |  Fixed Header Area   |            |  Fixed Header Area      |
     |                      |            |                         |
  12 |                      |            |                         |
  13 |  ICODE = 1           |            |                         |
  14 |  ICNWD(1)            |            |                         |
  15 |  ICNWD(2)            |            |                         |
  16 |  IBUFL               |            |                         |
  17 | (IPRM1/Z-BUF OFFSET) |            +-------------------------+
  18 | (IPRM2/Z-BUF LENGTH) |
  19 | 0 (Place Holder)     |
  20 | (KEYWORD (RN))       |
     +----------------------+
```

WRITE

```
              Request                                    Reply
     +----------------------+            +-------------------------+
   0 | Stream # 5           |          0 | 40005 (8)               |
     |                      |            |                         |
     |  Fixed Header Area   |            |  Fixed Header Area      |
     |                      |            |                         |
  12 |                      |         12 |                         |
  13 | ICODE = 2            |            +-------------------------+
  14 | ICNWD (1)            |
  15 | ICNWD (2)            |           Note:  EQT5 (I/O Status is
  16 | IBUFL                |           returned in ECOD1 and to the
  17 | (IPRM1/Z-BUF OFFSET) |           caller in the A-Register.
  18 | (IPRM2/Z-BUF LENGTH) |           Transmission log is returned
  19 | 0 (Place Holder)     |           in ECOD2 and to the caller
  20 | (KEYWORD (RN))       |           in the B-Register.
     +----------------------+
```

Note: Parameters shown in parenthesis are optionally specified by the DEXEC user, but the parameters or zero place holders are always transmitted to the Slave Monitor (EXECW/EXECM).

I/O Control

```
              Request                                        Reply
     +-----------------------+                    +-------------------------+
   0 | Stream # 5            |                  0 | 40005 (8)               |
     |                       |                    |                         |
     |  Fixed Header Area    |                    |  Fixed Header Area      |
     |                       |                    |                         |
     |                       |                 12 |                         |
  12 |                       |                    +-------------------------+
  13 | ICODE = 3             |                    Note: EQT 5 is returned in
  14 | ICNWD(1)              |                    ECOD1 and to the caller in
  15 | ICNWD(2)              |                    the A-Register.  B-register
  16 | (IPRAM1)              |                    and ECOD2 are meaningless.
  17 | (IPRAM3/Z-BUF OFFSET  |
  18 | (IPRAM4/Z-BUF LENGTH  |
  19 | (IPRAM2)              |
  20 | (KEYWORD(RN))         |
     +-----------------------+
```

I/O Status

```
              Request                                        Reply
     +-----------------------+                    +-------------------------+
   0 | Stream # 5            |                  0 | 40005 (8)               |
     |                       |                    |                         |
     |  Fixed Header Area    |                    |  Fixed Header Area      |
     |                       |                    |                         |
  12 |                       |                    |                         |
  13 | ICODE = 13 (Decimal)  |                    |                         |
  14 | ICNWD(1) =LU#         |                    |                         |
  15 | ICNWD(2) =FCN, Etc.   |                    |                         |
  16 | 0                     |                 12 |                         |
  17 | Place Holder          |                 13 | ISTA1 (EQT5)            |
  18 | 0/(Z-BUF LENGTH)      |                 14 | ISTA2 (EQT4)            |
  19 | Place Holder          |                 15 | ISTA3 (UP/DOWN SUBCHAN) |
  20 | 0                     |                 16 | ISTA4 (DVTP+1: RTE-L)   |
     +-----------------------+                    +-------------------------+
```

Note: A- and B-Registers (ECOD1/ECOD2) contain negative word count of Z-buffer status information, when the status request is executed in an RTE-L node.

Schedule Program Without Wait

```
              Request
    +-----------------------+
  0 | Stream # 5            |
    |                       |
    |  Fixed Header Area    |
    |                       |
    |                       |
 12 |                       |
 13 |  ICODE = 10 (Decimal)|
 14 |  P          R \       |
 15 |  G          N  >INAME |
 16 |  M          b /       |
 17 | (IPRM1)               |
 18 | (IPRM2)               |
 19 | (IPRM3)               |
 20 | (IPRM4)               |
 21 | (IPRM5)               |
 22 | (IBUFR)-PLACE HOLDER  |
 23 | (IBUFL)   ONLY        |
    +-----------------------+
```

```
                Reply
    +-------------------------+
  0 | 40005 (8)               |
    |                         |
    |  Fixed Header Area      |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
 12 |                         |
    +-------------------------+
```

Note: ECOD1 contains program status, which is returned to the caller in the A-Register.

Time Request

```
            Request                                     Reply
    +-----------------------+                +-------------------------+
 0  | Stream # 5            |            0   | 40005 (8)               |
    |                       |                |                         |
    |  Fixed Header Area    |                |  Fixed Header Area      |
    |                       |                |                         |
    |                       |                |                         |
    |                       |                |                         |
    |                       |                |                         |
12  |                       |           12   |                         |
13  | ICODE = 11 (Decimal)  |           13   | MSEC                    |
14  | \                     |           14   | SEC                     |
15  | |                     |           15   | MIN                     |
16  | | PLACE HOLDERS       |           16   | HRS                     |
17  | | FOR REPLY           |           17   | DAY OF YEAR             |
18  | /                     |           18   | YEAR                    |
    +-----------------------+                +-------------------------+
```

Timed Execution (Offset/Absolute)

```
            Request                                     Reply
    +-----------------------+                +-------------------------+
 0  | Stream # 5            |            0   | 40005 (8)               |
    |                       |                |                         |
    |  Fixed Header Area    |                |  Fixed Header Area      |
    |                       |                |                         |
    |                       |                |                         |
    |                       |                |                         |
    |                       |                |                         |
12  |                       |                |                         |
13  | ICODE = 12 (Decimal)  |                |                         |
14  | P          R\         |                |                         |
15  | G          N > IPROG  |                |                         |
16  | M          b/         |                +-------------------------+
17  | IRESL                 |
18  | MTPLE                 |
19  | IOFST/IHRS            |
20  | (NONE)/MINS           |
21  | (NONE)/ISECS          |
22  | (NONE)/MSECS          |
    +-----------------------+
```

Partition Status

```
             Request                                        Reply
    +-----------------------+                  +-------------------------+
 0  | Stream # 5            |               0  | 40005 (8)               |
    |                       |                  |                         |
    |  Fixed Header Area    |                  |  Fixed Header Area      |
    |                       |                  |                         |
    |                       |                  |                         |
    |                       |                  |                         |
12  |                       |                  |                         |
13  | ICODE = 25  (Decimal) |                  |                         |
14  | IPART (#)             |                  |                         |
15  | \                     |              12  |                         |
16  |  | PLACE HOLDERS      |              13  | IPAGE                   |
17  |  | FOR Reply          |              14  | IPNUM                   |
18  | /                     |              15  | ISTAT                   |
    +-----------------------+                  +-------------------------+
```

Program Status

```
             Request                                        Reply
    +-----------------------+                  +-------------------------+
 0  | Stream # 5            |               0  | 40005 (8)               |
    |                       |                  |                         |
    |  Fixed Header Area    |                  |  Fixed Header Area      |
    |                       |                  |                         |
    |                       |                  |                         |
    |                       |                  |                         |
12  |                       |                  |                         |
13  | ICODE = 99            |                  |                         |
14  | P         R\          |                  |                         |
15  | G         N > IPROG   |              12  |                         |
16  | M         b/          |              13  | (ISTAT)                 |
17  | (ISTAT)               |                  +-------------------------+
    +-----------------------+
```

Note: ECOD1 contains program status which is returned to the caller in the A-Register. ECOD2 = 0, which is returned to the caller in the B-Register. If an error occured: ECOD1/A = ASCII 'DS', and ECOD2/B = numeric -1; the sign of ECOD3 = 0.

Program Termination

```
               Request                                      Reply
     +----------------------+               +------------------------+
   0 | Stream #3 or 5       |             0 | STREAM + 40000 (8)     |
     |                      |               |                        |
     |  Fixed Header Area   |               |  Fixed Header Area     |
     |                      |               |                        |
     |                      |               |                        |
     |                      |               |                        |
     |                      |               |                        |
  12 |                      |               |                        |
  13 | ICODE = 6            |               |                        |
  14 | P          R\        |               |                        |
  15 | G          N > INAME |               |                        |
  16 | M          b/        |               +------------------------+
  17 | INUMB                |
  18 | (IPRM1) \  THESE ARE |
  19 | (IPRM2)  | MEANING-  |
  20 | (IPRM3)  | LESS      |
  21 | (IPRM4)  | WHEN      |
  22 | (IPRM5) /  INUMB > 0 |
     +----------------------+
```

Note: This request is originated on stream 5, but may be requeued onto stream 3 at the destination node, if EXECM determines that the specified program is not EXECM's son.

Program Schedule With Wait

```
                 Request
     +-----------------------+
   0 | Stream # 3            |
     |                       |
     |  Fixed Header Area    |
     |                       |
     |                       |
     |                       |
  12 |                       |
  13 | ICODE =9              |
  14 | P          R\         |
  15 | G          N > INAME  |
  16 | M          b/         |
  17 | (IPRM1)               |
  18 | (IPRM2)               |
  19 | (IPRM3)               |
  20 | (IPRM4)               |
  21 | (IPRM5)               |
  22 | (IBUFR)-PLACE HOLDER  |
  23 | (IBUFL)   ONLY        |
     +-----------------------+
```

```
                 Reply
     +-------------------------+
   0 | 40003 (8)               |
     |                         |
     |  Fixed Header Area      |
     |                         |
     |                         |
     |                         |
     |                         |
     |                         |
     |                         |
  12 |                         |
  13 | (RETURNED IPRM1)        |
  14 | (RETURNED IPRM2)        |
  15 | (RETURNED IPRM3)        |
  17 | (RETURNED IPRM4)        |
  18 | (RETURNED IPRM5)        |
     +-------------------------+
```

Program Queue With Wait

```
             Request                                    Reply
      +-----------------------+                 +-----------------------+
   0  | Stream # 3            |              0  | 40003 (8)             |
      |                       |                 |                       |
      |  Fixed Header Area    |                 |  Fixed Header Area    |
      |                       |                 |                       |
      |                       |                 |                       |
      |                       |                 |                       |
      |                       |                 |                       |
  12  |                       |             12  |                       |
  13  | ICODE = 23 (Decimal)  |             13  | (RETURNED IPRM1)      |
  14  |  P         R\         |             14  | (RETURNED IPRM2)      |
  15  |  G         N > INAME  |             15  | (RETURNED IPRM3)      |
  16  |  M         b/         |             16  | (RETURNED IPRM4)      |
  17  | (IPRM1)               |             17  | (RETURNED IPRM5)      |
  18  | (IPRM2)               |                 +-----------------------+
  19  | (IPRM3)               |
  20  | (IPRM4)               |
  21  | (IPRM5)               |
  22  | (IBUFR)-PLACE HOLDER  |
  23  | (IBUFL)   ONLY        |
      +-----------------------+
```

Program Queue Without Wait

```
             Request                                 Reply
     +-----------------------+          +-------------------------+
   0 | Stream # 3            |        0 | 40003 (8)               |
     |                       |          |                         |
     |  Fixed Header Area    |          |  Fixed Header Area      |
     |                       |          |                         |
     |                       |          |                         |
     |                       |          |                         |
     |                       |          |                         |
  12 |                       |          |                         |
  13 | ICODE = 24 (Decimal)  |          |                         |
  14 |   P        R\         |       12 |                         |
  15 |   G        N > INAME  |          +-------------------------+
  16 |   M        b/         |
  17 | (IPRM1)               |
  18 | (IPRM2)               |
  19 | (IPRM3)               |
  20 | (IPRM4)               |
  21 | (IPRM5)               |
  22 | (IBUFR)-PLACE HOLDER  |
  23 | (IBUFL)   ONLY        |
     +-----------------------+
```

Remote File Access

Error information:

```
              NUMERIC                OR  ASCII
          +---------------------+    +------------+
ECOD1     |                     |    | "a1"  "a2" |
ECOD2     | IERR (FMP ERROR #)  |    | "a3"  "a4" |
RNN       |   ERR. NODE         |    |   ERR.NODE |
          +---------------------+    +------------+
```

If error code >= 0, no error

RNN (Reporting Node Number) is returned as the IERLC parameter when it is specified by the user in the call.

The DCB array contains the following values:

```
DCB(1)  Master ID Segment Address
DCB(2)  Entry Number (for RFAM)
DCB(3)  Not used
DCB(4)  File's node
```

DPURG

```
            Request                                   Reply
    +-----------------------+            +-------------------------+
 0  | Stream # 6            |         0  | 40006 (8)               |
    |                       |            |                         |
    |  Fixed Header Area    |            |  Fixed Header Area      |
    |                       |            |                         |
    |                       |            |                         |
    |                       |            |                         |
    |                       |            |                         |
12  |                       |            |                         |
13  | FCN = 8               |            |                         |
14  | F            L\       |            |                         |
15  | N            A > NAME |            |                         |
16  | M            E/       |            +-------------------------+
17  | (ICR(1))              |
18  | (ID SEGMENT ADDRESS)  |
19  | (ISECU)               |
    +-----------------------+
```

DOPEN

Request

```
   +-----------------------+
 0 | Stream # 6            |
   |                       |
   |  Fixed Header Area    |
   |                       |
   |                       |
   |                       |
12 |                       |
13 | FCN = 6               |
14 | F            L\       |
15 | N            A > NAME |
16 | M            E/       |
17 | (ICR (1))             |
18 | (ID SEGMENT ADDRESS)  |
19 | (ISECU)               |
20 | (IOPTN)               |
   +-----------------------+
```

Reply

```
   +-------------------------+
 0 | 40006 (8)               |
   |                         |
   |  Fixed Header Area      |
   |                         |
   |                         |
   |                         |
12 |                         |
13 | ENTRY #                 |
   +-------------------------+
```

DWRIT

Request

```
   +-----------------------+
 0 | Stream # 6            |
   |                       |
   |  Fixed Header Area    |
   |                       |
   |                       |
   |                       |
12 |                       |
13 | FCN = 12              |
14 | DCB (1)               |
15 | DCB (2)               |
16 | DCB (3)               |
17 | IL                    |
18 | (NUM)                 |
   +-----------------------+
```

Reply

```
   +-------------------------+
 0 | 40006 (8)               |
   |                         |
   |  Fixed Header Area      |
   |                         |
   |                         |
12 |                         |
   +-------------------------+
```

DREAD

```
          Request                              Reply
   +----------------------+          +-------------------------+
 0 | Stream # 6           |        0 | 40006 (8)               |
   |                      |          |                         |
   |  Fixed Header Area   |          |  Fixed Header Area      |
   |                      |          |                         |
   |                      |          |                         |
   |                      |          |                         |
   |                      |          |                         |
12 |                      |       12 |                         |
13 | FCN = 9              |       13 | LEN (Transmission Log)  |
14 | DCB (1)              |          +-------------------------+
15 | DCB (2)              |
16 | DCB (3)              |
17 | IL                   |
18 | (NUM)                |
   +----------------------+
```

DPOSN

```
          Request                              Reply
   +----------------------+          +-------------------------+
 0 | Stream # 6           |        0 | 40006 (8)               |
   |                      |          |                         |
   |  Fixed Header Area   |          |  Fixed Header Area      |
   |                      |          |                         |
   |                      |          |                         |
   |                      |          |                         |
   |                      |       12 |                         |
12 |                      |          +-------------------------+
13 | FCN = 7              |
14 | DCB (1)              |
15 | DCB (2)              |
16 | DCB (3)              |
17 | NUR                  |
   | (IR)                 |
   +----------------------+
```

DWIND

```
           Request                                   Reply
   +-----------------------+               +-------------------------+
 0 | Stream # 6            |             0 | 40006 (8)               |
   |                       |               |                         |
   |  Fixed Header Area    |               |  Fixed Header Area      |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
   |                       |            12 |                         |
12 |                       |               +-------------------------+
13 | FCN = 11              |
14 | DCB (1)               |
15 | DCB (2)               |
16 | DCB (3)               |
   +-----------------------+
```

DNAME

```
           Request                                   Reply
   +-----------------------+               +-------------------------+
 0 | Stream # 6            |             0 | 40006 (8)               |
   |                       |               |                         |
   |  Fixed Header Area    |               |  Fixed Header Area      |
   |                       |               |                         |
   |                       |               |                         |
   |                       |               |                         |
   |                       |            12 |                         |
12 |                       |               +-------------------------+
13 | FCN = 5               |
14 | F           L\        |
15 | N           A > NAME  |
16 | M           E/        |
17 | (ICR (1))             |
18 | (ID SEGMENT ADDRESS)  |
19 | (ISECU)               |
20 |  N         E\         |
21 |  W         N > NNMAE  |
22 |  A         M/         |
   +-----------------------+
```

DCONT

```
            Request                                    Reply
    +-----------------------+               +-------------------------+
  0 | Stream # 6            |             0 | 40006 (8)               |
    |                       |               |                         |
    |  Fixed Header Area    |               |  Fixed Header Area      |
    |                       |               |                         |
    |                       |               |                         |
    |                       |               |                         |
    |                       |               |                         |
 12 |                       |               +-------------------------+
 13 | FCN = 2               |
 14 | DCB (1)               |
 15 | DCB (2)               |
 16 | DCB (3)               |
 17 | ICON1                 |
 18 | (ICON2)               |
    +-----------------------+
```

DLOCF

