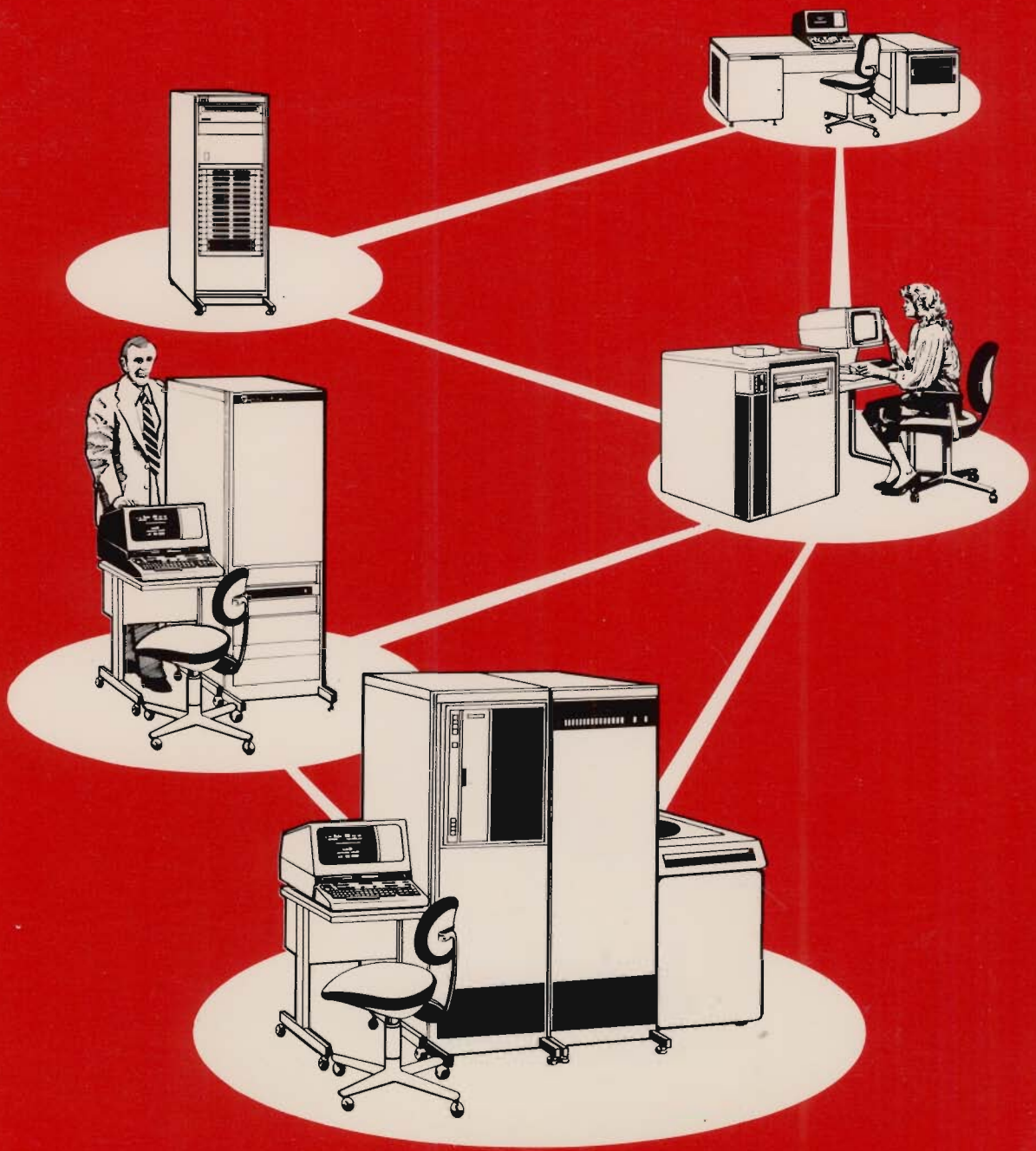
HEWLETT PACKARD

Getting Started with DS/1000-IV



HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Getting Started with DS/1000-IV

HEWLETT PACKARD

19420 Homestead Road, Cupertino, California 95014

Part No. 91750-90004

Printed in U.S.A. April 1982
U0482

PRINTING HISTORY

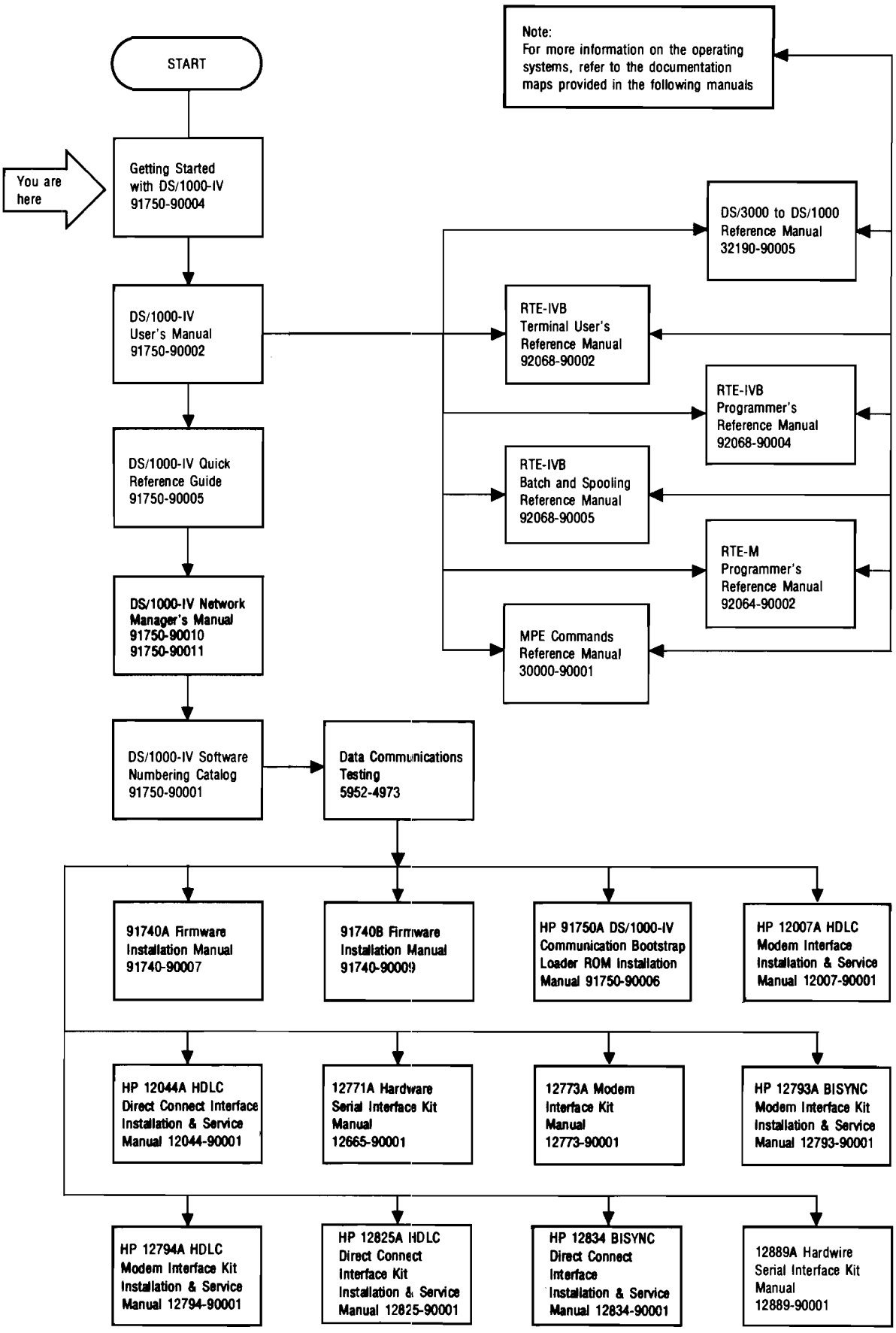
The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in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

First Edition	Oct 1980
Update No. 1	Jul 1981
Update No. 1 Incorp.	Jul 1981
Update No. 2	Apr 1982
Update No. 2 Incorp.	Apr 1982

DOCUMENTATION MAP
DS/1000-IV



PREFACE

This introduction to Distributed Systems/1000-IV presents the fundamentals of DS/1000-IV. This manual will be most valuable if used as a workbook, trying each example as you read the explanations of the different network features.

By following the examples, you will learn how to send operator commands from one system to another. You will learn how to use the editor at a remote system and how to transfer files from one system to another. Remote Program Development is discussed and you will learn how to compile, load and run programs at a remote system. DS/1000-IV program calls are shown allowing you to perform various functions remotely from a program. Some of these include Remote File Access calls, DEXEC calls (which are similar to EXEC calls), and Program-to-Program calls which provide a data interface between two or more programs on various systems. Several utility calls are discussed which provide unique DS/1000-IV functions, such as remote log-on and log-off from systems in a session environment.

Before using this manual, you should be familiar with the operating systems being used in your network. Refer to the appropriate introduction manuals for the various systems in your network.