```
            Request                                    Reply
    +-----------------------+               +-------------------------+
  0 | Stream # 6            |             0 | 40006 (8)               |
    |                       |               |                         |
    |  Fixed Header Area    |               |  Fixed Header Area      |
    |                       |               |                         |
    |                       |               |                         |
    |                       |               |                         |
    |                       |               |                         |
 12 |                       |            12 |                         |
 13 | FCN = 4               |            13 | IREC                    |
 14 | DCB (1)               |            14 | (IRB)                   |
 15 | DCB (2)               |            15 | (IOFF)                  |
 16 | DCB (3)               |            16 | (JSEC)                  |
    +-----------------------+            17 | (JLU)                   |
                                         18 | (JTY)                   |
                                         19 | (JREC)                  |
                                            +-------------------------+
```

DAPOS

```
             Request                                        Reply
     +-----------------------+                  +-------------------------+
   0 | Stream # 6            |                0 | 40006 (8)               |
     |                       |                  |                         |
     |  Fixed Header Area    |                  |  Fixed Header Area      |
     |                       |                  |                         |
     |                       |                  |                         |
     |                       |                  |                         |
     |                       |               12 |                         |
  12 |                       |                  +-------------------------+
  13 | FCB = 0               |
  14 | DCB (1)               |
  15 | DCB (2)               |
  16 | DCB (3)               |
  17 | IREC                  |
  18 | (IRB)                 |
  19 | (IOFF)                |
     +-----------------------+
```

DSTAT

```
             Request                                        Reply
     +-----------------------+                  +-------------------------+
   0 | Stream # 6            |                0 | 40006 (8)               |
     |                       |                  |                         |
     |  Fixed Header Area    |                  |  Fixed Header Area      |
     |                       |                  |                         |
     |                       |                  |                         |
     |                       |                  |                         |
     |                       |               12 |                         |
  12 |                       |                  +-------------------------+
  13 | FCN = 10              |
  14 | \ PLACE HOLDERS       |
  15 | / FOR Reply           |
     +-----------------------+
```

DCLOS

```
              Request                                   Reply
    +-----------------------+              +-------------------------+
  0 | Stream # 6            |            0 | 40006 (8)               |
    |                       |              |                         |
    |  Fixed Header Area    |              |  Fixed Header Area      |
    |                       |              |                         |
    |                       |              |                         |
    |                       |           12 |                         |
    |                       |              +-------------------------+
 12 |                       |
 13 | FCN = 1               |
 14 | DCB (1)               |
 15 | DCB (2)               |
 16 | DCB (3)               |
 17 | (ITRUN)               |
    +-----------------------+
```

DCRET

```
              Request                                   Reply
    +-----------------------+              +-------------------------+
  0 | Stream # 6            |            0 | 40006 (8)               |
    |                       |              |                         |
    |  Fixed Header Area    |              |  Fixed Header Area      |
    |                       |              |                         |
    |                       |              |                         |
    |                       |              |                         |
    |                       |              |                         |
 12 |                       |              |                         |
 13 | FCN = 3               |              |                         |
 14 |    F         L\       |           12 |                         |
 15 |    N         A > NAME |           13 | ENTRY #                 |
 16 |    M         E/       |              +-------------------------+
 17 | [ICR (1)]             |
 18 | @ (ID SEG)            |
 19 | [ISECU]               |
 20 | [ISIZE(1)]            |
 21 | [ISIZE(2)]            |
 22 | ITYPE                 |
    +-----------------------+
```

DS/1000-IV Remote Operator Requests

```
              Request                                    Reply
      +-----------------------+                  +-------------------------+
    0 | Stream # 7            |                0 | 40007 (8)               |
      |                       |                  |                         |
      |  Fixed Header Area    |                  |  Fixed Header Area      |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
   12 |                       |                  |                         |
   13 | Command length (in    |                  |                         |
      |      positive  bytes) |               12 |                         |
      |                       |               13 | Reply LENGTH (+ WORDS)  |
      |                       |                  |                         |
      |       COMMAND         |                  |          Reply          |
      |       (ASCII)         |                  |         (ASCII)         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      |                       |                  |                         |
      +-----------------------+                  +-------------------------+
```

DS/1000-IV LOGON Request

```
              REQUEST
    +-------------------------+
 0  | STREAM = 7              |
    |                         |
    |  Fixed Header Area      |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
12  |                         |
13  | SRC SESSN |      0      |
14  | COMMAND LENGTH = 2      |
15  |    "X"    |     "X"     |
16  | ICODE = 1               |
17  | NAME LEN <= 32          |
18  |                         |
19  |        USER             |
20  |                         |
21  |        NAME.            |
22  |                         |
23  |        GROUP            |
24  |                         |
25  |        NAME/            |
26  |                         |
27  |        PASS             |
28  |                         |
29  |        WORD             |
30  |                         |
    +-------------------------+
```

```
               REPLY
    +-------------------------+
 0  | 40007 (8)               |
    |                         |
    |  Fixed Header Area      |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
    |                         |
12  |                         |
13  | SRC SESSN | DST SESSN   |
    +-------------------------+
```

LEN is specified in positive bytes. The field for the character strings is variable length. Note that source and destination session ID are NOT reversed for the reply as for DS/3000 "from process number" and "to process number".

DS/1000-IV LOGOF Request

```
              REQUEST                                     REPLY
   +-------------------------+                 +-------------------------+
 0 | STREAM = 7              |           0     | 40007 (8)               |
   |                         |                 |                         |
   |  Fixed Header Area      |                 |  Fixed Header Area      |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
   |                         |                 |                         |
12 |                         |          12     |                         |
13 | SRC SESSN |      0      |          13     | SRC SESSN |      0      |
14 | COMMAND LENGTH = 2      |                 +-------------------------+
15 |    "X"    |     "X"     |
16 | ICODE = 0 OR -1         |
17 | SESSION ID              |
   +-------------------------+
```

ICODE = -1 is used by UPLIN to indicate "no reply" to RSM.

DS/1000-IV Non-Session Access Request

```
            REQUEST                                       REPLY
    +-------------------------+               +-------------------------+
 0  | STREAM = 7              |               | 40007 (8)               |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
    |                         |               |                         |
12  |                         |           12  |                         |
13  | SRC SESSN |      0      |           13  | SRC SESSN |     254     |
14  | COMMAND LENGTH = 2      |               +-------------------------+
15  |    "X"    |     "X"     |
16  | ICODE = 2               |
17  | PASSWORD LEN <= 10      |
18  |                         |
19  |        PASS             |
20  |                         |
21  |        WORD             |
22  |                         |
    +-------------------------+
```

DS/1000-IV RDBA Request

```
                REQUEST
      +------------------------+
   0  | STREAM = 10            |
      |                        |
      |  Fixed Header Area     |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
  15  | call index             |
  16  | call mode              |
  17  | call item or set #     |    (for DBOPN, the level code word)
  20  | search item # (DBFND)  |
  21  |     (2 words)          |
  22  | max. return RT         |    (Run Table size DBOPN)
  23  | word size-IBASE parame |
  24  | IBASE parameter        |
      |                        |
      +------------------------+

                 REPLY
      +------------------------+
   0  | 40012 (8)              |
      |                        |
      |  Fixed Header Area     |
      |                        |
      |                        |
      |                        |
      |                        |
      |                        |
  15  |  RDBA call status array|
   |  |                        |
   |  |                        |
   v  |                        |
  26  |                        |
  27  | RDBA database #(DBOPN) |
      +------------------------+
```

Global Block (Common To All Modules)

```
*
*****************************************************************
*                                                               *
*     G L O B A L   B L O C K                                   *
*                                                               *
*     GLOBAL OFFSETS INTO DS/1000-IV MESSAGE BUFFERS, USED BY:  *
*                                                               *
*         REMAT, RFMST, DEXEC, DMESS, FLOAD, POPEN, #MAST       *
*         GET,   #SLAV, RQCNV, RPCNV, GRPM,  DINIT, PTOPM       *
*         EXECM, EXECW, OPERM, RFAM1, RFAM2, DLIST,             *
*         DSTIO, LUMAP, #CMGT, INCNV, OTCNV, RMTIO              *
*         RSM,   DLGON, #DISM, #DSSM, #MSSM, #SCSM, #UPSM       *
*****************************************************************
*
***!!!!! THE ORDER OF THE FIRST 8 WORDS (#STR THRU #LVL) IS     *
***!!!!!     FIXED BY THE REQUIREMENT THAT THE STREAM, ADDRESSES *
***!!!!!     ERROR CODES & LEVEL # ALWAYS BE IN THE SAME PLACE,  *
***!!!!!     REGARDLESS OF MESSAGE FORMAT.  THIS ALSO MAKES      *
***!!!!!     STORE-AND-FORWARD CODE MUCH SIMPLER.                *
#STR  EQU 0        STREAM WORD.
#SEQ  EQU #STR+1   SEQUENCE NUMBER.
#SRC  EQU #SEQ+1   SOURCE NODE #.
#DST  EQU #SRC+1   DEST. NODE #.
#EC1  EQU #DST+1   REPLY ECOD1.
#EC2  EQU #EC1+1   REPLY ECOD2.
#ENO  EQU #EC2+1   NUMBER OF NODE REPORTING ERROR.
#ECQ  EQU #ENO+1   ERROR CODE QUALIFIER (BITS 4 TO 7)
#LVL  EQU #ECQ     MESSAGE FORMAT LEVEL (BITS 0 TO 3)
*
#MAS  EQU #LVL+1   MA "SEND" SEQ. #
#MAR  EQU #MAS+1   MA "RECV" SEQ. #
#MAC  EQU #MAR+1   MA "CANCEL" FLAGS
#HCT  EQU #MAC+1   HOP COUNT
#SID  EQU #HCT+1   SESSION ID WORD
*
#EHD  EQU #SID     LAST ITEM OF HEADER
#MHD  EQU #EHD+1   MINIMUM HEADER SIZE
#REQ  EQU #MHD     START OF REQUEST SPECIFIC AREA
#REP  EQU #MHD     START OF REPLY SPECIFIC AREA
*
#MXR  EQU #MHD+24  <<< MAXIMUM DS REQ/REPLY BUFFER SIZE >>>
#LSZ  EQU 2        <<< SIZE OF LOCAL APPENDAGE AREA >>>
*
*****************************************************************
*
```

Remote 'EXEC' Block

```
* DXBLK-START
*
*****************************************************************
*                                                               *
*      D E X E C   B L O C K                                    *
*                                                               *
*      OFFSETS INTO DS/1000-IV DEXEC MESSAGE BUFFERS, USED BY:  *
*                                                               *
*       DEXEC, EXECM, EXECW, RQCNV, RPCNV, FLOAD, REMAT         *
*                                                               *
*****************************************************************
*
* OFFSETS INTO DEXEC REQUEST BUFFERS.
*
#ICD  EQU #REQ      ICODE FOR DEXEC(ALL)
#CNW  EQU #ICD+1    CONWD FOR DEXEC(1,2,3,13)
#CWX  EQU #CNW+1    DLUEX EXTENSION FOR DEXEC(1,2,3,13)
#BFL  EQU #CWX+1    IBUFL FOR DEXEC(1,2)
#PM1  EQU #BFL+1    IPRM1 FOR DEXEC(1,2)
#PM2  EQU #PM1+1    IPRM2 FOR DEXEC(1,2)
#ZOF  EQU #PM1      Z-BUFFER OFFSET FOR DEXEC(1,2,3,13)
#ZLN  EQU #PM2      Z-BUFFER LENGTH FOR DEXEC(1,2,3,13)
#PR2  EQU #PM2+1    2ND OPT. PARAMETER FOR DEXEC(3) [A/L-Series].
#KEY  EQU #PR2+1    KEYWORD(RN) FOR DEXEC(1,2,3) [A/L-Series].
#PRM  EQU #CWX+1    IPRAM FOR DEXEC(3)
#PGN  EQU #ICD+1    PRGNM FOR DEXEC(6,9,10,12,23,24,99)
#INU  EQU #PGN+3    INUMB FOR DEXEC(6)
#DPM  EQU #INU+1    PARMS FOR DEXEC(6)           (5-WORD AREA)
#PMS  EQU #PGN+3    PARMS FOR DEXEC(9,10,23,24) (5-WORD AREA)
#IBF  EQU #PMS+5    IBUFR FOR DEXEC(9,10,23,24)
#IBL  EQU #IBF+1    IBUFL FOR DEXEC(9,10,23,24)
#FNO  EQU #IBL+1    FNOD  FOR DEXEC(9)            (APLDR)
#RSL  EQU #PGN+3    IRESL FOR DEXEC(12)
#MPL  EQU #RSL+1    MTPLE FOR DEXEC(12)
#HRS  EQU #MPL+1    IHRS  FOR DEXEC(12)
#MIN  EQU #HRS+1    IMIN  FOR DEXEC(12)
#SEC  EQU #MIN+1    ISECS FOR DEXEC(12)
#MSC  EQU #SEC+1    MSECS FOR DEXEC(12)
#PAR  EQU #ICD+1    PARTI FOR DEXEC(25)          (PARTITION #)
#IST  EQU #PGN+3    ISTAT FOR DEXEC(99)
*
* OFFSETS INTO DEXEC REPLY BUFFERS.
*
#EQ5  EQU #EC1      EQT 5 FOR DEXEC(1,2,3)
#XML  EQU #EC2      TRANSMISSION LOG (DEXEC 1,2)
```

```
#RPM  EQU #REP       PRAMS FOR DEXEC(9,23)      (5-WORD AREA)
#TMS  EQU #REP       MSEC  FOR DEXEC(11)
#TSC  EQU #TMS+1     SEC   FOR DEXEC(11)
#TMN  EQU #TSC+1     MIN   FOR DEXEC(11)
#THR  EQU #TMN+1     HRS   FOR DEXEC(11)
#TDA  EQU #THR+1     DAY   FOR DEXEC(11)
#TYR  EQU #TDA+1     YEAR  FOR DEXEC(11)
#ST1  EQU #REP       ISTA1 FOR DEXEC(13)
#ST2  EQU #ST1+1     ISTA2 FOR DEXEC(13)
#ST3  EQU #ST2+1     ISTA3 FOR DEXEC(13)
#ST4  EQU #ST3+1     ISTA4 FOR DEXEC(13) [RTE-L].
#PAG  EQU #REP       IPAGE FOR DEXEC(25)
#IPN  EQU #PAG+1     IPNUM FOR DEXEC(25)
#PST  EQU #IPN+1     ISTAT FOR DEXEC(25)
#KST  EQU #REP       ISTAT FOR DEXEC(99)
*
* MAXIMUM SIZE OF DEXEC REQUEST/REPLY BUFFER.
*
#DLW  EQU #MHD+11+#LSZ   M A X I M U M   S I Z E   ! ! !
*
* MAXIMUM SIZE OF DEXEC/EXECM DATA BUFFER.
*
#DBS  EQU 512            M A X I M U M   S I Z E   ! ! !
*
```

RFA Block

```
*****************************************************************
*       R F A   B L O C K               REV 2013  791119        *
*                                                               *
*       OFFSETS INTO DS/1000-IV RFA MESSAGE BUFFERS, USED BY:   *
*                                                               *
*           RFMST, RFAM1, RFAM2, REMAT, RQCNV, RPCNV            *
*                                                               *
*****************************************************************
*
* OFFSETS INTO RFA REQUEST BUFFERS.
*
#FCN  EQU #REQ      RFA FUNCTION CODE.
#DCB  EQU #FCN+1    DCB/FILENAME AREA.
#IRC  EQU #DCB+3    DAPOS: IREC
#IRB  EQU #IRC+1           IRB
#XIB  EQU #IRC+2           IRB   (DXAPO)
#IOF  EQU #IRB+1           IOFF
#XIO  EQU #XIB+2           IOFF  (DXAPO)
#ITR  EQU #DCB+3    DCLOS: ITRUN
#IC1  EQU #DCB+3    DCONT: ICON1
#IC2  EQU #IC1+1           ICON2
#ICR  EQU #DCB+3    DCRET,DNAME,DOPEN,DPURG: ICR(1)
#ID   EQU #ICR+1                             IDSEG
#ISC  EQU #ID+1                              ISECU
#SIZ  EQU #ISC+1    DCRET: ISIZE(1)
#SZ2  EQU #SIZ+1           ISIZE(2)
#XRS  EQU #SIZ+2           RECSZ (DXCRE)
#TYP  EQU #SZ2+1           ITYPE
#XTY  EQU #XRS+2           ITYPE (DXCRE)
#NNM  EQU #ISC+1    DNAME: NNAME
#IOP  EQU #ISC+1    DOPEN: IOPTN
#NUR  EQU #DCB+3    DPOSN: NUR
#IR   EQU #NUR+1           IR
#XIR  EQU #NUR+2           IR    (DXPOS)
#IL   EQU #DCB+3    DREAD,DWRIT: IL
#NUM  EQU #IL+1                  NUM
#LEN  EQU #FCN+1    DSTAT: ILEN
#FOR  EQU #LEN+1           IFORM
#OPT  EQU #FOR+1           IOP
#NOD  EQU #ICR+1    "FLUSH" REQUEST: NODE NUMBER
*
* OFFSETS INTO RFA REPLY BUFFERS.
*
#RFD  EQU #REP      DCRET,DOPEN: RFAMD ENTRY #
#JSZ  EQU #RFD+1    DCRET: JSIZE (DXCRE)
#LOG  EQU #REP      DREAD: XLOG
```

```
#REC  EQU #REP      DLOCF: IREC
#RB   EQU #REC+1           IRB
#XRB  EQU #REC+2           IRB    (DXLOC)
#OFF  EQU #RB+1            IOFF
#XOF  EQU #XRB+2           IOFF   (DXLOC)
#JSC  EQU #OFF+1           JSECT
#XJS  EQU #XOF+1           JSECT  (DXLOC)
#JLU  EQU #JSC+1           JLU
#XJL  EQU #XJS+2           JLU    (DXLOC)
#JTY  EQU #JLU+1           JTY
#XJT  EQU #XJL+1           JTY    (DXLOC)
#JRC  EQU #JTY+1           JREC
#XJR  EQU #XJT+1           JREC   (DXLOC)
#IAD  EQU #REP      DSTAT: IADD
*
* MAXIMUM SIZE OF RFA REQUEST/REPLY BUFFER.
*
#RLW  EQU #MHD+13      M A X I M U M   S I Z E   ! ! !
```

Operator Request Block

```
*
*****************************************************************
*                                                               *
*      O P R E Q   B L O C K            REV 2013 791119         *
*                                                               *
*      OFFSETS INTO DS/1000-IV OPREQ MESSAGE BUFFERS, USED BY:  *
*                                                               *
*                  DMESS, OPERM, RQCNV, RPCNV                   *
*                  RSM,   DLGON, #MSSM, #UPSM                   *
*****************************************************************
*
* OFFSETS INTO OPREQ REQUEST AND REPLY BUFFERS.
*
#CML  EQU #REQ      COMMAND LENGTH.
#CMS  EQU #CML+1    COMMAND STRING.
#LGC  EQU #CMS+1    LOGON REQUEST CODE
#LNL  EQU #LGC+1    LENGTH OF USER NAME
#LUN  EQU #LNL+1    LOGON USER NAME
*
#RLN  EQU #REP      REPLY LENGTH.
#MSG  EQU #RLN+1    REPLY MESSAGE.
*
* MAXIMUM SIZE OF OPREQ REQUEST/REPLY BUFFER.
*
#OLW  EQU #CMS+23     M A X I M U M   S I Z E   ! ! !
*
```

Program-To-Program Communication Block

```
*
*******************************************************************
*                                                                 *
*     P T O P   B L O C K                    REV 2013 791119      *
*                                                                 *
*     OFFSETS INTO DS/1000-IV PTOP MESSAGE BUFFERS, USED BY:      *
*                                                                 *
*   POPEN, PTOPM, GET/ACEPT/REJCT, RQCNV, RPCNV, DINIT, REMAT     *
*   #SCSM                                                         *
*******************************************************************
*
* OFFSETS INTO PTOP REQUEST AND REPLY BUFFERS.
*
#FCD  EQU #REP      FUNCTION CODE.
#PCB  EQU #FCD+1    PCB AREA (3 WORDS).
#TAG  EQU #PCB+3    TAG AREA (20 WORDS).
*
* MAXIMUM SIZE OF PTOP REQUEST/REPLY BUFFER.
*
#PLW  EQU #MXR      M A X I M U M   S I Z E   ! ! !
*
```

Appendix B
1000 – 3000 Message Formats

Introduction

The diagrams in this appendix show the parameter information passed to and from an HP 3000. There are two or three diagrams for each call and command. The first two diagrams indicate the first block of data passed. The third diagram, when needed, indicates the second block of data. The intervening eight-word fixed format intermediate reply is not shown.

CAUTION

DO NOT ATTEMPT TO ACCESS NETWORK FACILITIES BY BUILDING AND SENDING COMMUNICATIONS MESSAGES. HEWLETT-PACKARD SUPPORTS ONLY THOSE NETWORK FACILITIES DEFINED IN THE DS/1000-IV USERS MANUAL, NOT AT ANY OTHER LEVEL.

The first eight words of the message (the header) have a fixed format. The meanings are shown in Figure B-1. Table B-1 is a summary of the message formats, in the order that they appear, and their message class and stream type.

The second portion of the DS/3000 message, which contains parameters specific to the function, is called the appendage. The final part of the message contains the data.

The left byte of the first word contains the number of words in the header and appendage areas. The last word of the header contains the number of bytes in the appendage and data areas.

Unless otherwise noted, all numbers are decimal. Also note that " : " indicates a continuation of the preceding line.

Abbreviations used in this appendix:

- AL - the length, in bytes, of the Appendage area
- DL - the length, in bytes, of the Data area

```
           REQUEST                              REPLY
  +----------------------+             +----------------------+
 0| L1   | Message Class |            0| L1   | Message Class |
 1| Reserved             |            1| Reserved             |
 2| Stream Word          |            2| Stream Word          |
 3| Reserved             |            3| Reserved             |
 4| From       |  To     |            4| From       | To      |
  | Process #| Process # |             | Process #| Process # |
 5| RTE Sequence Number  |            5| RTE Sequence Number  |
 6| Capability Mask Word |            6| Capability Mask Word |
 7| Byte Length (n)      |            7| Byte Length (n)      |
  +----------------------+             +----------------------+
  | Appendage, i.e.,     |             | Appendage, i.e.,     |
  |  calling parameters  |             |  return parameters   |
  +----------------------+             +----------------------+
  | User Data, i.e.,     |             | User Data, i.e.,     |
  | associated with an   |             | associated with an   |
  | FWRITE call          |             | FREAD call           |
  +----------------------+             +----------------------+
```

Figure B-1. FIXED FORMAT

n (word 7) = length in bytes of appendage + data

total length = fixed format + n = 16 + n bytes = (17+n)/2 words

L1 (left byte of word 0) - Length in words of fixed format (8 words) plus any apendage relating to Functional calls, excluding actual user data.

Stream Word - Bits of word 2 have the following meaning:

```
Bit  Meaning
---  -------
15   Request (off) or Reply (on)
14   Reject (on) else (off)
13   Continuation record to follow (on) else (off)
7-0  DS/3000 stream type
```

Process # - In a multi-programming environment is used to identify the process to which a particular stream type/operation belongs. The "From" number in a request is echoed as the "To" number in a reply.

Note: For master requests from the HP 1000 to the HP 3000, the "From Process #" is actually the console LU number specified by HELLO.

Capability Mask Word - Only used for initialization request/reply.

Bit	Meaning
15	Reply bit. Set in initialization reply if capability mask is valid for replying machine.
14	Set if sequence numbers are supported (not currently supported by DS/1000-IV).
13	Set if exclusive mode is supported without exclusive mode protocol (not currently supported by DS/1000-IV).
12	Set if $STDIN/$STDLIST continuation records are supported.

Table B-1. Commands, Message Classes, and Stream Types

MESSAGE CLASS	STREAM TYPE (OCTAL)	MEANING
0	20	Initialization *
	21	Termination *
3	20	REMOTE command (except Class 6 commands)
	22	DSLINE
4	22	PREAD
	23	PWRITE
	24	PCONTROL
	26	ACCEPT (Reply to PREAD/PWRITE/PCONTROL)
	27	REJECT (Reply to PREAD/PWRITE/PCONTROL)
5	20	$STDLIST to directed terminal
	21	Read request against master terminal (READ)
	23	Fcontrol request against master terminal
6	20	Remote HELLO
	21	Remote BYE
	22	BREAK *
	23	ABORT PROGRAM *
	24	RESUME *
	25	Control-Y *
	27	KILL JOB *
7	20	RFA Call (FS/3000)
	21	POPEN or PCLOSE request
	22	DSLINE
	26	PTOP accept of POPEN
	27	PTOP reject of POPEN
8	20	RTE FMP RFA
	21	Remote EXEC Call

* These message formats are not shown as they consist of only the 8-word header.

Remote Command: Class 3, Stream 20

```
              REQUEST                                    REPLY
  +---------------------+                  +---------------------+
 0| (17+AL)/2 |       3 |                 0|         8 |       3 |
 1|                     |                 1|                     |
 2|           | octal 20|                 2|           | octal 20|
 3|                     |                 3|                     |
 4|                     |                 4|                     |
 5|                     |                 5|                     |
 6|                     |                 6|                     |
 7|   AL + 0            |                 7|  0                  |
  +---------------------+                  +---------------------+
 8| Remote Command      |
  | (ASCII String)      |
  |    :                |
  |    :                |
  |    :                |
  +---------------------+
```

PREAD: Class 4

Request: Stream 22
Accept Reply: Stream 26
Reject Reply: Stream 27

```
         REQUEST                                  REPLY
  +---------------------+              +---------------------+
 0|      33 |       4   |             0|      33 |        4  |
 1|                     |             1|                     |
 2|         | octal 22  |             2|         | 26 or 27  |
 3|                     |             3|                     |
 4|                     |             4|                     |
 5|                     |             5|                     |
 6|                     |             6|                     |
 7|  50 + 0             |             7|  50 + DL            |
  +---------------------+              +---------------------+
 0|                     |             0| Error Code          |
 1| Tcount              |             1| Tcount'             |
 2| Slave Program ID    |             2| Slave Program ID    |
 3| Slave I/O Class     |             3| Slave I/O Class     |
 4|                     |             4|                     |
 5| Tag Field           |             5| Tag Field           |
  |     :               |              |     :               |
  |     : (20           |              |     : (20           |
  |     :  words        |              |     :  words        |
  |     :   total)      |              |     :   total)      |
24|     :               |            24|     :               |
  +---------------------+              +---------------------+
                                     25| Data                |
                                       |  Words              |
                                       |                     |
                                       |   (Accept only)     |
                                       |                     |
                                       |                     |
                                       |                     |
                                       |                     |
                                       |                     |
                                       |                     |
                                       |                     |
                                       +---------------------+
```

Note:

Tcount specifies the requested read length.

Tcount' specified the actual amount read.

CONTINUATION RECORD
(ACCEPT ONLY)

REPLY

```
 +---------------------+
0|       13 |        4 |
1|                     |
2|          | octal 26 |
3|                     |
4|                     |
5|                     |
6|                     |
7|  10 + DL            |
 +---------------------+
0|                     |
1| Tcount'             |
2| Slave Program ID    |
3| Slave I/O Class     |
4|                     |
 +---------------------+
5| Data                |
 |  Words              |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 +---------------------+
```

PWRITE: Class 4

Request: Stream 23
Accept Reply: Stream 26
Reject Reply: Stream 27

```
            REQUEST                                  REPLY
  +---------------------+               +---------------------+
 0|      33 |         4 |              0|      13 |         4 |
 1|                     |              1|                     |
 2|         | octal 23  |              2|         | 26 or 27  |
 3|                     |              3|                     |
 4|                     |              4|                     |
 5|                     |              5|                     |
 6|                     |              6|                     |
 7|   50 + DL           |              7|   50 + 0            |
  +---------------------+               +---------------------+
 0|                     |              0| Error Code          |
 1| Tcount              |              1|                     |
 2| Slave Program ID    |              2| Slave Program ID    |
 3| Slave I/O Class     |              3| Slave I/O Class     |
 4|                     |              4|                     |
 5| Tag Field           |              5| Tag Field           |
  |     :               |               |     :               |
  |     : (20           |               |     : (20           |
  |     :  words        |               |     :  words        |
  |     :   total)      |               |     :   total)      |
24|     :               |             24|     :               |
  +---------------------+               +---------------------+
25| Data                |
  |  Words              |
  |   (Accept only)     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  +---------------------+
```

CONTINUATION RECORD
(ACCEPT ONLY)

```
          REQUEST
  +----------------------+
 0|        13 |        4 |
 1|                      |
 2|           | octal 26 |
 3|                      |
 4|                      |
 5|                      |
 6|                      |
 7|    10 + DL           |
  +----------------------+
 0|                      |
 1| Tcount               |
 2| Slave Program ID     |
 3| Slave I/O Class      |
 4|                      |
  +----------------------+
 5| Data                 |
  |  Words               |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  +----------------------+
```

PCONTROL: Class 4

Request: Stream 24
Accept Reply: Stream 26
Reject Reply: Stream 27

```
             REQUEST                                 REPLY
  +----------------------+              +----------------------+
 0|        33 |        4 |             0|        33 |        4 |
 1|                      |             1|                      |
 2|           | octal 24 |             2|           | 26 or 27 |
 3|                      |             3|                      |
 4|                      |             4|                      |
 5|                      |             5|                      |
 6|                      |             6|                      |
 7|    50 + 0            |             7|    50 + 0            |
  +----------------------+              +----------------------+
 0|                      |             0| Error Code           |
 1|                      |             1|                      |
 2| Slave Program ID     |             2| Slave Program ID     |
 3| Slave I/O Class      |             3| Slave I/O Class      |
 4|                      |             4|                      |
 5| Tag Field            |             5| Tag Field            |
  |     :                |              |     :                |
  |     : (20            |              |     : (20            |
  |     :  words         |              |     :  words         |
  |     :   total)       |              |     :   total)       |
24|     :                |            24|     :                |
  +----------------------+              +----------------------+
```

$STDLIST (PRINT): Class 5, Stream 20

```
            REQUEST
 +---------------------+
0|       10 |        5 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7| Write Length+2      |
 +---------------------+
0| Control*            |
1| Blockmode indication|
2| Characters          |
 |  to be              |
 |   printed.          |
 |      :              |
 |      :              |
 |      :              |
 +---------------------+
```

```
      CONTINUATION REQUEST
 +---------------------+
0|        8 |        5 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|    Data Length      |
 +---------------------+
0| Characters to be    |
1|  to be              |
2|   printed.          |
 |      :              |
 |      :              |
 |      :              |
 +---------------------+
```

```
             REPLY
 +---------------------+
0|        9 |        5 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|  2 + 0              |
 +---------------------+
0| Status Word         |
 +---------------------+
```

```
       CONTINUATION REPLY
 +---------------------+
0|        8 |        5 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|                     |
 +---------------------+
```

*Refer to the MPE Intrinsics Manual under carriage control from more information.

$STD IN (READ/READX): Class 5, Stream 21

```
          REQUEST                              REPLY
 +---------------------+            +---------------------+
0|         11 |     5  |           0|         10 |     5  |
1|                     |           1|                     |
2|            | octal 21|          2|            | octal 21|
3|                     |           3|                     |
4|                     |           4|                     |
5|                     |           5|                     |
6|                     |           6|                     |
7|    6 + 0            |           7| Read Length+4       |
 +---------------------+            +---------------------+
0| Length              |           0| Status              |
1| Unused              |           1| Read Length         |
2| Blockmode indication|           2| Characters          |
 +---------------------+            |  that were          |
                                    |   read.             |
                                    |      :              |
                                    |      :              |
                                    +---------------------+

   CONTINUATION REQUEST              CONTINUATION REPLY
 +---------------------+            +---------------------+
0|          8|     5   |           0|          8 |     5  |
1|                     |           1|                     |
2|           | octal 21 |          2|            | octal 21|
3|                     |           3|                     |
4|                     |           4|                     |
5|                     |           5|                     |
6|                     |           6|                     |
7|  0                  |           7|  Read Length        |
 +---------------------+            +---------------------+
                                   0| Characters          |
                                   1|  that were          |
                                   2|   read.             |
                                    |      :              |
                                    |      :              |
                                    |      :              |
                                    +---------------------+
```

FCONTROL for $STDIN/$STDLIST: Class 5, Stream 23

```
           REQUEST                              REPLY
 +---------------------+              +---------------------+
0|        11 |       5 |             0|        10 |       5 |
1|                     |             1|                     |
2|           | octal 23 |            2|           | octal 23 |
3|                     |             3|                     |
4|                     |             4|                     |
5|                     |             5|                     |
6|                     |             6|                     |
7|   6 + 0             |             7|   4 + 0             |
 +---------------------+              +---------------------+
0| Filenum             |             0| Status              |
1| Control Code        |             1| Param               |
2| Param               |              +---------------------+
 +---------------------+
```

Remote HELLO: Class 6, Stream 20

```
            REQUEST                                   REPLY
  +----------------------+                +----------------------+
 0|(17+AL)/2 |        6  |               0|         8 |        6 |
 1|                      |               1|                      |
 2|          | octal 20  |               2|           | octal 20 |
 3|                      |               3|                      |
 4|                      |               4|                      |
 5|                      |               5|                      |
 6|                      |               6|                      |
 7|  AL + 0              |               7|  0                   |
  +----------------------+                +----------------------+
 0| MPE HELLO Command    |
  |       :              |
  |       :              |
  |       :              |
  +----------------------+
```

Remote BYE: Class 6, Stream 21

```
            REQUEST
 +---------------------+
0|       10 |        6 |
1|                     |
2|          | octal 21 |
3|                     |
4|                     |
5|                     |
6|                     |
7|   4 + 0             |
 +---------------------+
0| ASCII "BY"          |
1| ASCII "E "          |
 +---------------------+
```

```
            REPLY
 +---------------------+
0|        8 |        6 |
1|                     |
2|          | octal 21 |
3|                     |
4|                     |
5|                     |
6|                     |
7| 0                   |
 +---------------------+
```

FOPEN: Class 7, Stream 20

File System Intrinsic Number 1

```
           REQUEST
   +---------------------+
  0|(17+AL)/2 |        7 |
  1|                     |
  2|          | octal 20 |
  3|                     |
  4|                     |
  5|                     |
  6|                     |
  7|  AL + 0             |
   +---------------------+
  0| ASCII "RF"          |
  1| ASCII "A "          |
  2| Intrinsic No. = 1   |
  3| @Formaldesignator   |
  4| Foptions            |
  5| Aoptions            |
  6| Recsize             |
  7| @Device Name        |
  8| @Formmsg            |
  9| Userlabels          |
 10| Blockfactor         |
 11| Numbuffers          |
 12| Filesize            |
 13|   :                 |
 14| Numextents          |
 15| Initialloc          |
 16| Filecode            |
 17|B|K| mask            |
   |---------------------|
   | Formaldesignator    |
   | (variable length)   |
   |                     |
   |---------------------|
   |                     |
   | Device Name         |
   | (variable length)   |
   |                     |
   |---------------------|
   |                     |
   | Formmsg             |
   | (variable length)   |
   |                     |
   +---------------------+
```