There are many network possibilities containing any one of the assorted types of HP 1000 computer systems supported by DS/1000-IV. As an aid in making this manual an effective learning tool, this manual will assume your network includes at least two RTE-IVB systems (both containing DS/1000-IV) and an HP 3000. You should therefore have a basic user-level understanding of the RTE-IVB (Real-Time-Executive) operating system. If your network includes an HP 3000, you should also have a basic user-level understanding of the MPE (Multi-Programming Executive) operating system. These assumptions are made to make this manual simple to use and are not meant to exclude interfacing with RTE-M, RTE-L, RTE/XL and RTE-IVB MTM (Multi-Terminal-Monitor) nodes. All references to RTE-L in this manual also apply to RTE-XL unless stated otherwise, likewise all references to RTE-MIII apply to RTE-IVE unless stated otherwise.

If your network is slightly different, many of the examples presented in this manual will still operate correctly except for a few minor exceptions. If your network does not include an HP 3000, you may skip over these sections, though you may find the numerous capabilities available interesting.

This manual also assumes, although not as important, that you know how to program in Fortran or Assembly. You should therefore have the appropriate language manual handy in addition to your system's programming manuals while doing the exercises in order to check errors or to determine correct statement formats.

Occasionally this manual will recommend that you consult with the Network Manager (the person who has overall responsibility for the network). This is a good idea when you may encounter an unexplainable error or if an example does not run as expected. If the Network Manager is not available to assist you, go to the next most experienced person at your site.

Before attempting any of the examples presented in this manual, obtain a diagram of your network from your Network Manager. This diagram will be most useful if it includes:

1. What computer systems exist in your network (i.e., RTE-IVB, RTE-L, RTE-M, RTE-XL, etc.)?

2. What computer systems are directly connected to one another?

3. The node number of each node.

4. The LU number of each DS/1000-IV line.

You may not understand the significance of these items at this moment, but will find them useful when attempting the examples presented in this manual.

In addition to these items, it will also be useful to determine:

1. What nodes are off limits?

2. Which accounts, if any, may you access?

3. Which cartridges may you use?

4. What is the System LU of your terminal?

5. Which nodes have dics and which ones can you use?

Table of Contents

Chapter 1
Introduction

The HP Distributed Systems 1000-IV (DS/1000-IV) is an integrated set of high-level facilities and procedures which support distributed network connections between HP computer systems. These connections can be made over hardwired lines, which connect one system directly to another, or over telephone lines via modems.

DS/1000-IV provides you with many advantages over single computer systems. Briefly, some of these are:

- Moving or accessing large and small amounts of data or programs on other systems in the network.
- Communicating between user-written application programs on different systems.
- Sharing peripheral devices on one system by users at other systems in the network.
- Scheduling programs remotely from another system. This may be done programmatically or from a remote terminal.
- Sending system commands from one system to be processed on another system.
- Developing, loading and compiling programs residing on remote systems.
- Preparing a system generation and down-loading it to a remote CPU.
- Communicating and transferring data between any two HP 1000 systems regardless of your network's design using the Store and Forward capability of DS/1000-IV.

This manual will introduce you to many of these capabilities and point out some possible applications. Remember that this manual is only an introduction to Distributed Systems/1000-IV and therefore illustrates many of the interactive uses of DS/1000-IV as a quick learning tool instead of showing more of the programmatic uses. Working interactively with DS/1000-IV should only be thought of as a small part of DS/1000-IV's total function. Refer to your other DS/1000-IV manuals for further discussion on the many other capabilities of Distributed Systems/1000-IV.

Chapter 2
Networks

When your network is first designed, each HP 1000 system in the network is given a node number and paths between nodes are set up by your Network Manager. The possible paths of communication are determined by the configuration of your network. In some networks, there may be only one path from one node to another. In other networks, there may be several. The Network Manager determines the routes between the systems in your network either by creating a routing table or by making use of the automatic route-choosing features of DS/1000-IV.

A Distributed System network may consist of a number of different HP 1000 or HP 3000 computer systems in a variety of different configurations. One possible network may look something like Figure 1. Each HP 1000 node in the network can communicate with any other HP 1000 node simply by providing the appropriate nodal address.

For example, in Figure 1, nodes 12, 13 and 14 may all communicate with one another. Information from node 12 or 13 is passed to node 11 and then node 11 automatically passes the information to its destination, node 14. This process of storing information in one node, then passing it on to another node until the information reaches its destination, is called STORE-AND-FORWARD. This feature allows you to simply provide the node number of the computer system you wish to communicate with without worrying about what path the data takes to arrive at its final destination.

The HP 3000 may be accessed only if the program you are running (RMOTE or an application program) is located at a node directly connected to the HP 3000. A node that is directly connected to another node is called a NEIGHBOR node.

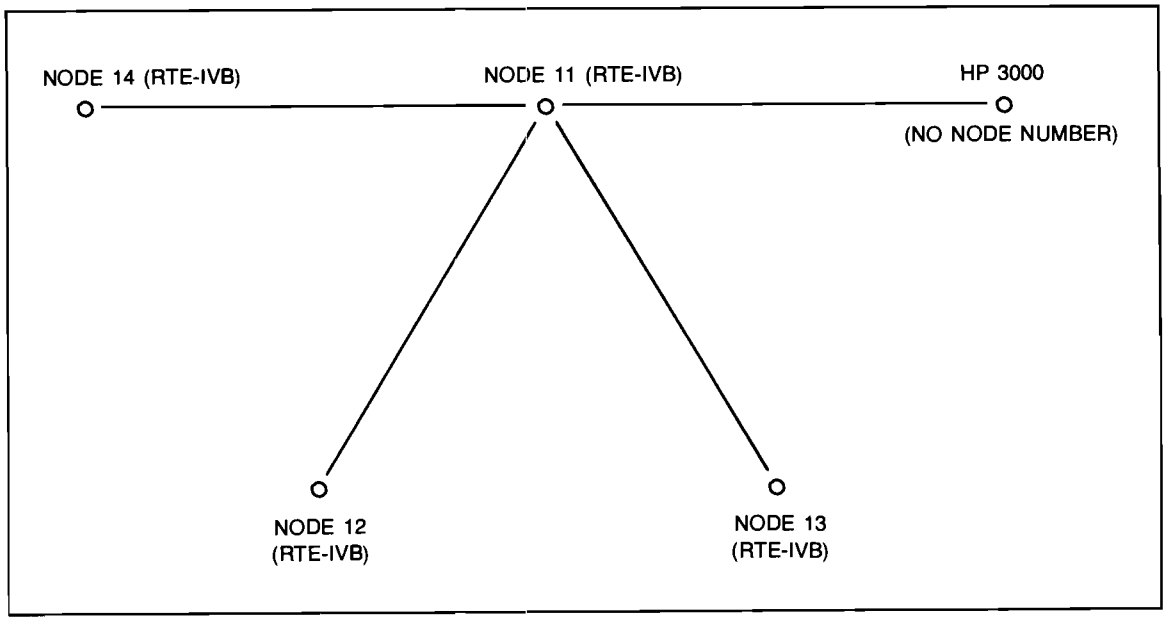


Figure 1. Example Network Configuration

Your network will probably not look like the one in Figure 1. You may not have an HP 3000, or you may have an entirely different network configuration. The placement of computers and their interconnections in your network depends upon the application for which it was designed. The examples presented in this manual will use the network depicted in Figure 1.

Chapter 3
1000 To 1000 DS Network

RTE OPERATING SYSTEM

For the following examples, let us assume you are at an RTE session terminal and running under FMGR. You should have a colon (:) as a prompt.

When you have this prompt, you may enter any RTE command.

```
:SYTI <-------------------------- Enter this command. The characters "SY" before the
                                  TIme command allows you to execute RTE system
                                  commands from FMGR.

1980  336  15  6  59 <---------- You will receive a response that appears like this.
                                  The year, Julian day, hour, minutes and seconds
                                  currently recorded in the real-time clock are
                                  displayed.
```

When the SYTI command is issued, it is sent to the local operating system.

REMAT

If you want to send a command to a remote operating system you can use the program REMAT.

```
:RU,REMAT <-------------------- Enter this command.

$ <---------------------------- REMAT will prompt you with a dollar sign.
```

At this point any command entered will still be sent to your local operating system even though you have a dollar sign as a prompt.

```
$TI <-------------------------- Enter this command

1980  336  15  7  20 <--------- You will get a response like this.
```

To find what cartridges are mounted in your local session try the following:

```
$CL <-------------------------- Enter this command

                                        (your local node number)
                                                  |
                                                  v
  LU  LAST TRACK   CR   LOCK  REMOTE NODE= 00012

  02     00255   00002
  03     00255   00003
  31     00202   45523
  54     00097   21328
```

Now find what files exist in your local session on a particular cartridge:

```
$DL,3 <------------------------ Enter this command to find what files exist in
                                cartridge 3.
```

```
(cartridge label)                          (node number)   (cartridge number)
      ↓                                         ↓               ↓
ILAB= AUX      REMOTE DLIST: NODE=  00012  CR#= 00003  DIR TRKS= 01

NAME      TYPE  #BLKS/LU OPEN TO
RTFILE    00003   00001
ADVSAV    00001   00080
DSOMAN    00004   00045
PREP      00004   00010
%CL12     00005   00002
&CL12     00004   00002
*NAMRC    00004   00001
$KHS      00005   00010   FMG70
QINFO     00004   00035
DSUSER    00004   00002
FIG5      00003   00024
A91740    00004   00012   FMG60
SEC33     00004   00006
&NAMRC    00004   00013
%NAMRC    00005   00009
%RASK     00005   00012
%DBMS3    00005   00048
CHASE     00003   00126
ACC       00004   00006
```

To send a command to a remote system, more commonly referred to as a remote node, you must tell REMAT the node where you wish the command to execute. When a DS/1000-IV Network is designed, each HP 1000 node is given a unique number. In our example network, the nodes are numbered 11, 12, 13 and 14. These numbers are referred to as NODE NUMBERS or NODE ADDRESSES.

To find what your local node number is, try the following:

```
$LC <-------------------------------  Give this command.

LOCAL NODE  = 12 <------------------  This is the Node Number assigned to your local
                                      operating system.
```

Assume you are at node 12 and you want to execute some commands at node 13 (as shown in Figure 2).

```
$SW,13,12,DS <----------------------  Enter the SWitch command. The number 13 is the
                                      remote node from which commands are executed.
                                      The number 12 is the local node number to which
                                      data from certain REMAT commands that specify
                                      two nodes (DU, LI, and ST) are sent. Data from
                                      commands that require only one node are always
                                      returned to the local node where REMAT is execut-
                                      ing. These different types of REMAT commands will
                                      be discussed in greater detail later on in this chap-
                                      ter. DS is the local node's security code. The number
                                      12 is an optional parameter when its value does not
                                      change from a previous SWitch command.

#TI <-------------------------------  When you are talking to a remote node, the prompt
                                      is a pound sign (#).

1980  336  15  7  40 <--------------  The time at the remote node (node 13) is displayed.
```

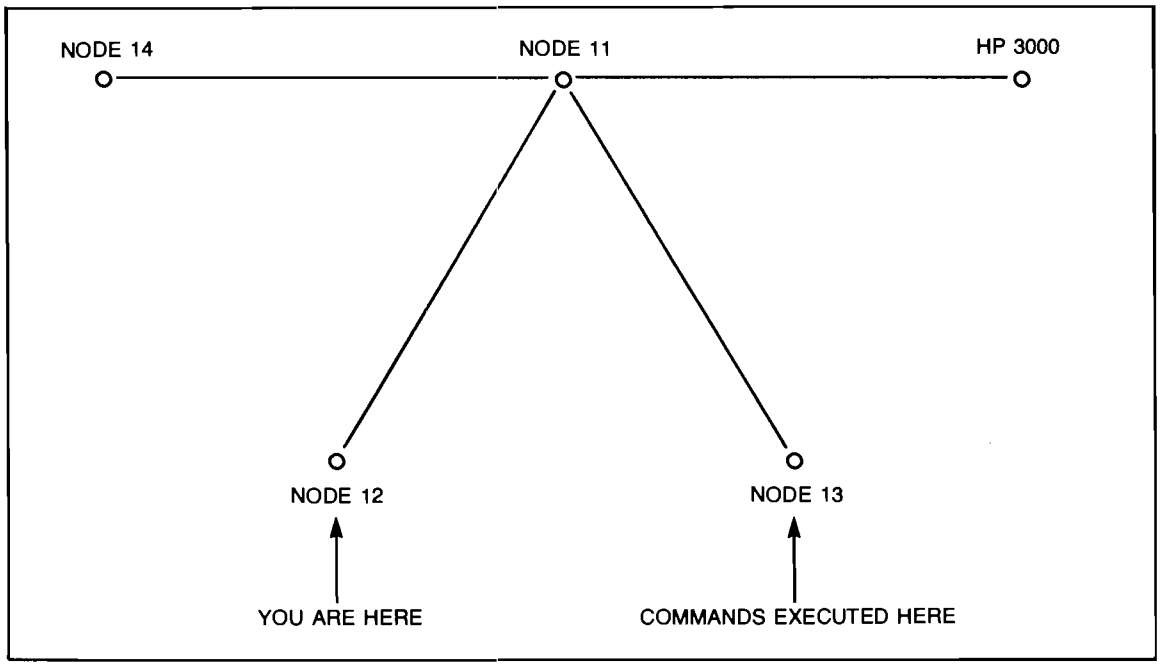


Figure 2. Sending Remote Operator Commands

To find what cartridges are mounted in your remote session try the following:

```
#CL <------------------------ Enter this command.

                                        (remote node number)
                                                 |
                                                 v
LU  LAST TRACK   CR   LOCK  REMOTE NODE= 00013

16      00255   00016
12      00404   23456
19      00113   32767
03      00255   00003
02      00255   00002
```


Now find what files exist in one of the mounted cartridges. A negative LU or positive cartridge reference number may be given.

```
#DL,-12 <------------------------ Give this command

ILAB= LU012   REMOTE DLIST: NODE=  00013  CR#= 123456  DIR TRKS= 01

NAME      TYPE  #BLKS/LU OPEN TO
WHILE.    00004   00001
RPT.      00003   00001
TCODE     00004   00006
FONTS     00004   00036
\DBDSX    00004   00002
/BTD      00004   00001
SWSG1     00006   00007
SWSG2     00006   00010
RT4G1     00006   00008
RT4G2     00006   00018
RT4G3     00006   00019
RT4G4     00006   00016
RT4G5     00006   00017
RT4G6     00006   00016
```

To identify your current local list and log device while running under REMAT, enter the following:

```
#LL <------------------------ Enter this command

LISTLU =     1          LOGLU =     1
```

These values default to your terminal (LU 1) when you first run REMAT.

If you want a listing to be printed on a local printer or an LU other than your terminal, enter the following commands:

```
#LL,6 <------------------------ Give this command to change the current local list
                                device to be the local printer. Check with your Net-
                                work Manager to see if the LU of your printer is an
                                LU other than 6 or if other printers exist. In RTE-
                                IVB systems, you may want to set up spooling to a
                                printer before running REMAT and issuing this
                                command.

#DL,-12 <---------------------- The output from this command will now go to the
                                local printer.

#LL,1 <------------------------ Be sure to enter this command to change the local
                                list device back to your terminal.
```

If you wish to identify the origin node where data originates (i.e., where commands are executed) and the destination node where, in certain cases, data is sent, in addition to the account name you are logged on under, enter the following command:

```
#SW <------------------------ Enter this command.

NODE1 =     13          NODE2 =     12
ACCOUNT NAME =          ACCOUNT NAME =
DEFAULT SESSION         USER GENERAL
     ^                       ^
     |                       |
origin node             destination node
```

In a Session environment, DS/1000-IV allows you to specify an account name in the remote node when issuing a SWitch command. REMAT will log you on under this account name and subsequently, all referenced files must reside on cartridges available to the specified account.

1. If you use REMAT to log on under a specific account name that exists in a remote node, you may reference those files that reside on a private or group cartridge available to that account name, as well as system cartridges, which are globally mounted and accessible to all accounts.

   For example, if you wish to log on to the account TECH.MARK on node 13, you would enter the following commands from REMAT:

   ```
   #SW,13:TECH.MARK/PASSWORD,,DS

         or

   #SW,13,,DS
   #AT,TECH.MARK/PASSWORD
   ```

   The command AT is a REMAT command that attaches you to a specific account in the node specified in a previous SWitch command.

2. If no account name is given, REMAT will log you on under a DEFAULT SESSION. You may reference those files that reside on a private, group, or system cartridge available to the default session account name.

3. If your remote node is a Multi-Terminal-Monitor node (non-session) then no account structure exists, so all files are accessible.

For simplicity, we took the default account name at node 13 as shown by the ACCOUNT NAME for node 13 showing "DEFAULT SESSION". For further information on how to log on under a specific account name in the remote node, refer to the DS/1000-IV User's Manual under the discussion of the SWitch command.

In our example, NODE1 (node 13) is the origin node while NODE2 (node 12) is the destination node. This may be confusing at first, but if you keep the concept of origin and destination nodes in mind you will find the SWitch command very easy to work with.

Remember that the first node given in the SWitch command is the origin node where REMAT commands are always executed. The second node given in the SWitch command is the destination node where data is sent when certain commands (DU, LI, or ST) requiring two specific nodes are entered. All other REMAT commands will always list data to the local node where REMAT is executing.

```
:SW,13,12,DS
   ↑  ↑
   |  destination node
originating node
```

Now suppose you want to create a file in a remote node and store data in this file from your local node. Let's say you want to create this file at remote node 13. Refer to Figure 3.