```
            REPLY
   +---------------------+
  0|       10 |        7 |
  1|                     |
  2|          | octal 20 |
  3|                     |
  4|                     |
  5|                     |
  6|                     |
  7|   4 + 0             |
   +---------------------+
  0| Status Word         |
  1| mask                |
   +---------------------+
```

Right byte of status word contains the file number.

@ = byte pointer (relative to n) to area within the request buffer.

B of mask is used to pass the privleged/non-privleged status of the code calling FOPEN on the local side (sign bit).

K of mask is used to signal that the secondary entry point of KSOPEN was called to do the FOPEN.

In all RFA requests, appendage words 0 and 1 contain the ASCII code for "RFA".

FCLOSE: Class 7, Stream 20

File System Intrinsic Number 2

```
            REQUEST                                 REPLY
  +---------------------+               +---------------------+
 0|          14 |      7 |             0|          9 |       7 |
 1|                     |              1|                     |
 2|             | octal 20 |           2|            | octal 20 |
 3|                     |              3|                     |
 4|                     |              4|                     |
 5|                     |              5|                     |
 6|                     |              6|                     |
 7|   12 + 0            |              7|    2 + 0            |
  +---------------------+               +---------------------+
 0| ASCII "RF"          |              0| Status Word         |
 1| ASCII "A "          |               +---------------------+
 2| Intrinsic No. = 2   |
 3| Filenum             |
 4| Disposition         |
 5| Seccode             |
  +---------------------+
```

FREAD (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 3

```
              REQUEST                                       REPLY
  +---------------------+                      +---------------------+
 0|      13 |         7 |                     0|      10 |         7 |
 1|                     |                     1|                     |
 2|         | octal 20  |                     2|         | octal 20  |
 3|                     |                     3|                     |
 4|                     |                     4|                     |
 5|                     |                     5|                     |
 6|                     |                     6|                     |
 7|  10 + 0             |                     7|  4 + DL             |
  +---------------------+                      +---------------------+
 0| ASCII "RF"          |                     0| Status Word         |
 1| ASCII "A "          |                     1| Tcount'             |
 2| ERR #   |        3  |                      +---------------------+
 3| Filenum             |                     2|                     |
 4| Tcount              |                      | Data                |
  +---------------------+                      |   Words             |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               |                     |
                                               +---------------------+
```

ERR #: Used when local system finds an error in a file. ERR # passed to remote system.

```
        CONTINUATION
           RECORD
          (REPLY)
  +----------------------+
 0|         8 |        7 |
 1|                      |
 2|           | octal 20 |
 3|                      |
 4|                      |
 5|                      |
 6|                      |
 7|   0 + DL             |
  +----------------------+
 0|                      |
  |  Data                |
  |    Words             |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  +----------------------+
```

FREADDIR (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 4

```
           REQUEST                                 REPLY
  +----------------------+            +----------------------+
 0|       15 |        7  |           0|       10 |        7  |
 1|                      |           1|                      |
 2|          | octal 20  |           2|          | octal 20  |
 3|                      |           3|                      |
 4|                      |           4|                      |
 5|                      |           5|                      |
 6|                      |           6|                      |
 7|   14 + 0             |           7|   4 + DL             |
  +----------------------+            +----------------------+
 0| ASCII "RF"           |           0| Status Word          |
 1| ASCII "A "           |           1| Tcount'              |
 2| ERR #    |     4     |            +----------------------+
 3| Filenum              |           2|                      |
 4| Tcount               |            | Data                 |
 5| Recnum               |            |   Words              |
 6|   :                  |            |                      |
  +----------------------+            |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      |                      |
                                      +----------------------+
```

```
        CONTINUATION
           RECORD
          (REPLY)
  +----------------------+
 0|         8 |        7 |
 1|                      |
 2|           | octal 20 |
 3|                      |
 4|                      |
 5|                      |
 6|                      |
 7|   0 + DL             |
  +----------------------+
 0|                      |
  |  Data                |
  |    Words             |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  +----------------------+
```

FREADSEEK: Class 7, Stream 20

File System Intrinsic Number 5

```
           REQUEST                                   REPLY
  +----------------------+                +----------------------+
 0|        14 |        7 |               0|         9 |        7 |
 1|                      |               1|                      |
 2|           | octal 20 |               2|           | octal 20 |
 3|                      |               3|                      |
 4|                      |               4|                      |
 5|                      |               5|                      |
 6|                      |               6|                      |
 7|  12 + 0              |               7|  2 + 0               |
  +----------------------+                +----------------------+
 0| ASCII "RF"           |               0| Status Word          |
 1| ASCII "A "           |                +----------------------+
 2| Intrinsic No. = 5    |
 3| Filenum              |
 4| Recnum               |
 5|   :                  |
  +----------------------+
```

FWRITE (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 6

```
             REQUEST                              REPLY
   +---------------------+            +---------------------+
  0|        14 |       7 |           0|         9 |       7 |
  1|                     |           1|                     |
  2|           | octal 20|           2|           | octal 20|
  3|                     |           3|                     |
  4|                     |           4|                     |
  5|                     |           5|                     |
  6|                     |           6|                     |
  7|  12 + DL            |           7|   2 + 0             |
   +---------------------+            +---------------------+
  0| ASCII "RF"          |           0| Status Word         |
  1| ASCII "A "          |            +---------------------+
  2| ERR #     |    6    |
  3| Filenum             |
  4| Tcount              |
  5| Control             |
   +---------------------+
  6|                     |
   | Data                |
   |   Words             |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   |                     |
   +---------------------+
```

```
        CONTINUATION
           RECORD
          (REQUEST)
  +---------------------+
 0|         8 |        7 |
 1|                      |
 2|           | octal 20 |
 3|                      |
 4|                      |
 5|                      |
 6|                      |
 7|   0 + DL             |
  +---------------------+
 0|                      |
  | Data                 |
  |   Words              |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  +---------------------+
```

FWRITEDIR (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 7

```
           REQUEST                              REPLY
 +---------------------+            +---------------------+
0|       15 |        7 |           0|        9 |        7 |
1|                     |           1|                     |
2|          | octal 20 |           2|          | octal 20 |
3|                     |           3|                     |
4|                     |           4|                     |
5|                     |           5|                     |
6|                     |           6|                     |
7|  14 + DL            |           7|   2 + 0             |
 +---------------------+            +---------------------+
0| ASCII "RF"          |           0| Status Word         |
1| ASCII "A "          |            +---------------------+
2| ERR #    |     7    |
3| Filenum             |
4| Tcount              |
5| Recnum              |
6|   :                 |
 +---------------------+
7|                     |
 | Data                |
 |   Words             |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 +---------------------+
```

```
      CONTINUATION
         RECORD
       (REQUEST)
 +---------------------+
0|        8 |        7 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|  0 + DL             |
 +---------------------+
0|                     |
 | Data                |
 |   Words             |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 +---------------------+
```

FREADLABEL: Class 7, Stream 20

File System Intrinsic Number 8

```
             REQUEST                                  REPLY
  +---------------------+                +---------------------+
 0|        15 |       7 |               0|(17+AL)/2 |       7 |
 1|                     |               1|                     |
 2|           | octal 20 |              2|          | octal 20 |
 3|                     |               3|                     |
 4|                     |               4|                     |
 5|                     |               5|                     |
 6|                     |               6|                     |
 7|  14 + 0             |               7| (Label Length*2)+2  |
  +---------------------+                +---------------------+
 0| ASCII "RF"          |               0| Status Word         |
 1| ASCII "A "          |               1| Label (Up to        |
 2| Intrinsic 10(octal) |                |   128 words)        |
 3| Filenum             |                |      :              |
 4| Tcount              |                |      :              |
 5| Labelid             |                |      :              |
 6| mask                |                |      :              |
  +---------------------+                |      :              |
                                         |      :              |
                                         |      :              |
                                         +---------------------+
```

FWRITELABEL: Class 7, Stream 20

File System Intrinsic Number 9

```
        REQUEST                                   REPLY
 +---------------------+               +---------------------+
0|(17+AL)/2 |        7 |              0|         9 |        7 |
1|                     |              1|                     |
2|          | octal 20 |              2|           | octal 20 |
3|                     |              3|                     |
4|                     |              4|                     |
5|                     |              5|                     |
6|                     |              6|                     |
7|  Label Length+14    |              7|   2 + 0             |
 +---------------------+               +---------------------+
0| ASCII "RF"          |              0| Status Word         |
1| ASCII "A "          |               +---------------------+
2| Intrinsic 11(octal) |
3| Filenum             |
4| Tcount              |
5| Labelid             |
6| mask                |
7| Label (Up to        |
 |   128 words)        |
 |     :               |
 |     :               |
 |     :               |
 |     :               |
 |     :               |
 |     :               |
 +---------------------+
```

FUPDATE (Not Multirecord): Class 7, Stream 20

File System Intrinsic Number 10

```
             REQUEST                                  REPLY
  +---------------------+                  +---------------------+
 0|       13 |       7  |                 0|        9 |       7  |
 1|                     |                 1|                     |
 2|          | octal 20 |                 2|          | octal 20 |
 3|                     |                 3|                     |
 4|                     |                 4|                     |
 5|                     |                 5|                     |
 6|                     |                 6|                     |
 7|  10 + DL            |                 7|   2 + 0             |
  +---------------------+                  +---------------------+
 0| ASCII "RF"          |                 0| Status Word         |
 1| ASCII "A "          |                  +---------------------+
 2| Intrinsic 12(octal) |
 3| Filenum             |
 4| Tcount              |
  +---------------------+
 5|                     |
  | Data                |
  |   Words             |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  +---------------------+
```

```
           CONTINUATION
              RECORD
            (REQUEST)
   +----------------------+
  0|          8 |        7 |
  1|                       |
  2|            | octal 20 |
  3|                       |
  4|                       |
  5|                       |
  6|                       |
  7|  0 + DL               |
   +----------------------+
  0|                       |
   | Data                  |
   |   Words               |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   |                       |
   +----------------------+
```

FSPACE: Class 7, Stream 20

File System Intrinsic Number 11

```
             REQUEST                                   REPLY
  +---------------------+                 +---------------------+
 0|        13 |       7 |                0|         9 |       7 |
 1|                     |                1|                     |
 2|           | octal 20 |               2|           | octal 20 |
 3|                     |                3|                     |
 4|                     |                4|                     |
 5|                     |                5|                     |
 6|                     |                6|                     |
 7|   10 + 0            |                7|   2 + 0             |
  +---------------------+                 +---------------------+
 0| ASCII "RF"          |                0| Status Word         |
 1| ASCII "A "          |                 +---------------------+
 2| Intrinsic 13(octal) |
 3| Filenum             |
 4| Displacement        |
  +---------------------+
```

FPOINT: Class 7, Stream 20

File System Intrinsic Number 12

```
          REQUEST                                     REPLY
  +---------------------+                   +---------------------+
 0|       14 |        7 |                  0|        9 |        7 |
 1|                     |                  1|                     |
 2|          | octal 20 |                  2|          | octal 20 |
 3|                     |                  3|                     |
 4|                     |                  4|                     |
 5|                     |                  5|                     |
 6|                     |                  6|                     |
 7|   12 + 0            |                  7|   2 + 0             |
  +---------------------+                   +---------------------+
 0| ASCII "RF"          |                  0| Status Word         |
 1| ASCII "A "          |                   +---------------------+
 2| Intrinsic 14(octal) |
 3| Filenum             |
 4| Recnum              |
 5|   :                 |
  +---------------------+
```

FGETINFO: Class 7, Stream 20

File System Intrinsic Number 13

```
           REQUEST                                   REPLY
  +----------------------+                  +----------------------+
 0|       14 |         7 |                 0|       52 |         7 |
 1|                      |                 1|                      |
 2|          | octal 20  |                 2|          | octal 20  |
 3|                      |                 3|                      |
 4|                      |                 4|                      |
 5|                      |                 5|                      |
 6|                      |                 6|                      |
 7|  12 + 0              |                 7|   88 + 0             |
  +----------------------+                  +----------------------+
 0| ASCII "RF"           |                 0| Status Word          |
 1| ASCII "A "           |                 1| File name (words     |
 2| Intrinsic 15(octal)  |                  |  4 through 14)       |
 3| Filenum              |                  |     :                |
 4| mask                 |                15| Foptions             |
 5|   :                  |                16| Aoptions             |
  +----------------------+                17| Recsize              |
                                          18| Devtype              |
                                          19| Ldnum                |
                                          20| Hdaddr               |
                                          21| Filecode             |
                                          22| Recpt                |
                                          23|    :                 |
                                          24| EOF                  |
                                          25|  :                   |
                                          26| Flimit               |
                                          27|    :                 |
                                          28| Logcount             |
                                          29|    :                 |
                                          30| Physcount            |
                                          31|    :                 |
                                          32| Blksize              |
                                          33| Extsize              |
                                          34| Numextents           |
                                          35| Userlabels           |
                                          36| Creator ID           |
                                          37|    :                 |
                                          38|    :                 |
                                          39|    :                 |
                                          40| Labaddr              |
                                          41|    :                 |
                                          42| mask                 |
                                          43|    :                 |
                                            +----------------------+
```

FCHECK: Class 7, Stream 20

File System Intrinsic Number 14

```
            REQUEST                                  REPLY
  +---------------------+                 +---------------------+
 0|        13 |       7 |                0|        15 |       7 |
 1|                     |                1|                     |
 2|           | octal 20|                2|           | octal 20|
 3|                     |                3|                     |
 4|                     |                4|                     |
 5|                     |                5|                     |
 6|                     |                6|                     |
 7|   10 + 0            |                7|   14 + 0            |
  +---------------------+                 +---------------------+
 0| ASCII "RF"          |                0| Status Word         |
 1| ASCII "A "          |                1| Errorcode           |
 2| Intrinsic 16(octal) |                2| Tlog                |
 3| Filenum             |                3| Blknum              |
 4| mask                |                4|    :                |
  +---------------------+                5| Numrec              |
                                         6| mask                |
                                          +---------------------+
```

FCONTROL: Class 7, Stream 20

File System Intrinsic Number 15

```
          REQUEST                                REPLY
  +---------------------+               +---------------------+
 0|        14 |       7 |              0|        10 |       7 |
 1|                     |              1|                     |
 2|           | octal 20 |             2|           | octal 20 |
 3|                     |              3|                     |
 4|                     |              4|                     |
 5|                     |              5|                     |
 6|                     |              6|                     |
 7|  12 + 0             |              7|   4 + 0             |
  +---------------------+               +---------------------+
 0| ASCII "RF"          |              0| Status Word         |
 1| ASCII "A "          |              1| Param               |
 2| Intrinsic 17(octal) |               +---------------------+
 3| Filenum             |
 4| Control Code        |
 5| Param               |
  +---------------------+
```

FSETMODE: Class 7, Stream 20

File System Intrinsic Number 16

```
              REQUEST                                      REPLY
   +----------------------+                  +----------------------+
  0|         13 |       7 |                 0|          9 |       7 |
  1|                      |                 1|                      |
  2|            | octal 20|                 2|            | octal 20|
  3|                      |                 3|                      |
  4|                      |                 4|                      |
  5|                      |                 5|                      |
  6|                      |                 6|                      |
  7|   10 + 0             |                 7|    2 + 0             |
   +----------------------+                  +----------------------+
  0| ASCII "RF"           |                 0| Status Word          |
  1| ASCII "A "           |                  +----------------------+
  2| Intrinsic 20(octal)  |
  3| Filenum              |
  4| Modeflags            |
   +----------------------+
```

FRENAME: Class 7, Stream 20

File System Intrinsic Number 17

```
          REQUEST                                 REPLY
  +---------------------+               +---------------------+
 0|(AL+17)/2 |         7|              0|          9 |        7|
 1|                     |              1|                     |
 2|          | octal 20 |              2|            | octal 20|
 3|                     |              3|                     |
 4|                     |              4|                     |
 5|                     |              5|                     |
 6|                     |              6|                     |
 7| Name Length+8       |              7|   2 + 0             |
  +---------------------+               +---------------------+
 0| ASCII "RF"          |              0| Status Word         |
 1| ASCII "A "          |               +---------------------+
 2| Intrinsic 21(octal) |
 3| Filenum             |
 4| New File Name       |
 5|  (Up to 28          |
  |   characters)       |
  |      :              |
  |      :              |
  |      :              |
  |      :              |
  |      :              |
  |      :              |
  |      :              |
  +---------------------+
```

FRELATE: Class 7, Stream 20

File System Intrinsic Number 18

```
          REQUEST
 +---------------------+
0|       13 |        7 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|  10 + 0             |
 +---------------------+
0| ASCII "RF"          |
1| ASCII "A "          |
2| Intrinsic 22(octal) |
3| Infilenum           |
4| Listfilenum         |
 +---------------------+
```

```
          REPLY
 +---------------------+
0|       10 |        7 |
1|                     |
2|          | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|  4 + 0              |
 +---------------------+
0| Status Word         |
1| reply               |
 +---------------------+
```

FLOCK: Class 7, Stream 20

File System Intrinsic Number 19

```
           REQUEST                                 REPLY
  +----------------------+              +----------------------+
 0|        15 |        7 |             0|        11 |        7 |
 1|                      |             1|                      |
 2|           | octal 20 |             2|           | octal 20 |
 3|                      |             3|                      |
 4|                      |             4|                      |
 5|                      |             5|                      |
 6|                      |             6|                      |
 7|    14 + 0            |             7|    6 + 0             |
  +----------------------+              +----------------------+
 0| ASCII "RF"           |             0| Status Word          |
 1| ASCII "A "           |             1| Q - 7                |
 2| Intrinsic 23(octal)  |             2| Q - 6                |
 3| Filenum              |              +----------------------+
 4| Lockcond             |
 5| Q - 7                |
 6| Q - 6                |
  +----------------------+
```

FUNLOCK: Class 7, Stream 20

File System Intrinsic Number 20

```
            REQUEST                                  REPLY
  +----------------------+                +----------------------+
 0|        14 |        7 |               0|         9 |        7 |
 1|                      |               1|                      |
 2|           | octal 20 |               2|           | octal 20 |
 3|                      |               3|                      |
 4|                      |               4|                      |
 5|                      |               5|                      |
 6|                      |               6|                      |
 7|  12 + 0              |               7|   2 + 0              |
  +----------------------+                +----------------------+
 0| ASCII "RF"           |               0| Status Word          |
 1| ASCII "A "           |                +----------------------+
 2| Intrinsic 24(octal)  |
 3| Filenum              |
 4| Q - 7                |
 5| Q - 6                |
  +----------------------+
```

POPEN: Class 7

Request: Stream 24
Accept Reply: Stream 26
Reject Reply: Stream 27
File System Intrinsic Number 21

```
              REQUEST
    +---------------------+
   0|        56 |        7 |
   1|                      |
   2|           | octal 21 |
   3|                      |
   4|                      |
   5|                      |
   6|                      |
   7|  96 + 0              |
    +---------------------+
   0| ASCII "RF"          |
   1| ASCII "A "          |
   2| Intrinsic 25(octal) |
   3| Program Name        |
    |  (14 words)         |
    |     :               |
    |     :               |
  17| Entry Point         |
  18|  (4 words)          |
  19|      :              |
  20|      :              |
  21| Tag Field           |
    |      :              |
    |      : (20          |
    |      :  words       |
    |      :   total)     |
  40|      :              |
  41| Param               |
  42| Flags               |
  43| Stacksize           |
  44| DL Size             |
  45| Maxdata             |
  46| Bufsize             |
  47| mask                |
    +---------------------+
```

```
              REPLY
    +---------------------+
   0|        33 |        7 |
   1|             octal    |
   2|           | 26 or 27 |
   3|                      |
   4|                      |
   5|                      |
   6|                      |
   7|  50 + 0              |
    +---------------------+
   0| Error Code          |
   1|                     |
   2| Slave Program ID    |
   3| Slave I/O Class     |
   4|                     |
   5| Tag Field           |
    |      :              |
    |      : (20          |
    |      :  words       |
    |      :   total)     |
  25| mask                |
    +---------------------+
```

"mask" indicates which parameters were in provided (bit = 0 means use default):

9 - Program Name
8 - Entry Point
6 - Param
5 - Flags
4 - Stacksize
3 - DL Size
2 - Maxdata

PCLOSE: Class 7, Stream 21

File System Intrinsic Number 22

```
          REQUEST                                   REPLY
 +---------------------+                  +---------------------+
0|        14 |       7 |                 0|         8 |       7 |
1|                     |                 1|                     |
2|           | octal 21|                 2|           | octal 21|
3|                     |                 3|                     |
4|                     |                 4|                     |
5|                     |                 5|                     |
6|                     |                 6|                     |
7|   12 + 0            |                 7|    0 + 0            |
 +---------------------+                  +---------------------+
0| ASCII "RF"          |
1| ASCII "A "          |
2| Intrinsic 26(octal) |
3| Slave Program ID    |
4| Slave I/O Class     |
5|                     |
 +---------------------+
```

DCRET: Class 8, Stream 20

Intrinsic Number 150

```
            REQUEST                                      REPLY
   +---------------------+                    +---------------------+
  0|         20 |       8 |                  0|         11 |       8 |
  1|                      |                  1|                      |
  2|            | octal 20 |                 2|            | octal 20 |
  3|                      |                  3|                      |
  4|                      |                  4|                      |
  5|                      |                  5|                      |
  6|                      |                  6|                      |
  7|  24 + 0              |                  7|    6 + 0             |
   +---------------------+                    +---------------------+
  0| Intrinsic 226(octal)|                   0| A-Register           |
  1|                      |                  1| RFAM Entry Number    |
  2|                      |                  2| IERR                 |
  3|                      |                   +---------------------+
  4| Name                 |
  5|   :                  |
  6|   :                  |
  7| ISIZE(1)             |
  8| ISIZE(2)             |
  9| File Type            |
 10| Security Code        |
 11| Cartridge Number     |
   +---------------------+
```

DPURG: Class 8, Stream 20

Intrinsic Number 151

```
            REQUEST
  +---------------------+
 0|       17 |        8 |
 1|                     |
 2|          | octal 20 |
 3|                     |
 4|                     |
 5|                     |
 6|                     |
 7|   18 + 0            |
  +---------------------+
 0| Intrinsic 227(octal)|
 1|                     |
 2|                     |
 3|                     |
 4| Name                |
 5|  :                  |
 6|  :                  |
 7| Security Code       |
 8| Cartridge Number    |
  +---------------------+
```

```
             REPLY
  +---------------------+
 0|       11 |        8 |
 1|                     |
 2|          | octal 20 |
 3|                     |
 4|                     |
 5|                     |
 6|                     |
 7|   6 + 0             |
  +---------------------+
 0| A-Register          |
 1| 0                   |
 2| IERR                |
  +---------------------+
```

DOPEN: Class 8, Stream 20

Intrinsic Number 152

```
        REQUEST                              REPLY
 +---------------------+             +---------------------+
0|       18 |        8 |            0|       11 |        8 |
1|                     |            1|                     |
2|          | octal 20 |            2|          | octal 20 |
3|                     |            3|                     |
4|                     |            4|                     |
5|                     |            5|                     |
6|                     |            6|                     |
7|  20 + 0             |            7|   6 + 0             |
 +---------------------+             +---------------------+
0| Intrinsic 230(octal)|            0| A-Register          |
1|                     |            1| RFAM Entry Number   |
2|                     |            2| IERR                |
3|                     |             +---------------------+
4| Name                |
5|   :                 |
6|   :                 |
7| Options             |
8| Security Code       |
9| Cartridge Number    |
 +---------------------+
```

DWRIT: Class 8, Stream 20

Intrinsic Number 153

```
           REQUEST                                   REPLY
  +---------------------+                  +---------------------+
 0|        17 |        8 |                0|        11 |        8 |
 1|                      |                1|                      |
 2|           | octal 20 |                2|           | octal 20 |
 3|                      |                3|                      |
 4|                      |                4|                      |
 5|                      |                5|                      |
 6|                      |                6|                      |
 7|  18 + DL             |                7|   6 + 0              |
  +---------------------+                  +---------------------+
 0| Intrinsic 231(octal)|                 0| A-Register           |
 1|                      |                1| 0                    |
 2| Number Data Words    |                2| IERR                 |
 3|                      |                 +---------------------+
 4| DCB                  |
 5|  :                   |
 6|  :                   |
 7| IL                   |
 8| NUM                  |
  +---------------------+
 9|                      |
  |  Data                |
  |   Words              |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  |                      |
  +---------------------+
```

```
      CONTINUATION
         RECORD
       (REQUEST)
  +----------------------+
 0|         8 |         8 |
 1|                       |
 2|           | octal 20  |
 3|                       |
 4|                       |
 5|                       |
 6|                       |
 7|   0 + DL              |
  +----------------------+
 0|                       |
  |   Data                |
  |    Words              |
  |                       |
  |                       |
  |                       |
  |                       |
  |                       |
  |                       |
  |                       |
  |                       |
  +----------------------+
```

DREAD: Class 8, Stream 20

Intrinsic Number 154

```
          REQUEST
 +---------------------+
0|        18 |       8 |
1|                     |
2|           | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|   20 + 0            |
 +---------------------+
0| Intrinsic 232(octal)|
1|                     |
2| Number of Data Words|
3|                     |
4| DCB                 |
5|   :                 |
6|   :                 |
7| IL                  |
8|                     |
9| NUM                 |
 +---------------------+
```

```
        FIRST RECORD
          (REPLY)
 +---------------------+
0|         8 |       8 |
1|                     |
2|           | octal 20 |
3|                     |
4|                     |
5|                     |
6|                     |
7|   0 + DL            |
 +---------------------+
0| Data                |
 |  Words              |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 |                     |
 +---------------------+
```

```
            LAST
           RECORD
          (REPLY)
 +----------------------+
0|        12 |         8 |
1|                      |
2|           | octal 20 |
3|                      |
4|                      |
5|                      |
6|                      |
7|    8 + 0             |
 +----------------------+
0| A-Register           |
1| 0                    |
2| IERR                 |
3| LEN                  |
 +----------------------+
```

DPOSN: Class 8, Stream 20

Intrinsic Number 155

```
           REQUEST                                 REPLY
  +----------------------+              +----------------------+
 0|        17 |        8 |             0|        11 |        8 |
 1|                      |             1|                      |
 2|           | octal 20 |             2|           | octal 20 |
 3|                      |             3|                      |
 4|                      |             4|                      |
 5|                      |             5|                      |
 6|                      |             6|                      |
 7|   18 + 0             |             7|   6 + 0              |
  +----------------------+              +----------------------+
 0| Intrinsic 233(octal) |             0| A-Register           |
 1|                      |             1| 0                    |
 2|                      |             2| IERR                 |
 3|                      |              +----------------------+
 4| DCB                  |
 5|   :                  |
 6|   :                  |
 7| NUR                  |
 8| IR                   |
  +----------------------+
```

DWIND: Class 8, Stream 20

Intrinsic Number 156

```
           REQUEST                                      REPLY
  +---------------------+                     +---------------------+
 0|        15 |       8 |                    0|        11 |       8 |
 1|                     |                    1|                     |
 2|           | octal 20 |                   2|           | octal 20 |
 3|                     |                    3|                     |
 4|                     |                    4|                     |
 5|                     |                    5|                     |
 6|                     |                    6|                     |
 7|   14 + 0            |                    7|   6 + 0             |
  +---------------------+                     +---------------------+
 0| Intrinsic 234(octal)|                    0| A-Register          |
 1|                     |                    1| 0                   |
 2|                     |                    2| IERR                |
 3|                     |                     +---------------------+
 4| DCB                 |
 5|   :                 |
 6|   :                 |
  +---------------------+
```

DCLOS: Class 8, Stream 20

Intrinsic Number 157

```
           REQUEST                                    REPLY
  +---------------------+                   +---------------------+
 0|        16 |       8 |                  0|        11 |       8 |
 1|                     |                  1|                     |
 2|           | octal 20|                  2|           | octal 20|
 3|                     |                  3|                     |
 4|                     |                  4|                     |
 5|                     |                  5|                     |
 6|                     |                  6|                     |
 7|   16 + 0            |                  7|   6 + 0             |
  +---------------------+                   +---------------------+
 0| Intrinsic 235(octal)|                  0| A-Register          |
 1|                     |                  1| 0                   |
 2|                     |                  2| IERR                |
 3|                     |                   +---------------------+
 4| DCB                 |
 5|  :                  |
 6|  :                  |
 7| ITRUN               |
  +---------------------+
```

DNAME: Class 8, Stream 20

Intrinsic Number 158

```
          REQUEST                                  REPLY
  +---------------------+                  +---------------------+
 0|        20 |       8 |                 0|        11 |       8 |
 1|                     |                 1|                     |
 2|           | octal 20|                 2|           | octal 20|
 3|                     |                 3|                     |
 4|                     |                 4|                     |
 5|                     |                 5|                     |
 6|                     |                 6|                     |
 7|  24 + 0             |                 7|  6 + 0              |
  +---------------------+                  +---------------------+
 0| Intrinsic 236(octal)|                 0| A-Register          |
 1|                     |                 1| 0                   |
 2|                     |                 2| IERR                |
 3|                     |                  +---------------------+
 4| Old Name            |
 5|  :                  |
 6|  :                  |
 7| New Name            |
 8|  :                  |
 9|  :                  |
10| Security Code       |
11| Cartridge Number    |
  +---------------------+
```

DCONT: Class 8, Stream 20

Intrinsic Number 159

```
              REQUEST                                 REPLY
   +----------------------+              +----------------------+
  0|        17 |        8 |             0|        11 |        8 |
  1|                      |             1|                      |
  2|           | octal 20 |             2|           | octal 20 |
  3|                      |             3|                      |
  4|                      |             4|                      |
  5|                      |             5|                      |
  6|                      |             6|                      |
  7|   18 + 0             |             7|   6 + 0              |
   +----------------------+              +----------------------+
  0| Intrinsic 237(octal) |             0| A-Register           |
  1|                      |             1| 0                    |
  2|                      |             2| IERR                 |
  3|                      |              +----------------------+
  4| DCB                  |
  5|  :                   |
  6|  :                   |
  7| ICNWD1               |
  8| ICNWD2               |
   +----------------------+
```

DLOCF: Class 8, Stream 20

Intrinsic Number 160

```
           REQUEST                                  REPLY
  +---------------------+                +---------------------+
 0|       15 |        8 |               0|       18 |        8 |
 1|                     |               1|                     |
 2|          | octal 20 |               2|          | octal 20 |
 3|                     |               3|                     |
 4|                     |               4|                     |
 5|                     |               5|                     |
 6|                     |               6|                     |
 7|   14 + 0            |               7|  20 + 0             |
  +---------------------+                +---------------------+
 0| Intrinsic 240(octal)|               0| A-Register          |
 1|                     |               1| 0                   |
 2|                     |               2| IERR                |
 3|                     |               3| IREC                |
 4| DCB                 |               4| IRB                 |
 5|  :                  |               5| IOFF                |
 6|  :                  |               6| JSEC                |
  +---------------------+               7| JLU                 |
                                        8| JTY                 |
                                        9| JREC                |
                                         +---------------------+
```

DAPOS: Class 8, Stream 20

Intrinsic Number 161

```
           REQUEST                                  REPLY
  +----------------------+              +----------------------+
 0|        18 |        8 |             0|        11 |        8 |
 1|                      |             1|                      |
 2|           | octal 20 |             2|           | octal 20 |
 3|                      |             3|                      |
 4|                      |             4|                      |
 5|                      |             5|                      |
 6|                      |             6|                      |
 7|   20 + 0             |             7|   6 + 0              |
  +----------------------+              +----------------------+
 0| Intrinsic 241(octal) |             0| A-Register           |
 1|                      |             1| 0                    |
 2|                      |             2| IERR                 |
 3|                      |              +----------------------+
 4| DCB                  |
 5|  :                   |
 6|  :                   |
 7| IREC                 |
 8| IRB                  |
 9| IOFF                 |
  +----------------------+
```

DSTAT: Class 8, Stream 20

Intrinsic Number 162

```
          REQUEST                              FIRST RECORD
                                                  REPLY
 +----------------------+              +----------------------+
0|       12 |         8 |             0|       10 |         8 |
1|                      |             1|                      |
2|          | octal 20  |             2|          | octal 20  |
3|                      |             3|                      |
4|                      |             4|                      |
5|                      |             5|                      |
6|                      |             6|                      |
7|   8 + 0              |             7|   0 + 248            |
 +----------------------+              +----------------------+
0| Intrinsic 242(octal)|             0| Status               |
1|                      |              |  Array               |
2| 124                  |              |   (124 words)        |
3|                      |              |                      |
 +----------------------+              |  (Sent as Data)      |
                                       |                      |
                                    123|                      |
                                       +----------------------+

                                            LAST RECORD
                                              (REPLY)

                                       +----------------------+
                                     0 |       10 |         8 |
                                     1 |                      |
                                     2 |          | octal 20  |
                                     3 |                      |
                                     4 |                      |
                                     5 |                      |
                                     6 |                      |
                                     7 |  4 + 0               |
                                       +----------------------+
                                     0 |  A-Register          |
                                     1 |  0                   |
                                       +----------------------+
```

DEXEC READ: Class 8, Stream 21

Intrinsic Number 163, ICODE = 1

```
         REQUEST
  +---------------------+
 0|        17 |       8 |
 1|                     |
 2|           | octal 21|
 3|                     |
 4|                     |
 5|                     |
 6|                     |
 7|   18 + 0            |
  +---------------------+
 0| Intrinsic 243(octal)|
 1|                     |
 2| Number of Data Words|
 3|                     |
 4| ICODE = 1           |
 5| ICNWD               |
 6| IBUFL               |
 7| IPRM1               |
 8| IPRM2               |
  +---------------------+
```

```
       FIRST RECORD
         (REPLY)
  +---------------------+
 0|         8 |       8 |
 1|                     |
 2|           | octal 21|
 3|                     |
 4|                     |
 5|                     |
 6|                     |
 7|   0 + DL            |
  +---------------------+
 0|  Data               |
  |   Words             |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  +---------------------+
```