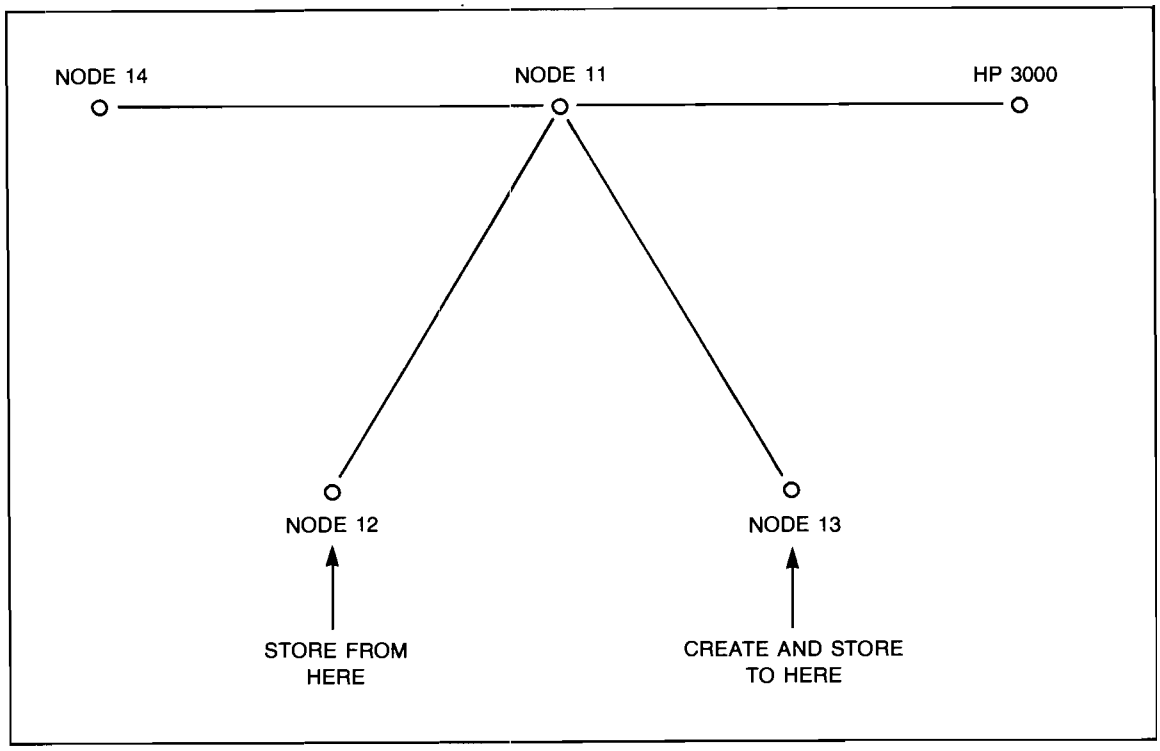


Figure 3. Create and Store Remote Files

```
#SW,12,13,DS
```
Since you will want to STore from node 12, the origin node, to remote node 13, the destination node, set the value of NODE1 to 12 and the value of NODE2 to 13. DS is the local security code and is an optional parameter if it has already been given in a previous SWitch command.

Specifying the correct node values in this SWitch command is slightly different from when you were specifying node values for TI, CL, and DL commands. In both cases, the REMAT command is executed at NODE1, however, in an ST command (as well as in the DU and LI command) data is sent to NODE2, whether it is the local node where REMAT is executing, or a different node.

Once the SWitch command has been properly entered, a file may then be created in a remote node and data sent to this file from your local node. There are various ways to create and develop a remote file. You may use your local or remote editor, or you may develop a file from your terminal, sending records directly to the remote file from your terminal. In a later section we will discuss development of a file from an editor. For learning purposes we will now develop a file directly from your terminal.

```
#ST,1,URFILE
```
STore from your terminal, LU 1, at node 12 (your node) to the file URFILE at remote node 13. The file will be created for you.

```
/
```
Your prompt for input to the file is a slash (/). You should wait to receive this prompt before entering data.

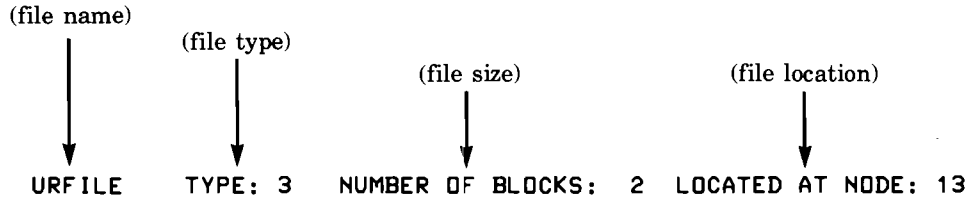
```
/FTN4,L
/       PROGRAM PROG1
/       DIMENSION NAME (10)
/       DATA NAME/10*2H  /
/       WRITE(1,30)
/  30   FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
/       READ(1,40) NAME
/  40   FORMAT(10A2)
/       WRITE(1,50)NAME
/  50   FORMAT("YOUR PROGRAM WORKS, ",10A2)
/       END
```
Enter a control-D here to mark an end of file

```
#
```
The pound sign (#) is now your prompt again.

Next let's list this file on your terminal.

```
#SW,13,12
```
You must change the node values again. In the previous SWitch command, you were sending data from your local node to the remote node. Now, you will want to send data from the remote node to your local node in order to receive a listing of URFILE.

```
#LI,URFILE,1
```
The number 1 is optional since this parameter defaults to the current local list device which is presently LU 1 (your terminal).




```
0001  FTN4,L
0002        PROGRAM PROG1
0003        DIMENSION NAME (10)
0004        DATA NAME/10*2H  /
0005        WRITE(1,30)
0006    30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
0007        READ(1,40) NAME
0008    40  FORMAT(10A2)
0009        WRITE(1,50)NAME
0010    50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
0011        END
```

If desired, you could list URFILE to your left cartridge unit (logical unit 4).

```
:LI,URFILE,4 <-------------- Enter this command.
```

Another way to list the file (or send it to your left cartridge unit) is to use the DUmp command. This command will list the file without line numbers.

```
:DU,URFILE,1 <-------------- Enter this command

FTN4,L
      PROGRAM PROG1
      DIMENSION NAME (10)
      DATA NAME/10*2H  /
      WRITE(1,30)
  30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
      READ(1,40) NAME
  40  FORMAT(10A2)
      WRITE(1,50)NAME
  50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
      END
```

Try DUmping this file to your left cartridge unit.

So far you have seen some of the ways to manipulate files between similar nodes. Nodes may differ depending on what remote system and account name you are accessing.

It is important to keep in mind the origin and destination node concept when using REMAT's SWitch command. A difficulty that occurs in REMAT is made by reversing the order of the origin and destination node numbers. After issuing the SWitch command, you should make it a habit of entering a CL command. This command will not only give you a list of cartridges available, but it will also identify the origin node.

You have also seen some of the commands from the previous examples you are able to use. Most RTE operator commands can be used from REMAT subject to account capability limitations. Look at your RTE programmer's manual and try some more on your own.

Finally, to terminate REMAT you give the EXit command.

```
:EX <---------------------- Give this command.
```

RTE EDITOR

To change existing files or to create new ones, you use the editor.

Remember, depending on what account name you have REMAT log you on under, only certain cartridges may be accessed. Even though you are aware of a particular file existing on a remote system, you will not be able to access that file if you are logged on under the wrong account name.

In DS/1000-IV you have many options available when editing a file. The most commonly used one is to schedule the remote editor from REMAT using the RW command.

For example, assume you are at node 12 (an RTE-IVB node) and you want to edit the file URFILE (created in an earlier example) which is at node 13. For learning purposes, let us also assume that the only editor in the network exists at node 14 (an RTE-IVB node). This configuration is shown in Figure 4.

Since, in this example, the file to be edited needs to be at the same node as the editor, you must transfer the file from node 13, where the file exists, to node 14 where the editor resides.

```
:RU,REMAT <-------- Use REMAT.

$SW,13,14,DS <-------- You want to store the file from node 13 (origin node)
                       to node 14 (destination node).

#ST,URFILE,SOURCE <-------- Store the file URFILE in node 13 to a new file
                            SOURCE in node 14. URFILE will remain
                            unchanged.
```