```
            LAST
           RECORD
           (REPLY)
  +----------------------+
 0|       10 |         8 |
 1|                      |
 2|          | octal 21  |
 3|                      |
 4|                      |
 5|                      |
 6|                      |
 7|   4 + 0              |
  +----------------------+
 0| A-Register (EQT5)    |
 1| B-Register (XMSN)    |
  +----------------------+
```

DEXEC WRITE: Class 8, Stream 21

Intrinsic Number 163, ICODE = 2

```
            FIRST
           REQUEST                                   REPLY
  +---------------------+           +---------------------+
 0|       17 |        8 |          0|       10 |        8 |
 1|                     |          1|                     |
 2|          | octal 21 |          2|          | octal 21 |
 3|                     |          3|                     |
 4|                     |          4|                     |
 5|                     |          5|                     |
 6|                     |          6|                     |
 7|   18 + DL           |          7|   4 + 0             |
  +---------------------+           +---------------------+
 0| Intrinsic 243(octal)|          0| A-Register (EQT 5)  |
 1|                     |          1| B-Register (XMSN)   |
 2| Number of Data Words|           +---------------------+
 3|                     |
 4| ICODE = 2           |
 5| ICNWD               |
 6| IBUFL               |
 7| IPRM1               |
 8| IPRM2               |
  +---------------------+
 9|                     |
  | Data                |
  |  Words              |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  |                     |
  +---------------------+
```

```
             ADDITIONAL
               DATA
             (REQUEST)
   +----------------------+
  0|        8 |         8 |
  1|                      |
  2|          | octal 21  |
  3|                      |
  4|                      |
  5|                      |
  6|                      |
  7|   0 + DL             |
   +----------------------+
  0|                      |
   | Data                 |
   |  Words               |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   |                      |
   +----------------------+
```

DEXEC I/O CONTROL: Class 8, Stream 21

Intrinsic Number 163, ICODE = 3

```
          REQUEST                                     REPLY
 +---------------------+                    +---------------------+
0|        15 |       8 |                   0|        10 |       8 |
1|                     |                   1|                     |
2|           | octal 21|                   2|           | octal 21|
3|                     |                   3|                     |
4|                     |                   4|                     |
5|                     |                   5|                     |
6|                     |                   6|                     |
7|  14 + 0             |                   7|   4 + 0             |
 +---------------------+                    +---------------------+
0| Intrinsic 243(octal)|                   0| A-Register          |
1|                     |                   1| B-Register          |
2|                     |                    +---------------------+
3|                     |
4| ICODE = 3           |
5| ICNWD               |
6| IPRAM               |
 +---------------------+
```

DEXEC SCHEDULE: Class 8, Stream 21

Intrinsic Number 163, ICODE = 10

```
          REQUEST                              REPLY
  +---------------------+              +---------------------+
 0|        21 |       8 |             0|        10 |       8 |
 1|                     |             1|                     |
 2|           | octal 21|             2|           | octal 21|
 3|                     |             3|                     |
 4|                     |             4|                     |
 5|                     |             5|                     |
 6|                     |             6|                     |
 7|  26 + 0             |             7|   4 + 0             |
  +---------------------+              +---------------------+
 0| Intrinsic 243(octal)|             0| A-Register          |
 1|                     |             1| B-Register          |
 2|                     |              +---------------------+
 3|                     |
 4| ICODE = 12 (octal)  |
 5| Program Name        |
 6|   :                 |
 7|   :                 |
 8| Scheduling Param 1  |
 9| Parameter 2         |
10| Parameter 3         |
11| Parameter 4         |
12| Parameter 5         |
  +---------------------+
```

DEXEC TIME: Class 8, Stream 21

Intrinsic Number 163, ICODE = 11

```
          REQUEST                               REPLY
  +--------------------+               +--------------------+
 0|        13 |      8 |              0|        15 |      8 |
 1|                    |              1|                    |
 2|           | octal 21 |            2|           | octal 21 |
 3|                    |              3|                    |
 4|                    |              4|                    |
 5|                    |              5|                    |
 6|                    |              6|                    |
 7|    10 + 0          |              7| 14 + 0             |
  +--------------------+               +--------------------+
 0| Intrinsic 243(octal)|             0| A-Register         |
 1|                    |              1| B-Register         |
 2|                    |              2| 10s of Milliseconds |
 3|                    |              3| Seconds            |
 4| ICODE = 13 (octal) |              4| Minutes            |
  +--------------------+              5| Hours              |
                                      6| Day of Year        |
                                       +--------------------+
```

DEXEC EXECUTION TIME: Class 8, Stream 21

Intrinsic Number 163, ICODE = 12

```
              REQUEST                                           REPLY
   +----------------------+                        +----------------------+
  0|        22 |        8 |                       0|        10 |        8 |
  1|                      |                       1|                      |
  2|           | octal 21 |                       2|           | octal 21 |
  3|                      |                       3|                      |
  4|                      |                       4|                      |
  5|                      |                       5|                      |
  6|                      |                       6|                      |
  7|   28 + 0             |                       7|   4 + 0              |
   +----------------------+                        +----------------------+
  0| Intrinsic 243(octal) |                       0| A-Register           |
  1|                      |                       1| B-Register           |
  2|                      |                        +----------------------+
  3|                      |
  4| ICODE = 14 (octal)   |
  5| Program Name         |
  6|   :                  |
  7|   :                  |
  8| Resolution           |
  9| Multiple             |
 10| Offset(-)/Hours(+)   |
 11| Minutes              | \  Not used
 12| Seconds              |  >  if word 10
 13| 10s of Milliseconds  | /    is negative.
   +----------------------+
```

DEXEC I/O STATUS: Class 8, Stream 21

Intrinsic Number 163, ICODE = 13

```
           REQUEST                                    REPLY
  +----------------------+            +----------------------+
 0|        14 |        8 |           0|        13 |        8 |
 1|                      |           1|                      |
 2|           | octal 21 |           2|           | octal 21 |
 3|                      |           3|                      |
 4|                      |           4|                      |
 5|                      |           5|                      |
 6|                      |           6|                      |
 7|   12 + 0             |           7|   10 + 0             |
  +----------------------+            +----------------------+
 0| Intrinsic 243(octal) |           0| A-Register           |
 1|                      |           1| B-Register           |
 2|                      |           2| ISTA1 (EQT 5)        |
 3|                      |           3| ISTA2 (EQT 4)        |
 4| ICODE = 15 (octal)   |           4| ISTA3 (Subchannel)   |
 5| CONWD                |            +----------------------+
  +----------------------+
```

Appendix C
DS/1000-IV Internal Table Formats

Pseudo-DCB Format

Each program calling a Remote File Access subroutine (DOPEN, DREAD, etc.) requires a four-word pseudo-DCB array. Its format is shown below:

WORD	CONTENTS
1	Master program's ID segment address
2	Address of nine-word status entry for DCB
3	Reserved for future use
4	Node number in which file is open

Figure C-1. Format of Pseudo-DCB.

PTOP Control Block (PCB)

Each program calling Program-to-Program Communication routines (POPEN, PREAD, etc.) requires a four-word PCB. The format is shown in Figure C-2.

WORD	CONTENTS
1	Slave program's ID segment address
2	Slave program's class number
3	Reserved for future use
4	Node number in which slave program is running

Figure C-2. Four-Word PCB Format.

RES Formats

RES is a memory-resident system library module used by the DS/1000 software package to provide controlled-access common storage. Items stored in RES are network-global constants and the heads of various lists. The lists, NRV, and TST are maintained in SAM.

Process Number List (#PNLH) Format

#PNLH contains the address of the first Process Number List entry. A free TCB entry is taken and put in the PNL list whenever a session is established at a remote node. This list is maintained so that if active programs terminate without issuing a BYE or explicitly logging off, the resources allocated to them can be deleted by UPLIN. Entries are created by the HELLO and DLOGN/#MSSM subroutines when REMAT services an ATtach command, or when RMOTE services a HELLO command. They are deleted by the BYE or DLOGF subroutines, or when RMOTE services a BYE command (or forces a "BYE" when an EXIT command is entered with a HELLO outstanding), when REMAT services a DE or EX command, when UPLIN cleans up after a 3000 link goes down or when any program that allocated a PNL entry terminates. The format is shown in Figure C-3.

```
**********************************************************************
*                                                                    *
*  +---------+            +---------------------------------------+  *
*  |         |            | address of next process # entry, or 0 |  *
*  |  #PNLH  |----------->|   (each entry has the same format as  |  *
*  |         |            |    shown below)                       |  *
*  +---------+            +---------------------------------------+  *
*                         | |3K|               0                  |  *
*                         +---------------------------------------+  *
*                         | remote node number (-LU for MPE   )   |  *
*                         +---------------------------------------+  *
*                         |      0          | user's terminal LU  |  *
*                         |                 |    (8 bits)         |  *
*                         +---------------------------------------+  *
*                         | B | I.D. Segment address of session's |  *
*                         |   |        owner                      |  *
*                         +---------------------------------------+  *
*                         |remote sub-level| remote session ID    |  *
*                         +---------------------------------------+  *
*                                                                    *
*  Legend:                                                           *
*                                                                    *
*  B (bit 15), if set, means entry is to be deleted by UPLIN         *
*  3K: (bit 14) set to 1 if session is at a 3000, else 0 if at RTE   *
*                                                                    *
*  Note:  0 in list head or first word of entry signals end of list. *
*                                                                    *
**********************************************************************
```

Figure C-3. #PNLH List Format

Master TCB List (#MRTH) Format

#MRTH contains the address of the first master list entry. This threaded list contains one entry for each master request currently outstanding (not yet serviced). The format is shown in Figure C-4.

```
***********************************************************************
*                                                                     *
*                          15 14 13 12 11 10  9  8  7  6  5  4 3 2 1  0
*   +--------+            +------------------------------------------+  *
*   | #MRTH |----------->| Address of next master-list entry or     |  *
*   +--------+            | zero.  All entries have the same format  |  *
*                         +------------------------------------------+  *
*                         | U | D | F | XXX | T |Timeout Counter (0-7)|  *
*                         +------------------------------------------+  *
*                         |Local transaction sequence number         |  *
*                         +------------------------------------------+  *
*                         | C | Master Class Number                  |  *
*                         +------------------------------------------+  *
*                         | B | ID segment address of master program |  *
*                         +------------------------------------------+  *
*                         | MA Seq #| Channel ID (index into MA table)|  *
*                         +------------------------------------------+  *
*                                                                     *
*  Legend:                                                            *
*                                                                     *
*  U (bit 15), if set, is a temporary bit used by UPLIN.              *
*  D (bit 14), if set, means the request is an HP 3000 request,       *
*                 either master or slave.                             *
*  F (bit 13), if set, indicates MA has acknowledged receipt of the   *
*                 request at the remote node.                         *
*  C (bit 15), if set, means that a timeout of approximately 20       *
*                 minutes is to be used.                              *
*  B (bit 15), if set means entry is a duplicate, which UPLIN will    *
*                 release at its next execution.                      *
*  XXX        , Reserved for future use.                              *
*  T (bit 8)  , if set, means that the TCB has timed-out              *
*                                                                     *
*  Note:  0 in list head or first word of entry signals end of list.  *
*                                                                     *
***********************************************************************
```

Figure C-4. #MRTH List Format

#STxx List Format (SLAVE TCBs)

#ST00 thru #ST11, respectively, contain the addresses of the first slave-stream entries, by stream-type. Entries in stream-type 0 are in the list headed by #ST00, stream-type 1 entries are headed by #ST01, etc. The lists are all threaded. The format is the same for all slave streams and is shown in Figure C-5.

```
***********************************************************************
*                                                                     *
*  Storage in RES            Storage in System Available Memory (SAM) *
*  --------------            ---------------------------------------- *
*                                                                     *
*  +------+  +----------+    +---------------------------------------+ *
*  |#STxx|-->| Address  |-->| Address of next Stream Type xx entry,| *
*  +------+  | of first|    | or zero.                              | *
*            | Stream   |    +---------------------------------------+ *
*            | Type xx  |    | U | D |     Timeout  Counter          | *
*            | entry    |    +---------------------------------------+ *
*            +----------+    | Local transaction sequence number     | *
*            |Monitor's|    +---------------------------------------+ *
*            |  class   |    | Original transaction sequence number | *
*            |  number  |    +---------------------------------------+ *
*            +----------+    |Origin Node number(1000) or -LU (3000)| *
*            |A|Monitor|    +---------------------------------------+ *
*            | |Name    |    | Reserved                              | *
*            | |Ascii   |    +---------------------------------------+ *
*            +----------+                                             *
*                                                                     *
*                                                                     *
*  Legend:                                                            *
*                                                                     *
*  A (bit 15), if set, means monitor may be aborted by UPLIN if       *
*              slave TCB is timed out (i.e., monitor contains no      *
*              tables which must be saved).                           *
*                                                                     *
*  U (bit 15), if set, is a temporary bit used by UPLIN.              *
*                                                                     *
*  D (bit 14), if set, means the request is an HP 3000 request,       *
*              either master or slave.                                *
*                                                                     *
*  Note: 0 in list head or first word of entry signals end of list.   *
*                                                                     *
***********************************************************************
```

Figure C-5. #STxx List Format

Network Routing Vector Table (NRV)

#NODE contains the local node number. #NCNT contains the two's complement of the number of NRV word-triplett entries.

```
***********************************************************************
*                        15 14 13 12 11 10 9 8 7 6 5 4  3  2  1  0    *
*  +-------+             +------------------------------------------+ *
*  | #NRV  |------------>|  Node address  ( range: 0 to 32767)      | *
*  +-------+             +------------------------------------------+ *
*                        |      Timeout       |Reserved| Upgrade    | *
*                        |      8 bits        | 4 bits |  Level     | *
*                        +------------------------------------------+ *
*                        |C|    Reserved      |N|      LU           | *
*                        +------------------------------------------+ *
*                        |                                          | *
*                        | (entries repeat, three words per entry,  | *
*                        |  as above.  Length given by #NCNT )      | *
*                        |                                          | *
*                        +------------------------------------------+ *
*                                                                     *
*   Legend:                                                           *
*                                                                     *
*   Upgrade level = 0 if node is DS/1000 (91740),                     *
*                   1 if DS/1000-IV (91750A)                          *
*               N = 1 if node in word 1 is neighbor on link LU in     *
*                     word 3                                          *
*              LU = the link LU to which a message for this node      *
*                     will be directed                                *
*           If LU = 0, then no link exists to this node (except if    *
*                     it is local node number)                        *
*               C = change bit, used by re-routing                    *
*                                                                     *
***********************************************************************
```

Figure C-6. Network Route Vector Table

Link Vector (LV)

The Link Vector is used to store such information as the Link LU, the status of the link (Up/Down), the COST value associated with this link, the system time used by the Up/Down Count, the Up/Down Counter and the Neighbor Node Number. #LV is stored in RES to point to the begining of Link Vector. #LCNT contains the number of LV entries.

```
*****************************************************************
*                                                               *
*  +-----+           +------------------------------------------+  *
*  | #LV |---------->|S|                  |     Link   LU       |  *
*  +-----+           +------------------------------------------+  *
*                    |        COST value for this link          |  *
*                    +------------------------------------------|  *
*                    |        System  Time  in   -hour          |  *
*                    | - - - - - - - - - - - - - - - - - - - -  |  *
*                    |            and    -seconds               |  *
*                    +------------------------------------------+  *
*                    |            Up/Down  Counter              |  *
*                    +------------------------------------------+  *
*                    |         Neighbor  Node  Number           |  *
*                    +------------------------------------------+  *
*                    |                                          |  *
*                    -  (entries repeat, six words per entry,   -  *
*                    -  as above.  Length given by #LCNT )      -  *
*                    |                                          |  *
*                    +------------------------------------------+  *
*                                                               *
*****************************************************************
```

Figure C-7. Rerouting Link Vector.

The status bit S is set to 0 if the link is downed and 1 if up.

Cost Matrix (CM)

The cost Matrix is a 2-dimensional matrix which gives the "cost" value to each destination over all possible outgoing links. #CM points to the begining of Cost Matrix.

```
******************************************************************
*                                                                *
*                  <-m ( #of rerouting links in this node)--->   *
*  +-----+         +-----------------------------------------+   *
*  | #CM |-------->|CM(1,1)|CM(1,2)|.................|CM(1,m)|   *
*  +-----+    |    +-----------------------------------------+   *
*             |    |CM(2,1)|CM(2,2)|.................|CM(2,m)|   *
*             |    +-----------------------------------------+   *
*             |    |   .                                     |   *
*             n    +-----------------------------------------+   *
*             |    |   .                                     |   *
*          (No. of +-----------------------------------------+   *
*           Nodes) |   .                                     |   *
*             |    +-----------------------------------------+   *
*             |    |   .                                     |   *
*             |    +-----------------------------------------+   *
*             |    |CM(n,1)|CM(n,2)|.................|CM(n,m)|   *
*             -    +-----------------------------------------+   *
*                                                                *
******************************************************************
```

Figure C-8. Cost Matrix.

Each entry is two words long and the entries are stored by row (i.e. all entries for a node are stored first followed by entries for the next node). The first word of any entry is the user-specified "cost" value to the destination over a particular link, and the second word is the hop count to the destination from this node.

Message Accounting Table Format

Each channel in the network included in Message Accounting will have an entry in the MA Table.