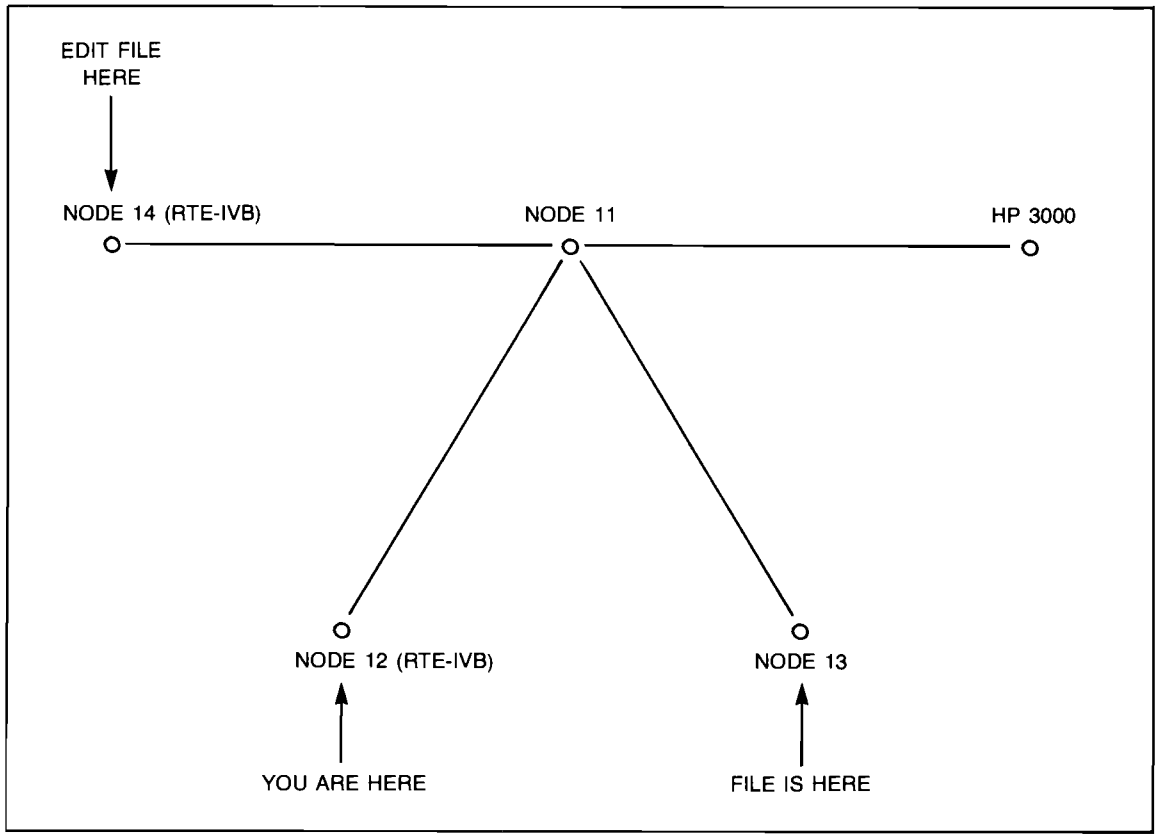


Figure 4. Editing a File at a Remote Node

Now that the file and editor are located in the same node, you need to enter another SW command so that the remote editor may be scheduled from your local node.

```
#SW,14,12 <-------------------- Node 14 is where the editor is located and node 12 is
                                your local node.

#RW,EDITR,1,,14,12 <---------- Run with wait the remote editor located at node 14.
                                The number 1 is the session LU number of your
                                terminal. The numbers 14 and 12 are the remote and
                                local nodes respectively. These values are required
                                so that the remote editor knows who to communicate
                                with. This command string works properly when the
                                remote node has Session Monitor. If session does not
                                exist in the remote node, then the last two charac-
                                ters (TR) of EDITR must be replaced by a suffix (T9,
                                for example) in order to schedule a specific copy of
                                the editor. You can get the name of your remote
                                copy, if necessary, from your Network Manager.
                                Refer to the DS/1000-IV User's Manual for further
                                discussion on the various ways to schedule a remote
                                editor.

EDITING AT NODE 14 <---------- You will receive this message indicating the node at
                                which you are editing.

SOURCE FILE? <---------------- The remote editor asks this.

/SOURCE <--------------------- The editor prompt is a slash (/). Give the name of the
                                file you wish to edit.

FTN4,L <---------------------- The first line of the file is printed by the editor.

/L20 <------------------------ List 20 lines.

  FTN4,L
        PROGRAM PROG1
        DIMENSION NAME (10)
        DATA NAME/10*2H  /
        WRITE(1,30)
   30   FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
        READ(1,40) NAME
   40   FORMAT(10A2)
        WRITE(1,50)NAME
   50   FORMAT("YOUR PROGRAM WORKS, ",10A2)
        END

EOF <------------------------- An EOF signifies the end of the file.

/4 <-------------------------- Make line 4 the pending line.

        DATA NAME/10*2H  /<--- Line 4 is displayed.

/       CALL DNODE(14) <------ Enter a space as the first character immediately
                                after the slash and then the line you want to add.
                                DNODE(14) is a DS/1000-IV utility call which di-
                                rects I/O requests to a different node (node 14 in this
                                example). These calls will be discussed further in a
                                later section.
```

When you are done editing a file, list it to your terminal to verify your edits were made correctly.

```
FTN4,L
      PROGRAM PROG1
      DIMENSION NAME (10)
      DATA NAME/10*2H  /
      CALL DNODE(14)
      WRITE(1,30)
  30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
      READ(1,40) NAME
  40  FORMAT(10A2)
      WRITE(1,50)NAME
  50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
      END

/ER <-------------------------- Exit the editor and replace the old file with the
                                edited file.
```

This sequence of commands can be used regardless of the node at which you are located.

1. Move the file to the node where the editor is, or use the editor where the file is located.
2. Schedule the editor you wish to use.
3. Name the file to be edited. Remember that depending on what node you are in and what account name you are logged under, only certain files may be accessed.
4. Edit the file using standard editor commands. These commands are described in the editor reference manual.
5. Exit the editor and create a new file or replace the old file with the edited file.

Depending on your location and needs, different editors are used. Refer to Table 1.

1. To schedule a local RTE-M editor, give the following command:

   ```
   :RU,EDITM
   ```

2. To schedule an editor at an RTE-IVB non-session node from an RTE-M node, enter the following command:

   ```
   :RU,REDIT,,,rn,,sfx
   ```

   where rn is the number of the node at which the editor is to be scheduled and sfx is the last two characters of the remote editor's name (you can get the remote editor's name from your Network Manager). The other parameters are default parameters that allow you to change logical devices, line lengths and flags. Refer to the DS/1000-IV User's Manual for further discussion on these parameters.

3. To schedule a local RTE-L editor, give the following command:

   ```
   FMGR : RU,EDITR
   ```

4. To schedule an editor at an RTE-IVB node with Session Monitor from an RTE-M or RTE-L node, enter the following commands from REMAT:

   ```
   #SW,rn,,pw
   #RW,EDITR,lu,,rn,ln
   ```

where rn is the number of the node at which the editor is to be scheduled, pw is the local node's security code, lu is your local terminal's system LU, and ln is the local node number of your system. As with any remote program scheduled by REMAT using the RW command, you have a 20 minute time limit to complete your editing.

5. To schedule a local RTE-IVB editor, give the following command:

```
:RU,EDITR
```

6. To schedule an editor at a remote RTE-IVB node from another RTE-IVB node, enter the following commands from REMAT:

```
#SW,rn,,pw
#RW,EDITR,lu,,rn,ln
```

where the parameters are the same as described in option 4 above.

Table 1. What Editor To Use?

LOCAL NODE	LOCATION OF EDITOR	EDITOR SYNTAX
RTE-M	Local RTE-M node	:RU,EDITM
RTE-L	Local RTE-L node	:RU,EDITR
RTE-M	Remote RTE-IVB without Session Monitor	:RU,REDIT,lu,,rn,,sfx
RTE-M,RTE-L	Remote RTE-IVB node with Session Monitor	#SW,rn,,pw #RW,EDITR,lu,,rn,ln
RTE-IVB (Session)	Local RTE-IVB	:RU,EDITR
RTE-IVB (Session)	Remote RTE-IVB (Session)	#SW,rn,,pw #RW,EDITR,,,rn,ln

where:

ln = local node number
sfx = prefix
pw = password
lu = local terminal's system LU
rn = Node where the editor is to be scheduled

This table is only a list of the most commonly used editors in a network designed similar to our example. Refer to DS/1000-IV User's Manual for a more detailed discussion of all the editor possibilities.

REVIEW

So far, you have learned how to use some of the commands from REMAT and how to send commands to remote nodes and receive results from these nodes. When you have finished this manual, you can find out more about REMAT in the DS/1000-IV User's Manual.

You have also learned how to run the editor program from any node in the network. You can create files, transfer files, and edit files at any node that supports a file system.

Up to this point we have shown you a few of the interactive uses of DS/1000-IV at the operator level. There are many other areas of DS/1000-IV we have not covered, such as program development, DS/1000-IV program calls, DEXEC calls, Program-to-Program calls, initializing nodes and many more. In later sections we will expand on some of these topics.

PROGRAM DEVELOPMENT

This section will show you how to put a program in a file, compile, load and run it. You can do these steps at your local node, at a remote node, or at an RTE-IVB node with the program being down-loaded to an RTE-M or RTE-L node. Down-loading is the process whereby the Absolute Program Loader (APLDR) in a RTE-M or RTE-L node reads a program file from an RTE-IVB node and loads it into memory in the RTE-M or RTE-L node.

In the following sections we will discuss:

1. Local Program Development (compiling and loading programs at the local node).

2. Remote Program Development (compiling and loading programs at a remote node).

3. Adding DS/1000-IV Program Calls to your programs.

All these examples assume that you are at node 14, an RTE-IVB node. See Figure 5.

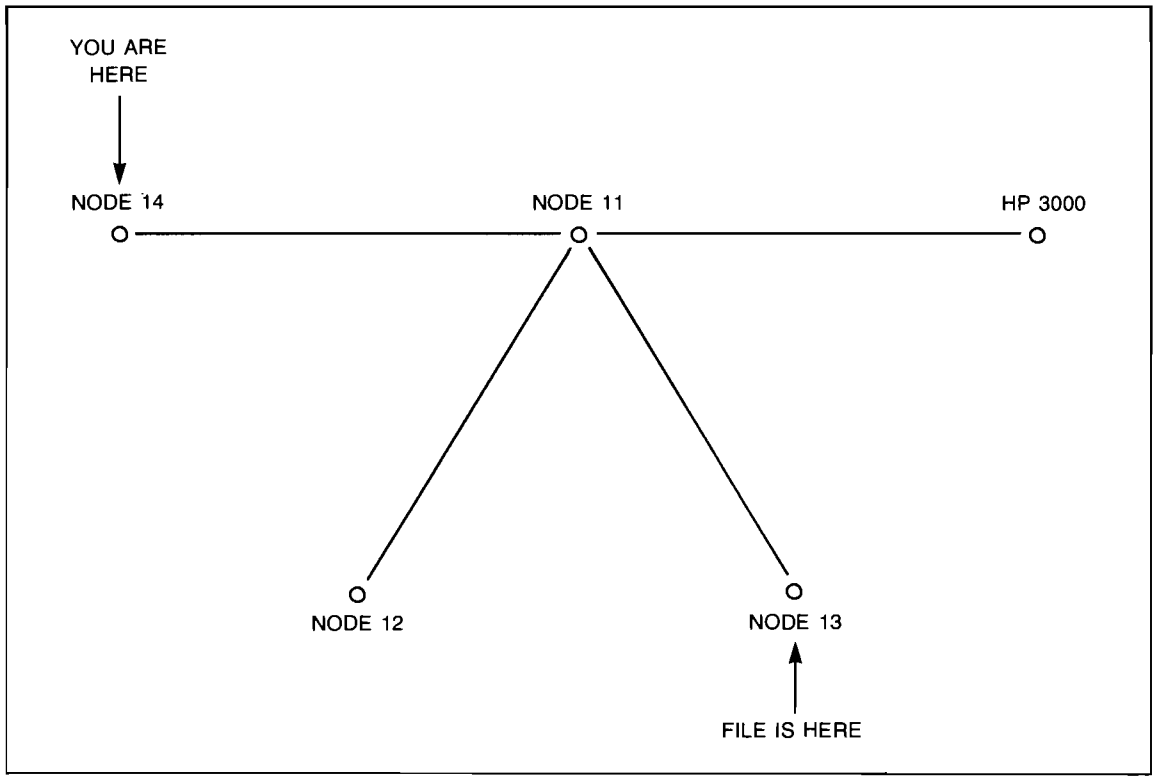


Figure 5. Program Development

LOCAL PROGRAM DEVELOPMENT

To run the compiler and loader locally, you need the file that is to be loaded located at the same node as the loader. Since the file in our example is at node 13 and the compiler and loader are located at node 14 (the local node), you need to move the file from node 13 to node 14.

```
:RU,REMAT <-------------------- Use REMAT.

$SW,13,,DS <------------------- SWitch node 13 to be the origin node and default
                                 your local node (node 14) to be the destination node.

#ST,URFILE,&PROG1 <------------ STore from node 13 to node 14.

#EX
```

Now that the file &PROG1 is in your local node, run the local compiler using &PROG1 as your source file.

```
:RU,FTN4,&PROG1,1,%PROG1 <---- Enter this command. The source file is &PROG1, the
                                listing is to go to LU 1 (an optional parameter that
                                defaults to your terminal) and the relocatable file
                                name is to be %PROG1.
```

The FTN4 listing follows.

```
PAGE 0001  FTN.   2:58 PM  WED., 19  DEC., 1980

0001  FTN4,L
0002        PROGRAM PROG1
0003        DIMENSION NAME (10)
0004        DATA NAME/10*2H  /
0005        WRITE(1,30)
0006    30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
0007        READ(1,40) NAME
0008    40  FORMAT(10A2)
0009        WRITE(1,50)NAME
0010    50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
0011        END

  FTN4 COMPILER: HP92060-16092 REV. 2001 (791101)

**  NO WARNINGS **  NO ERRORS **  PROGRAM = 00078  COMMON = 00000

$END FTN4: NO DISASTRS   NO ERRORS   NO WARNINGS
```

If no disasters, errors, or warnings occur, run the local loader.

```
:RU,LOADR,,%PROG1,1 <---------- Give this command. The first parameter, the device
                                or file namr for command entries, defaults to your
                                terminal (LU 1). The relocatable file is %PROG1 and
                                the listing is to go to your terminal.
```

A sample loader listing follows:

```
PROG1   42042 42157

FMTIO   42160 43456   24998-16002 REV.1926 790417
FRMTR   43457 47114   24998-16002 REV.1926 790503
FMT.E   47115 47115   24998-16002 REV.1901 781107
PNAME   47116 47163   771121  24998-16001
REIO    47164 47310  92067-16268 REV.1903 790316
```

```
 4 PAGES RELOCATED  4 PAGES REQ'D  NO PAGES EMA  NO PAGES MSEG
LINKS:BP    PROGRAM:BG   LOAD:TE   COMMON:NC
 /LOADR:PROG1  READY AT  3:00 PM  WED., 19  DEC., 1980

   /LOADR:$END
```

Now try running the program, PROG1

```
:RU,PROG1 <-------------- Enter this command.

PLEASE TYPE IN YOUR NAME AND <RETURN>
KELLY'S FRIEND
YOUR PROGRAM WORKS, KELLY'S FRIEND
```

REMOTE PROGRAM DEVELOPMENT

Program development may be done from any node with terminal capabilities to any other node. The node, however, upon which the various areas of program development are being done must contain those resources (compiler, loader, files, etc.) that are necessary, although the compiler could be on one node and the loader on another.

In the following examples, you are at node 14 and the program is developed and run at node 11. See Figure 6.

Remote program development is very similar to local program development. The major difference is that you run the loader and compiler remotely using REMAT. Since you are going to develop your program at node 11 and the file is at node 14, you need to STore the file, &PROG1, from node 14 to node 11.

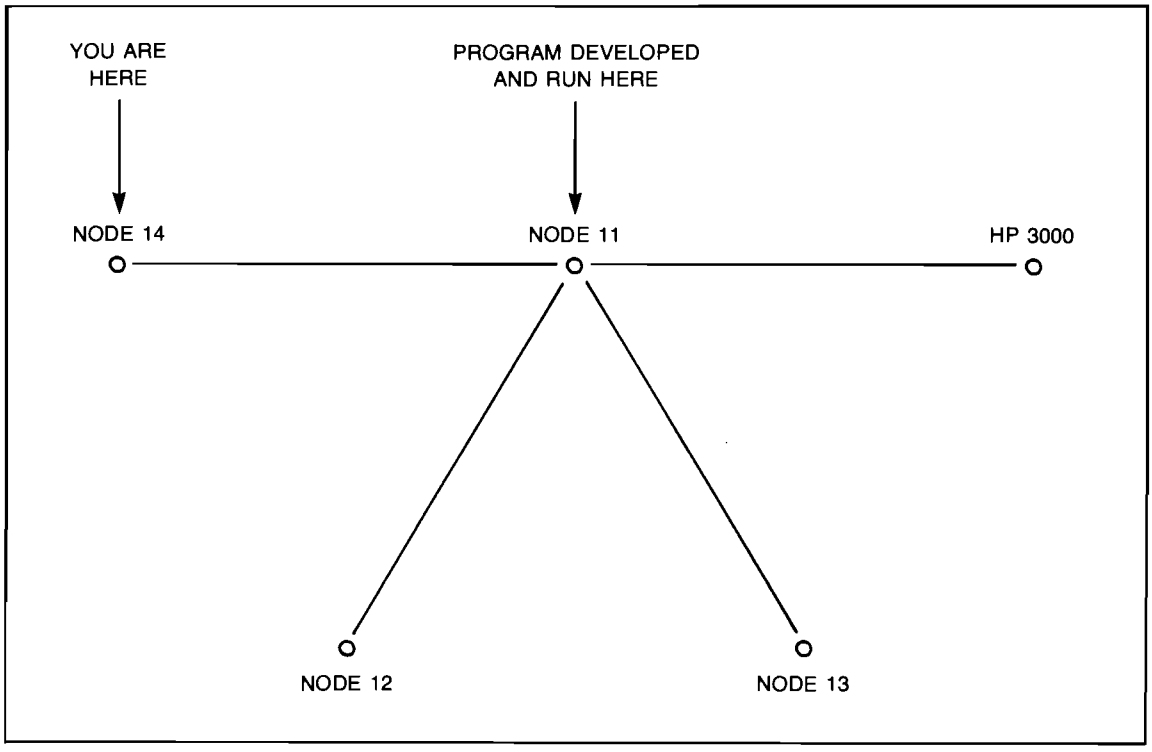


Figure 6. Remote Program Development

```
:RU,REMAT <-------------------- Use REMAT.

$SW,14,11,DS <----------------- Give this command to make node 14 the origin node
                                and node 11 the destination node.

#SW <-------------------------- Confirm you switched properly.

NODE1 =      14          NODE2 =      11
ACCOUNT NAME =           ACCOUNT NAME =
USER.GENERAL             DEFAULT SESSION
   ↑                        ↑
origin node              destination node

#ST,&PROG1,&PROG1 <------------ Transfer the file from node 14 to node 11.

#SW,11,14 <-------------------- Another SWitch command needs to be given here to
                                reverse the origin node so that now all listings and
                                command executions will originate from node 11.

#CL <-------------------------- Another way to confirm your switch other than
                                using the SW command is to enter a Cartridge List
                                command. This will not only give you the origin
                                node number, but it will also give you a list of car-
                                tridges available depending on the account you are
                                logged on under.

LU  LAST TRACK   CR   LOCK  REMOTE NODE= 00011

53      00101   19789
43      00202   19523
44      00202   18225
02      00202   00002
03      00255   00003
47      00074   21328
```

Now run the compiler located at node 11 using the command RW (Run with Wait). This command causes REMAT to schedule the compiler at node 11 and then wait for the compiler to complete execution before executing the next REMAT command. The source file is &PROG1, the list file is to be 'PROG1 and the relocatable output file is to be %PROG1. The last two file names are optional and may be replaced by a dash (-). When a dash is given, the compiler will create two files by the same name that we give below.

```
#RW,FTN4,&PROG1,'PROG1,%PROG1 <-- Give this command.
```

The compiler returns the following:

```
000000  000000  000000  000000  003721       Q
```

These parameters are returned by the compiler and are described in the error section of the Fortran Reference Manual. Briefly, the five parameters reference the total number of errors, number of disasters, number of regular errors, number of errors and the revision level of the compiler.