```
+------+
|#MCTR|
+------+               +-------------------------------------------+
|#MTBL| --->     +0    |  NODE - Node Number                       |
+------+               +-------------------------------------------+
                 +1    |  MASW - VSO(4) / R(2) / TMAX(8) / STATE(2)|
                       +-------------------------------------------+
                 +2    |  VA - Acknowledgement Variable            |
                       +-------------------------------------------+
                 +3    |  VT1/VT2 - Acknowledgement & Idle timers  |
                       +-------------------------------------------+
                 +4    |  VR - Receive State Variable              |
                       +-------------------------------------------+
                 +5    |  VF - Receive Flags                       |
                       +-------------------------------------------+
                 +6    |  VC - Cancel Flags (below VR)             |
                       +-------------------------------------------+
                 +7    |  VCC(4) / VCD(12)                         |
                       +-------------------------------------------+
                 +8    |  LERC - Last network error reported       |
                       +-------------------------------------------+
                 +9    |  LERN - Last reporting node number        |
                       +-------------------------------------------+

                       ~             (Next Record)                 ~
```

Figure C-9. Message Accounting Table.

where:

```
R     - Reserved
MASW  - MA State Word
VSO   - Send State offset from the last sequence number
        acknowledged, (VA).  This is the number of
        messages outstanding.  VSO + VA = VS.
TMAX  - Max retry "ticks" from the "MA timeout".
STATE - Current state of the channel,
        (down=0, non-ma=1, up=2, pending=3).
VCC   - Consecutive Cancellation counter
VCD   - Number of times the channel has gone down
```

Remote Session ID Pool (POOL)

#POOL contains the address of a block of words in System Available Memory (SAM) used for a table of n session identifiers, as follows. Entries are used by RSM. If #POOL is 0, then DINIT was unable to set up the pool.

```
                              +---------------------------------+
 #POOL -----------------> | negative # of #POOL entries    |
                          /+---------------------------------+
                         / |U|C|A|reserved|N|pool session ID|
                        /  +---------------------------------+
                       |   | owner's source node #          |
                       |   +---------------------------------+
                       |   |EXECW seq. #    |owners's source|
      n entries      <     |                |session ID     |
                       |   +---------------------------------+
                       |   |  P                  R           |
                       |   |  G                  N           |
                       |   |  M             |  blank         |
                        \  +---------------------------------+
                         \ |  timer word                     |
                          \+---------------------------------+
```

Figure C-10. Remote Session ID Pool.

U (bit 15)	= 1 if session ID in use
C (bit 14)	= 1 if a program was cloned for this session
A (bit 13)	= 1 if session being logged off
N (bit 8)	= 1 if program remotely scheduled by EXECW from a remote session
PRGNM	= Name of program scheduled under this session
Timer Word	= Number of 5-second intervals since session was last accessed.

Network Account Table

Every master DS program that is loaded by searching the $DSLSM or $DSSM library has appended the module #NAT, which contains a Network Account Table local to the user. The NAT identifies remote sessions (if any) in use by this program. Entries are created by DLGON and #MSSM. When #MAST calls #MSSM, the destination node in the message is used to find the destination session ID (if any) and the message header #SID word is then updated. All 16 entries have the same format.

```
+------+           +---------------------------+
| #NAT |---------->| Destination node #        |
+------+           +---------------------------+
                   |F|P|S|sub-  |  destination  |
                   | | | |level|   session ID  |
                   +---------------------------+
                   |address of user-name (in   |
                   | REMAT's #RMSM)            |
                   +---------------------------+
                   | reserved   | EXECW seq. # |
                   +---------------------------+
                   | PNL pointer                |
                   +----------------------------+
```

Figure C-11. Network Account Table.

F (bit 15) = 1 if another program owns the session.
P (bit 14) = 1 if the owner is in the same process group.
S (bit 13) = 1 if the program is slaving back to the session.

If you need more than 16 entries, a module similar to the following may be assembled and loaded with the program before searching any libraries.

```
ASMB,Q=L,O
      NAM  NAT,7 91750-1X276 REV.2301 821022 ALL
      SPC 1
*******************************************************************
*  * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1982. ALL RIGHTS       *
*  * RESERVED. NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,        *
*  * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT*
*  * THE PRIOR WRITTEN CONSENT OF THE HEWLETT-PACKARD COMPANY.    *
*******************************************************************
      SPC 1
*
* NAME:    NAT
* SOURCE: 91750-18276
* RELOC:  91750-1X276
      SPC 1
*
*  Modifications:
*  Date  Prgmr      Description
*
      SPC 2
*     Module description:
*
*   NAT is the storage for the NAT, used by  MSSM and  RMSM.
*  The NAT stores information about all the remote sessions
*  that this program can currently access.
*
*  16 entries, five words each:
*
*                 Node number
*                 Session ID
*                 Addr of ASCII user name (for  RMSM)
*                 EXECW Sequence Number
*                 Address of PNL entry for this session
*
*  To increase the number of remote sessions a single program is
*  allowed to have, increase SIZE to the number of sessions wanted.
*
*                           WARNING
*  Do not attempt to increase REMAT's capabilities this way; REMAT
*  has an internal table (the NATX) which must also be extended
*  in order to expand the number of nodes available.
*
      ENT NAT , NAT
SIZE  EQU 16
 NAT  DEF *+1
      REP SIZE*5
      OCT 0
NAT   ABS -SIZE
      END
```

3000 LU Table

A two word table with the value of each HP 3000 LU and its associated message buffer size. #LU3K points to the table in SAM. #3KLU specifies the number of entries in the table.

```
                        15                     7                    0
 +-------+             +----------------------------------------------+
 | #LU3K |------------>|  XXXXXX    | X | C |         LU              |
 +-------+             +----------------------------------------------+
                       |     Buffer size                              |
                       +----------------------------------------------+
                       |                                              |
                       -                                              -
                       | entries repeat, two words per entry, for     |
                       | each 3000 LU decared in DINIT                |
                       +----------------------------------------------+
```

Legend:

X (bit 9) , contains the X.25 flag. The flag will be 1 if the LU is an X.25 pool LU, 0 if otherwise.

C (bit 8) , contains the Continuation Record (CR) Flag. The CR Flag will be 1 if the HP 3000 accepts a new continuation record for $STDIN/$STDLIST, 0 if it does not.

XXXXXX , Reserved for future use.

Transaction Status Table

The TST is used to keep track of all master requests from an HP 3000. It provides storage for information from the DS/3000 fixed format header (which is not necessary to process a request, but is required to process a reply) and DS/1000-IV information generated by RQCNV.

```
                  15                    8                    0
+------+        +--------------------------------------------+  ----------
| #TST |--->|            DS/1000-IV Stream Word              |0      ^
|      |        +--------------------------------------------+       |
+------+        |            Local Sequence Number           |       |
                +--------------------------------------------+       |
                | Holding Class # (for continuation buffers) |    built by
                +--------------------------------------------+     RQCNV
                |               Monitor Class #              |       |
                +--------------------------------------------+       |
                +--------------------------------------------+       |
                |               POPEN Mask Word              |5      v
                +--------------------------------------------+------------
                |      Word Count      |    Message Class    |6      ^
                +--------------------------------------------+       |
                |                  Reserved                  |       |
                +--------------------------------------------+       |
                |             DS/3000 Stream Word            |       |
                +--------------------------------------------+     from
                |                  Reserved                  |    DS/3000
                +--------------------------------------------+    request
                |    From Process #    |    To Process #     |    header
                +--------------------------------------------+       |
                |               RTE Sequence #               |       |
                +--------------------------------------------+       |
                |                  Reserved                  |       |
                +--------------------------------------------+       |
                |                 HP 3000 LU                 |13     v
                +--------------------------------------------+  ----------
```

DS Driver EQT's

DVA66 EQT's

Two consecutive EQTs are required for each link supported by DVA66. They must be configured with no DCPC required and no automatic buffering. DVA66 must be specified as the driver. The first EQT must have a 12 word extension; no extension is necessary for the second EQT.

The EQT extension is the main area for long term storage of information required by DVA66 for each link. The driver uses the fact that the "driver processes timeout" bit is initially cleared by the generator to detect the first entry to the driver and initialize the extension.

Other words in the EQT's are used by the driver. Word 12 of each EQT points to the first word of the other EQT. Word 13 of each EQT points to the extension of the first EQT. Word 11 of each EQT indicates which process is servicing the request on that EQT, or zero if no request is pending.

Every time the driver is entered, it sets a pointer for each word of the EQT extension. These pointers are the sole means by which the driver references the extension, and each pointer has a unique mnemonic label. This allows the format of the extension to evolve by changing the pointers and the pointer set-up routine, without modifying the rest of the driver. The format of the EQT extension follows, with the label of the pointer to each word listed to the left of that word.

```
         15  14  13  12  11  10   9   8   7   6   5   4   3   2   1   0
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
RPTRY   |          retry counter / pointer to next word to read        |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
RLEN1   |     length of first read buffer / number of words to skip    |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
RLEN2   |     length of second read buffer / number of words to skip   |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
RDSIZ   |                 length of frame ready on card                |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
WPTR    |                 pointer to next word to write                |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
WLEN    |                    length of write buffers                   |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
WFSIZ   |                  maximum frame size of PSI                   |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
WBUFS   |               number of allocated output buffers             |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
RCONT   |               read process continuation address              |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
WCONT   |               write process continuation address             |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
FBITS   |FW |RC |SE |IN |ND |SM |LC |LT |MT |ST |WL |RL |WP |RP |WA |RA|
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
FMISC   |     TW        |     TR    |FM |CI |             LU            |
        +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+--+
```

Figure C-12. DVA66 EQT Extension.

The meaning of each field of the EQT extension is further described below. For each field, the range is given over which the value of the field satisfies the description. "ALL" indicates that the field is always meaningful. "R" or "W" followed by one or more digits indicate that the field is meaningful when the read or write processes respectively are executing in one of the parts specified by the digits. For fields which serve more than one purpose, a separate range and description are provided for each use.

In the descriptions, there is a distinction between the pending incomming message, which is data on the card, and the pending read message command, which is an EXEC request. Also, a word is deemed to be transferred to or from the card when the DCPC transfer which moves the word has been started, even though the word may be at the end of the transfer. All lengths are in words.

NAME	RANGE	DESCRIPTION
RPTRY	R3	the number of attempts to schedule QUEUE to generate a read request
	R4	points to the first word of the read request buffers not yet transferred from the card
RLEN1	R3	the length of the first buffer of the pending incoming message
	R4	if positive, the original number of words in the frame being read, if negative or zero, the number of words of the pending incoming message that must still be skipped before any words are transferred
RLEN2	R3	the length of the second buffer of the pending incoming message
	R4	the number of words of the pending read request that must still be transferred from the card
RDSIZ	ALL	the length of the pending incoming frame on the card, 0 if no frame is ready
WPTR	W2	points to the first word of the write request buffers not yet transferred to the card, sign bit set if the length words have not yet been transferred to the card
WLEN	W2	the number of words of the pending send message request that must still be transferred to the card

WFSIZ	ALL	the maximum frame size of the PSI
WBUFS	ALL	if non-negative, the number of output buffers reserved on the card; if negative, no buffers have been reserved and the card has been asked to inform the driver when some become available and WFUBS+60 half seconds have elapsed since the card was asked to allocate buffers.
RCONT	ALL	continuation address of the read process
WCONT	ALL	continuation address of the write process
FBITS		flags, bit is set if condition is true
FW	ALL	the VCP Wait Flag is set
RC	ALL	reconnect after power failure
SE	ALL	a severe error has occurred (unexpected timeout or power-fail), the write process is recovering
IN	ALL	link has been INitialized
ND	ALL	the Non-DS flag is set
SM	ALL	the current frame is the Start of a Message
LC	ALL	line is Logically Connected
LT	ALL	a Long (15 sec) Timeout has been requested
MT	ALL	a Medium (.5 sec) Timeout has been requested
ST	ALL	a Short (.1 sec) Timeout has been requested
WL	ALL	the backplane is Locked to the Write process
RL	ALL	the backplane is Locked to the Read process, if both WL and RL are set the backplane is executing a command from the backplane section
WP	ALL	a request serviced by the Write Process is pending and not aborted
RP	ALL	a request serviced by the Read Process is pending and not aborted
WA	ALL	request on Write process Aborted, set when abort request made, cleared by write process when noticed, both WA and WP may be set if one request arrives shortly after the previous one was aborted

RA	ALL	request on Read process Aborted, set when abort request made, cleared by read process when noticed, both RA and RP may be set if one request arrives shortly after the previous one was aborted

FMISC miscellaneous information

TW	ALL	the send message request type of the card, that is, the type that the card will assign to the next message sent by the driver.
TR	ALL	the type of the current incoming message, if any, on the card, otherwise, the type of the last incoming message from the card
FM	ALL	a VCP Message has been read since the last time a connect indication was sent
CI	ALL	a Connect Indication must be sent
LU	ALL	the Logical Unit number of the EQT

Driver 65 Status Word

DVA65 maintains status in the EQT and in the EQT extension. The formats of these tables are shown below:

BIT	MEANING IF SET	
0	always zero	
1	any error (bits 4-7 indicate type)	
2	write request	
3	non-DS request	
4	bits 4 thru 7 indicate type of error:	
	Octal code	Error
	0	No error
	1	Line failure, a parity or protocol error was detected and could not be resolved.
	2	Timeout, remote did not respond to a protocol character within the line time-out interval.
	3	Local busy, driver is currently processing a message going in the opposite direction, or both sides of the link attempted to send messages at the same time.
	4	Message aborted. A 'STOP' was received.
	5	Remote busy, remote side was unable to schedule QUEUE or to allocate sufficient SAM.

Figure C-13. EQT Word 5 Status Field Error Codes.

	Fatal errors:
10	Not initialized. An initialize link command has not been received yet. If in response to an initialize command, indicates that system tables were not configured correctly.
11	Wrong mode. Wrong type of traffic (ie, trying to do a download when driver is is in DS mode)
12	Illegal request. Command is not supported by DVA65 (DVA65 will give this in response to a FPL send/receive, break, set mode request with the wrong security code).
13	Bad card.
17	Unknown interrupt received.

Figure C-13 (Cont'd). EQT Word 5 Status Field Error Codes.

EQT WORD #	USE
1	DEFINED BY RTIOC (I/O list address)
2	DEFINED BY RTIOC (initiator address)
3	DEFINED BY RTIOC (continuator address)
4	DEFINED BY RTIOC (status/unit/subchannel)
5	DEFINED BY RTIOC (availability/type/status)
6	DEFINED BY RTIOC (control word)
7	ADDRESS OF DATA BUFFER
8	LENGTH OF DATA BUFFER
9	ADDRESS OF REQUEST BUFFER/LU for initialize rqst
10	LENGTH OF REQUEST BUFFER
11	COROUTINE ADDRESS
12	CURRENT STATUS WORD (SEE BREAKDOWN)
13	ADDRESS OF EQT EXTENSION
14	DEFINED...USED FOR SINGLE WORD TURN-AROUND TIMEOUT
15	DEFINED...EQT TIMEOUT WORKING COUNTER

Figure C-14. DVA65 EQT Word Usage Breakdown.

WORD	CONTENTS
1	COUNTER FOR DATA TRANSFER
2	LAST WORD RECEIVED OVER COMM LINE
3	VERTICAL PARITY WORD
4	DIAGONAL PARITY WORD/RP DATA LENGTH
5	COUNT OF TOTAL BLOCKS TRANSMITTED
6	COUNT OF TOTAL NUMBER OF RE-TRANSMISSIONS
7	LU number

Figure C-15. DVA65 EQT Extension Format.

BIT	USAGE
0-2	Retry counter
0-5	broken line counter
6	BROKEN LINE FLAG
7	non-DS mode
8	not used
9	REQUEST PENDING
10	enabled to receive link traffic
11	reserved(used by special FCL drvr)
12	LAST SUCCESSFUL OPERATION (1=WRITE)
13	FLAG FOR WRITE RETRY IN PROGRESS
14	MICROCODE READ/WRITE FLAG
15	power fail recovery in progress

Figure C-16. Breakdown of EQT Word 12.

NOTE

The counters of the total number of transmissions, and the total number of re-try transmissions maintained by DVA65 can be a useful statistic for determining noisy lines.

DVA66 Status Word

DVA66 maintains status in the EQT and in the EQT extension. The formats of these tables are shown below:

```
+--------+---------------------------------------------------------+
|  BIT   |          MEANING IF SET                                 |
+--------+---------------------------------------------------------+
|   0    |  Always zero.                                           |
|   1    |  Any error (bits 4-7 indicate type).                    |
|   2    |  Write request.                                         |
|   3    |  Non-DS request.                                        |
|   4    |  Bits 4 thru 7 indicate type of error.                  |
|        |                                                         |
|        |  Octal code    Error                                    |
|        |                                                         |
|        |   0            No error.                                |
|        |                                                         |
|        |   1            Link disconnected (line failure)         |
|        |                                                         |
|        |   2            Timeout.  Frame did arrive within        |
|        |                the line time-out interval.              |
|        |                                                         |
|        |   3            Local busy.  Driver is currently proc-   |
|        |                essing a request on the same channel.    |
|        |                                                         |
|        |   4            Message aborted.  Possible reasons:      |
|        |                  -incompatible incoming message type    |
|        |                  -message frame not in order            |
|        |                  -missing frame                         |
|        |                  -frame size too big.                   |
|        |                                                         |
|        |   5            Remote busy.  Remote side was unable to  |
|        |                schedule QUEUE or to allocate sufficient |
|        |                SAM.                                     |
|        |                                                         |
+--------+---------------------------------------------------------+
```

Figure C-17. DVA66 EQT Status Field Error Codes.

	Fatal errors:
10	Not initialized. An initialize link command has not been received yet. If in response to an initialize command, indicates that system tables were not configured correctly.
11	Wrong mode. Wrong type of traffic (ie, trying to do a download when driver is is in DS mode)
12	Illegal request. Command is not supported by DVA66 (DVA66 will give this in response to a FPL send/receive, break, set mode request with the wrong security code).
13	Bad card.
17	Unknown interrupt received.

Figure C-17 (Cont'd). DVA66 EQT Status Field Error Codes.

Appendix D
Communication Binary Loader Listing

There are two communication Bootstrap Loader roms used with DS/1000-IV, one for the HDLC card (CBL66) and one for the 12771 card (CBL65). These listings are provided below.

You can check the CBL Initial Binary Loader ROM by comparing the contents of memory with this listing. Halt the computer, select the CBL ROM and communication link select code as described in Chapter 3, Volume One of the Network Manager's Manual. Examine locations 77700 through 77777 and compare to this listing.

The following listing is of CBL66 used with the HDLC links.

```
 PAGE 0001 #01


0001                ASMB,A,L,C
**  NO ERRORS PASS#1 **RTE ASMB 92067-16011**
1
 PAGE 0002 #01
    CBL66 91750-18046 REV 2340 830405


                ASMB,A,L,C
*
*     NAME: CBL66
*   SOURCE: 91750-18046
*      ROM: 91750-80018
*
*****************************************************************
*   * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1980.  ALL RIGHTS     *
*   * RESERVED.  NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,      *
*   * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT*
*   * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.       *
*****************************************************************
*
*  THIS IS THE CODE USED IN THE CBL ROM FOR INITIATING DOWNLOADS FROM A
*  REMOTE COMPUTER.  ON ENTRY THE SWITCH REGISTER CONTAINS FIVE OCTAL
```

```
*  DIGITS WHICH ARE LATER CONCATENATED WITH AN ASCII 'P' TO PRODUCE
*  THE FILE NAME OF THE CORE IMAGE TO BE DOWNLOADED; E.G., "P77777".
*  IF BIT 15 IS "1", CBL PRESUMES IT IS A FORCED COLD LOAD REQUEST,
*  AND ONLY USES BITS 5-0 FOR PROGRAM NO., SINCE BITS 15-6 CONTAIN THE
*  ROM-SELECT AND I/O-SELECT CODES.
*  I/O INSTRUCTIONS ARE CONFIGURED DURING THE IBL PROCESS.
*
 77700               ORG 77700B
*
*     PSI DS/1000 CBL ROM
*
*
 00010         CARD  EQU 10B
 00001         SREG  EQU 1B
*
 77700 106700  BOOT  CLC 0
 77701 017757        JSB WAIT
 77702 002404        CLA,INA
 77703 101741        CAX
 77704 063772        LDA !LOAD
 77705 017752        JSB COMND
 77706 102501        LIA SREG
 77707 002020        SSA
 77710 013774        AND B77
 77711 103610        OTA CARD,C
 77712 017757        JSB WAIT
 77713 017737  RECRD JSB GET
 77714 101050        LSR 8
 77715 105751        CBY
 77716 017737        JSB GET
 77717 106601        OTB SREG
 77720 077776        STB RPNT

PAGE 0003 #01
   CBL66 91750-18046 REV 2013 790829

*
 77721 017737  WORD  JSB GET
 77722 101754        CYA
 77723 002103        CLE,SZA,RSS
 77724 027766        JMP DONE
 77725 063777        LDA MXWRD
 77726 043776        ADA RPNT
 77727 002040        SEZ
 77730 027770        JMP HALT
 77731 177776        STB RPNT,I
 77732 037776        ISZ RPNT
 77733 105771        DSY
 77734 027721        JMP WORD
 77735 017737        JSB GET
 77736 027713        JMP RECRD
```

```
 77737 000000  GET   NOP
 77740 105761        DSX
 77741 027747        JMP AVAIL
 77742 063773        LDA !GTFR
 77743 017752        JSB COMND
 77744 102510        LIA CARD
 77745 101741        CAX
 77746 017752        JSB COMND
 77747 017757  AVAIL JSB WAIT
 77750 107510        LIB CARD,C
 77751 127737        JMP GET,I
 77752 000000  COMND NOP
 77753 102610        OTA CARD
 77754 103710        STC CARD,C
 77755 017757        JSB WAIT
 77756 127752        JMP COMND,I
 77757 000000  WAIT  NOP
 77760 067775        LDB M180
 77761 102210  TWDLE SFC CARD
 77762 127757        JMP WAIT,I
 77763 002007        INA,SZA,RSS
 77764 006024        SSB,INB
 77765 027761        JMP TWDLE
 77766 006003  DONE  SZB,RSS
 77767 024002        JMP 2
 77770 106055  HALT  OCT 106055
 77771 027700        JMP BOOT

PAGE 0004 #01
   CBL66 91750-18046 REV 2013 790829


 77772 161001  !LOAD OCT 161001
 77773 061400  !GTFR OCT 61400
 77774 000077  B77   OCT 77
 77775 177514  M180  DEC -180
 77776 000000  RPNT  NOP
 77777 100100  MXWRD ABS -BOOT
                     END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

The following listing is for CBL65 the loader rom that operates on the DVA65/12771/12773 link.

```
 PAGE 0002 #01
    CBL 24999-18174 REV 1826 780719 *(C) HEWLETT-PACKARD CO. 1978*
                ASMB,A,L
*
*
*
*******************************************************************
*   * (C) COPYRIGHT HEWLETT-PACKARD COMPANY 1978.  ALL RIGHTS     *
*   * RESERVED.  NO PART OF THIS PROGRAM MAY BE PHOTOCOPIED,      *
*   * REPRODUCED OR TRANSLATED TO ANOTHER PROGRAM LANGUAGE WITHOUT*
*   * THE PRIOR WRITTEN CONSENT OF HEWLETT-PACKARD COMPANY.       *
*******************************************************************
*
* THIS IS THE CODE USED IN THE CBL ROM FOR INITIATING DOWNLOADS FROM A
* REMOTE COMPUTER.  ON ENTRY THE SWITCH REGISTER CONTAINS FIVE OCTAL
* DIGITS WHICH ARE LATER CONCATENATED WITH AN ASCII 'P' TO PRODUCE
* THE FILE NAME OF THE CORE IMAGE TO BE DOWNLOADED; E.G., "P77777".
* IF BIT 15 IS "1", CBL PRESUMES IT IS A FORCED COLD LOAD REQUEST,
* AND ONLY USES BITS 5-0 FOR PROGRAM NO., SINCE BITS 15-6 CONTAIN THE
* ROM-SELECT AND I/O-SELECT CODES.  MEMORY IS CLEARED (EXCEPT FOR THE
* UPPERMOST 64 WORDS) BEFORE REQUESTING A DOWNLOAD.
* I/O INSTRUCTIONS ARE CONFIGURED DURING THE IBL PROCESS.
*
  77700              ORG 77700B
  77700 007400  CBL  CCB
  77701 063761       LDA XE$       XE MACRO FOR CBL MICROCODE
  77702 100060       OCT 100060    CHECK MACHINE TYPE
  77703 073720       STA MCALL     CAN GET HERE ONLY IF XE
  77704 102501       LIA 1         GET PGM #
  77705 001275       RAL,CLE,SLA,ERA  IS THIS A FORCED COLD LOAD?
  77706 013760       AND B77        YES, ISOLATE BITS# 5-0 OF PGM. NO.
  77707 073756       STA SWREG     SAVE THE PROGRAM NUMBER.
  77710 002040       SEZ           IF THIS IS A FORCED COLD LOAD,
  77711 027732       JMP FCL        THEN CLEAR CORE BEFORE PROCEEDING.
*
  77712 107710  INITZ CLC 10B,C    INITIALIZE CARD: SET RECEIVE MODE
  77713 107510       LIB 10B,C                      CLEAR STATUS REG.
  77714 106510       LIB 10B                        CLEAR RECEIVE REG.
  77715 067776       LDB SCODE  PASS SELECT CODE IN B REG TO MICROCODE

 PAGE 0003 #01
    CBL 24999-18174 REV 1826 780719 *(C) HEWLETT-PACKARD CO. 1978*

  77716 063756       LDA SWREG     GET THE USER'S PROGRAM NUMBER,
  77717 102601       OTA 1       AND ENSURE THAT THE SWITCH REG. HAS IT.
*
*  EXECUTE CBL MICROCODE: <B>=SELECT CODE, <SW>=PROGRAM NO.
*
```

```
  77720 105522  MCALL OCT 105522     MODIFIED TO 105302 IF AN XE
  77721 102601        OTA 1          FAILED IF WE GOT HERE, SET
  77722 063757        LDA NCNTR       ERROR CODE IN SW REG AND
  77723 006400        CLB              DISPLKAY FOR 6 SECONDS (IF XE)
  77724 006006        INB,SZB           BEFORE HALTING
  77725 027724        JMP *-1
  77726 002006        INA,SZA        6 SECONDS UP?
  77727 027724        JMP *-3         NO, LOOP
  77730 106055        OCT 106055      YES: HALT, OR RE-START RPL PROCESS.
*
*  IF OPERATOR PUSHES "RUN", DO A RETRY
*
  77731 027712        JMP INITZ      RETRY THE DOWNLOAD REQUEST.
*
  77732 067777  FCL   LDB HIAD       GET -(ADDR OF CBL)
  77733 006004        INB
  77734 007000        CMB            B=ADDR(CBL)-2
  77735 105741        CBX
  77736 006400        CLB             FIRST ZERO ALL
  77737 105740        SBX 1            OF MEMORY
  77740 000001
  77741 105761        DSX
  77742 027737        JMP *-3
  77743 007400        CCB            IF THE CPU IS AN XE, THEN
  77744 100060        OCT 100060      THIS MAY BE AN RPL RE-LOAD,
  77745 006401        CLB,RSS          SO SKIP TO WAIT FOR <UPLIN>;
  77746 027712        JMP INITZ         ELSE, START THE DOWNLOAD NOW.
  77747 063757        LDA NCNTR      DELAY FOR SIX SECONDS
  77750 102601        OTA 1           AND SHOW SOME LIFE,
  77751 006006        INB,SZB           WHILE WAITING FOR
  77752 027751        JMP *-1             <UPLIN> TO RE-ENABLE
  77753 002006        INA,SZA              THE    "BROKEN"
  77754 027750        JMP *-4                COMMUNICATIONS LINE.
  77755 027712        JMP INITZ      START THE DOWNLOAD PROCESS.
*
  77756 000000  SWREG NOP
  77757 177717  NCNTR DEC -49        6 SECOND XE DELAY
  77760 000077  B77   OCT 77
  77761 105302  XE$   OCT 105302
  77762 105304  SDNGX OCT 105304     XE SELF DIAG MACRO CALL
*
```

```
PAGE 0004 #01
    CBL 24999-18174 REV 1826 780719 *(C) HEWLETT-PACKARD CO. 1978*

*      SELF-DIAGNOSTIC FOR CBL-DS/1000 MICROCODE
*
*      TO EXECUTE SELF-DIAG:
*         1.) SET "S" REG TO LOAD CBL, PRESET AND IBL
*         2.) DO NOT PRESS RUN.
*         3.) SET "P" REG TO X7770B.
*         4.) PRESS RUN.
*         5.) "S" REGISTER SHOULD CONTAIN REVISION CODE
*             (ENCODED AS 4 BCD DIGITS) OF DS/1000 MICROCODE.
*
*         A HLT 55B INDICATES THE MICROCODE ROMS ARE NOT
*         INSTALLED PROPERLY, SINCE THE MICROCODE INSTRUCTION
*         INCREMENTS THE PC (SKIPS NEXT INSTRUCTION)
*
  77770               ORG CBL+70B
  77770 063762  SELFD LDA SDNGX
  77771 100060        OCT 100060    CHECK MACHINE TYPE
  77772 073773        STA SDNGM     CAN GET HERE ONLY IF XE
  77773 105524  SDNGM OCT 105524    INVOKE DS/1000 SELF-DIAG
  77774 102055        HLT 55B       MACRO SKIPS THIS INSTRUCTION
  77775 027770        JMP SELFD     REPEAT
*
  77776               ORG CBL+76B
  77776 000010  SCODE OCT 10     RESERVED, CPU PUTS SELECT CODE HERE
  77777 100100  HIAD  ABS -CBL   RESERVED, CPU PUTS -(ADDR OF CBL) HERE.
                      END
**  NO ERRORS *TOTAL **RTE ASMB 92067-16011**
```

Appendix E
Distributed Systems Compatibility

The programmer's interface to DS/1000 is backward-compatible with the SCE/5 programmer interface defined in Hewlett-Packard's earlier distributed system, 91700A (DS/1B), with three minor exceptions:

- In program-to-program communications, the tag fields have been expanded to 20 words (DS/1B used 10). There should be little problem in this upgrade.

- Error codes are slightly different. If your programs contain error-recovery code which depends on recognizing certain error codes, you may need to modify the program to check for different code values. This change should also be minor.

- Computers are addressed differently. In DS/1B, only neighbor addresses are allowed and they are positive LU numbers. DS/1000 interprets positive node address values to mean CPU node numbers. Thus, if you have a "star" network and give all satellites node numbers equal to the LU number the central uses to communicate with each, programs at the central need not be modified. Programs at the "satellites" must be modified to specify either "neighbor" addresses or the node address of the central. The node number of the central should be zero.

The interface subroutines defined for SCE/3 and SCE/4 satellites are not offered in DS/1000.

DS/1000 nodes have a link protocol which is incompatible with the DS/1B protocol. Therefore, DS/1000 nodes will not communicate with DS/1B nodes. This includes the SCE/1 used in 91700A, as well as SCE/2, SCE/3, SCE/4, SCE/5, SCE/6, SCE/7, SCE/8, and SCE/9 satellites. A mixture of DS/1B and DS/1000 software will not work in the same computer.

Appendix F
DS/1000-DS/1000-IV Compatibility

In DS/1000-IV a large effort has been expanded to maintain backward compatibility with DS/1000.

The DS/1000 interfaces (12771/12773 cards) are supported and DVA65 has been modified to achieve compatibility at the link protocol level. Communication between DS/1000 nodes and DS/1000-IV is fully compatible and supported.

Message converters are supplied in DS/1000-IV to translate message formats between DS/1000-IV and DS/1000.

There are minor areas where new features of DS/1000-IV are unavailable when data is transmitted to or through old nodes (DS/1000) to reach other new nodes (DS/1000-IV).

1. The dynamic rerouting feature does not function for traffic using the old links (DVA65) and old nodes to reach other new nodes.

2. HDLC protocol is not available to old nodes.

3. Loop Traffic can't be detected if the message is in DS/1000 (91740) format, i.e., if the message has been converted to DS/1000 format.

When accessing an old node, the following capabilities are not available.

1. No Remote session capability exists on the old node. For this reason access to group and private cartridges of a RTE-IVB Session node is not possible. Access to Session extended LU's is not permitted; the user is restricted to accessing only System LU's on a RTE-IVB Session and DS/1000 node.

   When accessing a new node from an old node a default session (253) is established. The old node does have access to whatever is available in this default session.

2. Any new feature of DS/1000-IV is not available when communicating with a DS/1000 node (Message Accounting, Rerouting, Remote Session, etc). One exception is IO MAPPING.

   The IO mapping feature (not available in DS/1000) can be used in DS/1000-IV to map an LU in a DS/1000-IV node to an LU on an DS/1000 node. For example, a DS/1000-IV node without a printer can use IO Mapping to map LU 6 on the new node without printer to LU 6 of the DS/1000 node with a printer. All the software to do this exists in the DS/1000-IV node.

Appendix G
Miscellaneous Special Considerations

Modem Restriction

Downloading systems and programs to an RTE-IVE system over a dialup modem using DS/1000-IV HDLC Modem Interface, product number 12794A, is currently not supported.

Remat Commands

LO (load program)

The REMAT LO command is used to load programs on memory-based RTE-L/XL/A and RTE-MIII systems. A different procedure is used to load programs on RTE-IVE systems. The format of the LO command is:

```
RTE-MIII systems:  #LO[,namr[,partition[,pages]]]

RTE-L/XL/A systems:  #LO,namr
```

To load programs on an RTE-IVE system use the following command:

```
#RW,APLDR,namr [,[partition] [,node#] ]
               [, partition           ]
```

PL (list programs and number of available ID segments)

The PL command does not work with RTE-IVE.

Remote I/O Mapping

All remote I/O mappings, both to and from a node, must be disabled before shutting down DS at the node.

All interactive devices for which an I/O mapping exists must have a non-zero timeout value. This applies to mappings both to and from a node. Also, it is recommended that prompt mode be used (bit 14 set in the IOMAP destination LU parameter) when mapping interactive devices.

Dynamic Message Rerouting

If an LU is specified as a rerouting LU in response to DINIT, then it should NEVER be specified explicitly as an LU in response to the DINIT query:

```
/DINIT:  CPU#,LU,TIMEOUT,UPGRADE LEVEL,"N","MA",MA TIMEOUT
```

Conversely, if an LU is explicitly specified in response to the DINIT question it should never be specified in DINIT as a rerouting LU. If these recommendations are not followed, invalid routing tables may be constructed.

Index

N

P

Q

R

READER COMMENT SHEET

DS/1000-IV
Network Manager's Manual
Volume II

91750-90011 **March 1985**

Update No. __________

(If Applicable)

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO,CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Hewlett-Packard
Information Networks Division
19420 Homestead Road
Cupertino, California 95014
Atten: Customer Information Products Dept.

FOLD FOLD