To verify that your program compiled correctly without any warnings or errors, list the list file to your terminal.

```
#LI,'PROG1 <------------------- Give this command.

'PROG1   TYPE:  4   NUMBER OF LOCKS:  4   LOCATED AT NODE:  11
0001  1
0002   PAGE 0001  FTN.   3:02 AM  WED., 19  DEC., 1980
```

```
0003
0004
0005  0001  FTN4,L
0006  0002        PROGRAM PROG1
0007  0003        DIMENSION NAME (10)
0008  0004        DATA NAME/10*2H  /
0009  0005        WRITE(1,30)
0010  0006    30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
0011  0007        READ(1,40) NAME
0012  0008    40  FORMAT(10A2)
0013  0009        WRITE(1,50)NAME
0014  0010    50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
0015  0011        END
0016
0017
0018
0019    FTN4 COMPILER: HP92060-16092 REV. 2001 (791101)
0020
0021
0022  ** NO WARNINGS ** NO ERRORS ** PROGRAM = 00078 COMMON = 00000
```

Run the loader located at the same remote node using %PROG1 as the relocatable file and 'PROG1 as the list file. The missing parameter is the optional command file.

```
#RW,LOADR,,%PROG1,'PROG1 ◄— Give this command.
```

The loader returns the following parameters:

```
050122  047507  030440  020040  020040      PROG1
```

These parameters are the program's name (in octal and in ASCII), if no errors occured, or the loader error code if an error did occur.

To verify that your program loaded properly, list the list file to your terminal.

```
#LI,'PROG1 ◄———————————— Give this command.

'PROG1   TYPE:  4   NUMBER OF BLOCKS:  4   LOCATED AT NODE:  11
0001  PROG1  40042 40202
0002
0003  FMTIO  40203 41501   24998-16002 REV.1926 790417
0004  FRMTR  41502 45137   24998-16002 REV.1926 790503
0005  FMT.E  45140 45140   24998-16002 REV.1901 781107
0006  PNAME  45141 45206   771121  24998-16001
0007  REIO   45207 45333  92067-16268 REV.1903 790316
0008
0009  4 PAGES RELOCATED  4 PAGES REQ'D  NO PAGES EMA  NO PAGES MSEG
0010  LINKS:BP    PROGRAM:BG   LOAD:TE   COMMON:NC
0011   /LOADR:PROG1  READY AT  3:05 PM  WED., 19  DEC., 1980
0012
0013    /LOADR:$END
```

If you were to go to node 11 without exiting REMAT at node 14, you could run your program. This, of course, may be impractical, especially if your remote node is some distance away. Up to this point, the examples shown have been designed to communicate only with the system console at the node where the program is running. In the next section, you will learn how to use various DS/1000-IV features so programs may communicate with any device or even another program on any node in your network.

Once you log off from your session at node 14, PROG1 will be removed from memory and would have to be reloaded at the remote node if you wish to run PROG1 again.

DS/1000-IV PROGRAM CALLS

This section is written for the more advanced user who wishes to use the various DS/1000-IV program calls available in an RTE system network.

DS/1000-IV program calls can be divided into four groups:

1. Remote File Access (RFA)
2. DEXEC Calls
3. Program-to-Program Calls (PTOP)
4. Utility Calls

REMOTE FILE ACCESS

Remote File Access calls may be issued from your RTE application program to manipulate and control files either in your local or remote node. These calls are similar to the File Management Package (FMP) calls. DOPEN, for instance, is similar to the FMP call, OPEN

Remote File Access Call–

CALL DOPEN(IDCB,IERR,NAME[,IOPTN[,ISECU[,ICR[,IERLC]]]])

File Manager Progam Call–

CALL OPEN(IDCB,IERR,INAM[,IOPTN[,ISC[,ICR[,IDCBS]]]])

The parameters in these calls specify the name of the file you wish to access, the location of the file, and what function you wish to perform.

For example, the DS/1000-IV call DOPEN opens a file:

CALL DOPEN(IDCB,IERR,NAME[,IOPTN[,ISECU[,ICR[,IERLC]]]])

Four of the parameters specify the file to be opened:

NAME — a three-word array containing the name of the file to be opened.

IOPTN — file open option.

ISECU — the security code, if any, of the file.

ICR — a two-word array containing the disk cartridge on which the file exists and the number of the node where the file exists.

The other parameters are used by the system.

IDCB — information on the file is stored here in a 4-word array and is used exclusively by DS/1000-IV.

IERR — if an error condition occurs, the value representing the error code is returned by the system to this parameter.

IERLC — if an error condition occurs, the number of the node at which the error occurred is returned here.

RFA calls are used whenever you wish to programmatically access remote files. You can, however, also use these calls to access files on your local node. If you are developing a program to be run locally, you may want to use RFA calls instead of FMP calls. This will eliminate having to do any major re-editing of your program in case you move it to another node.

An example program that uses RFA calls DOPEN, DREAD and DCLOS follows:

```
FTN4,L
PROGRAM PROG3

C*************************************************************
C THIS PROGRAM OPENS A SPECIFIC FILE LOCATED AT A REMOTE     *
C NODE AND LISTS A SPECIFIED NUMBER OF RECORDS TO THE        *
C LOCAL USER'S TERMINAL                                      *
C*************************************************************

      DIMENSION IDCB(4),ICR(2),NAME(3),IBUF(40)

C   IL=NUMBER OF WORDS/RECORD TO PLACE IN IBUF
      DATA  IL/40/

      LU=LOGLU(I)

      WRITE (LU,10)
10    FORMAT(//"ENTER NAME OF FILE TO BE OPENED: ")
      READ (LU,20)NAME
20    FORMAT(3A2)

      WRITE (LU,30)
30    FORMAT (/"SECURITY CODE (0=NONE): ")
      READ (LU,*)ISECU

      WRITE (LU,40)
40    FORMAT(/"CARTRIDGE (NEGATIVE LU OR POSITIVE CRN):  "'')
      READ (LU,*)ICR(1)

      WRITE (LU,50)
50    FORMAT(/"NODE LOCATION OF FILE: ")
      READ (LU,*)ICR(2)

      WRITE (LU,60)
60    FORMAT(/"NUMBER OF RECORDS TO READ:  ")
      READ (LU,*)NUM

C   OPEN FILE
      CALL DOPEN(IDCB,IERR,NAME,0,ISECU,ICR,IERLC)

C   CHECK FOR AN OPEN ERROR
      IF (IERR .GE. 0) GO TO 80
      WRITE (LU,70)IERR,IERLC

70    FORMAT (//"ERROR ",I4," OCCURRED AT NODE ",I4)
      GO TO 999
```

```
C   READ RECORDS
80      DO 100 J=1,NUM
        CALL DREAD(IDCB,IERR,IBUF,IL,LEN,0,IERLC)

C   CHECK FOR A READ ERROR
        IF (IERR .GE. 0) GO TO 85
        WRITE (LU,70)IERR,IERLC
        GO TO 110

C   CHECK IF AN EOF HAS BEEN READ
85      IF (LEN .EQ. -1) GO TO 110

C   WRITE RECORD TO TERMINAL (LEN=LENGTH OF RECORD)
        WRITE (LU,90)(IBUF(I),I=1,LEN)
90      FORMAT (40A2)

100     CONTINUE

C   WHEN ALL RECORDS OR EOF IS READ, CLOSE FILE
110     CALL DCLOS(IDCB,IERR,0,IERLC)

C   CHECK FOR A CLOSE ERROR
        IF (IERR .GE. 0) GO TO 999
        WRITE (LU,70)IERR,IERLC
999     STOP
        END
```

When developing a program using any of the DS/1000-IV calls available, such as in the above program, remember to specify Subsystem Global Area (SSGA) when loading your program. For instance:

```
RU,LOADR,,%PROG3,,SS <------- the SS indicates you wish to have access to SSGA.
```

PROG3 will access only those files located in your local session account or those files in the default remote session account. To access files in a specific remote session account, you may add the calls DLGON and DLGOF(Remote log on/log off utilities which allow remote access to specific accounts). If you add this capability to the example program, it would look something like this:

```
FTN4,L
      PROGRAM PROG3

C*****************************************************************
C THIS PROGRAM OPENS A SPECIFIC FILE LOCATED AT A REMOTE      *
C NODE AND LISTS A SPECIFIED NUMBER OF RECORDS TO THE         *
C LOCAL USER'S TERMINAL                                       *
C*****************************************************************

       DIMENSION IDCB(4),ICR(2),NAME(3),IBUF(40),IACCT(16)
                 .
                 .
         <other program code>
                 .
                 .
50     FORMAT(/"NODE LOCATION OF FILE: ")
       READ (LU,*)ICR(2)

       WRITE (LU,55)
55     FORMAT(/"REMOTE SESSION ACCOUNT NAME: ")
       READ (LU,58)(IACCT(I),I=1,16)
58     FORMAT (16A2)
```

```
C   FIND LENGTH OF LAST INPUT STRING
       CALL ITLOG (LEN)

       WRITE (LU,60)
60     FORMAT(/"NUMBER OF RECORDS TO READ:  ")
       READ (LU,*)NUM

C   LOG ON TO REMOTE SESSION
       CALL DLGON (IERR,ICR(2),IACCT,LEN)

C   CHECK FOR AN ERROR
       IF (IERR .EQ. 0) GO TO 67
       WRITE (LU,65)IERR
65     FORMAT (/"REMOTE LOG-ON ERROR  ",I4,)
       GO TO 999

67     WRITE (LU,68)
68     FORMAT (/"LOG ON SUCCESSFUL")

C   OPEN FILE
       CALL DOPEN(IDCB,IERR,NAME,0,ISECU,ICR,IERLC)

C   CHECK FOR AN OPEN ERROR
       IF (IERR .GE. 0) GO TO 80
       WRITE (LU,70)IERR,IERLC
70     FORMAT (//"ERROR ",I4," OCCURRED AT NODE ",I4)
       GO TO 120
                  .
                  .
        <other program code>
                  .
                  .
C   CLOSE FILE
110    CALL DCLOS(IDCB,IERR,0,IERLC)

C   CHECK FOR A CLOSE ERROR
       IF (IERR .GE. 0) GO TO 120
       WRITE (LU,70)IERR,IERLC

C   LOG-OFF REMOTE SESSION ACCOUNT
120    CALL DLGOF(IERR,ICR(2))

C   CHECK FOR AN ERROR
       IF (IERR .EQ. 0) GO TO 999
       WRITE (LU,130)IERR
130    FORMAT (/"LOG-OFF ERROR ",I4)

999    STOP
       END
```

Another way to access a specific remote session is to run REMAT, SWitch to the remote node, and log on to a specific session account using the AT command. Once this is done, you may SWitch back to your local node and then schedule your program. Any requests made by your program to the remote node will be attached to the session created with the AT command. For example:

```
:RU,REMAT
$SW,11,,DS
#AT,TECH.FP/PASSWORD
#SW,LO
$RW,PROG3
```

RTE EXEC CALLS

DEXEC calls are used in programs to communicate with the RTE executive at a remote node. They allow you to do remote write and reads, control I/O devices, schedule programs and other system services.

DEXEC calls are similar to RTE EXEC calls. Their parameters are compatible with the following exceptions:

- DEXEC calls include a "destination node" parameter.
- In addition to regular program status and parameter location information returned to the A-register and B-register, DS/1000-IV error conditions and codes may also be returned.

DEXEC calls are useful when you wish to convert a program developed to run only locally using EXEC calls into a program that will pass remote processing to the EXEC module of local or remote operating systems.

Recall the example program that used FORTRAN WRITE and READ statements.

```
FTN4,L
      PROGRAM PROG1
      DIMENSION NAME (10)
      DATA NAME/10*2H  /
      WRITE(1,30)
  30  FORMAT("PLEASE TYPE YOUR NAME AND <RETURN>")
      READ(1,40) NAME
  40  FORMAT(10A2)
      WRITE(1,50)NAME
  50  FORMAT("YOUR PROGRAM WORKS, ",10A2)
      END
```

If we were to convert this program so that it performed read and write requests with DEXEC calls, it would look something like this:

```
FTN4,L
      PROGRAM PROG2
      DIMENSION NAME (10)
      DATA NODE/14/, LU/20/, NAME/10*2H  /

C DO AN EXEC WRITE AND WRITE THE PROMPT

      CALL DEXEC(NODE,2,LU, 30HPLEASE TYPE YOUR NAME AND <CR>,-30)

C DO AN EXEC READ TO GET THE RESPONSE

      CALL DEXEC(NODE,1,400B+LU,NAME,10)

C  RETURN THE NAME AND CONGRATULATION

      CALL DEXEC(NODE,2,LU,20HYOUR PROGRAM WORKS!!  ,-20)
      CALL DEXEC(NODE,2,LU,NAME,-20)
      END
```

To have this program operate correctly on your system, you would have to give the parameters NODE and LU different data values corresponding to your local node number and the system Logical Unit Number of your terminal. You may get these values from your Network Manager.

DESTINATION NODE NUMBERS. In the above program, we specified in the IDEST parameter the node number at which the DEXEC call is to be executed. This parameter can be specified in three different ways:

1. You may specify the number of the remote node, as was done in the previous example program. In the example network in this manual, the numbers are 11,12,13 and 14

2. You may specify the negative logical unit (LU) number of the communication link to the remote node. This number may only be used for neighboring nodes. In the example network, node 11 and 12 are neighbors. Nodes 12 and 13 are not. When you specify a logical unit number, you must specify the negative of the logical unit number.

3. If you want to use remote program calls, yet specify the local node, give the parameter a value of -1 or the local node number.

PROGRAM-TO-PROGRAM CALLS

Program-To-Program (PTOP) calls provide a data exchange interface between two independent programs located at two different nodes. The two programs are referred to as the MASTER program and the SLAVE program. The MASTER program is always in control of the data exchange and initiates the activity between the two programs. The SLAVE program simply responds to requests from the MASTER program.

The following example illustrates a useful way to use Program-to-Program communication to implement an efficient buffered file transfer program.

```
FTN4,L
      PROGRAM FCPY (3,50), PTOP File Copy (Master Program)

C*********************************************************************
C This PTOP Master/Slave program pair provides a high speed file     *
C transfer capability, utilizing highly efficient buffered data      *
C transfers.                                                         *
C                                                                    *
C The data (records) of the source file are read into a large buffer*
C then transferred all at once with one PWRITE request across the    *
C communication link. This technique dramatically increases the      *
C link throughput over the normal RFA techniques of                  *
C one-record-at-a-time transfer.                                     *
C                                                                    *
C*********************************************************************
C   RU,FCPY, Source namr, Destination namr, Destination Node Number  *
C*********************************************************************

      IMPLICIT INTEGER(A-Z)

      DIMENSION DCB(144),BUFFER(1024),IPBUF(10),PCB(4),TAG(20)
      DATA BFSZ/1024/,DMY/0/
      LU=LOGLU(I)

C Get the Run String.  B reg will contain length of string on return.
      CALL EXEC (14,1,BUFFER,-80)
      CALL ABREG (I,LEN)

C Start parsing after "RU,FCPY,".
      PTR=9
```

```
C Get NAMR1 and NAMR2. NAMR2 goes to the Tag buffer for slave program.
      I=NAMR (IPBUF,BUFFER,LEN,PTR)
      I=NAMR (TAG,BUFFER,LEN,PTR)

C If parameter 2 is defaulted, copy IPBUF to TAG.
      IF(TAG .NE. 0)GOTO 17

      DO 15 J=1,10
      TAG(J)=IPBUF(J)
15    CONTINUE

C Get security code and cartridge reference number.
17    SC  =IPBUF(5)
      CR  =IPBUF(6)

C Open the source file.
      CALL OPEN (DCB,ERR,IPBUF,1,SC,CR)
      IF(ERR .GE. 0) GO TO 5
      WRITE(LU,100)ERR
100   FORMAT(/"OPEN ERROR ",I3)
      STOP 1

C Get the size and type of source file.
5     CALL LOCF (DCB,ERR,REC,DMY,DMY,SZSEC,DMY,TYPE)
      SIZE  =SZSEC/2
      TAG(7)=TYPE
      TAG(8)=SIZE

C Get node number.
      I=NAMR (IPBUF,BUFFER,LEN,PTR)
      NODE=IPBUF(1)

C Now schedule the slave, tag field contains destination file namr
      CALL POPEN(PCB,ERR,6HFCPYS ,NODE,TAG)
      IF(ERR .EQ. 0) GO TO 10
      WRITE(LU,110)ERR
110   FORMAT(/"POPEN ERROR "I3)
      STOP 2

C Initialize record count and buffer index
10    BUFFER(1)=0
      IX=3

C Now loop to read and block records from the file
C into the BUFSZ buffer.

20    CALL READF (DCB,ERR,BUFFER(IX),80,LEN)
      IF(ERR .GE. 0) GO TO 25
      WRITE(LU,120)ERR
120   FORMAT(/"READF ERROR",I3)
      STOP 3

C Advance record count kept in BUFFER (1).
C Store length of current record in BUFFER (IX-1).
C Advance IX to next record position.

25    BUFFER(1)=BUFFER(1)+1
      BUFFER(IX-1)=LEN
      IX=IX+LEN+1

C Now, if we haven't exceeded the buffer length or encountered
C an end-of-file, go back and read another record.
      IF (IX .LT. BFSZ-80 .AND. LEN .NE. -1) GO TO 20
```

```
C If it was an EOF, backup the record counter.
      IF (LEN .EQ. -1) BUFFER(1)=BUFFER(1)-1

C Write the buffer to the slave program.
      CALL PWRIT (PCB,ERR,BUFFER,IX,TAG)

C If we don't get an error or an EOF, go read some more records.
      IF(ERR .EQ. 0  .AND.  LEN .NE. -1) GO TO 10

C Was it error or EOF?

C If PWRIT error, print message and quit.  The user will have to
C purge the destination file and try again or make
C this program more sophisticated.  You might indicate to the
C slave an error occurred by setting the tag field to some value.
C This value could then cause the slave to purge the output file
C in response to the aborted transfer attempt.
      IF(ERR .NE. 0)WRITE(LU,130)ERR
130   FORMAT("/FCPY: PWRIT ERROR = "I3)

C We are now finished. Tell the slave we are finished
C by sending PCONT.
      CALL PCONT (PCB,ERR,TAG)

C Close the input file.
      CALL CLOSE (DCB)
      END


FTN4,L
      PROGRAM FCPYS (3,50), PTOP File Copy (Slave Program)
C*********************************************************************
C
C   This program receives data from the master program, FCPY,
C   deblocks the data in the incoming buffers and stores the data
C   in a file.
C
C*********************************************************************

C The following example program has minimum error checking and
C processing code to keep this example relatively simple. More
C useful error checking and error messages should be written for
C a production program.

      IMPLICIT INTEGER (A-Z)
      DIMENSION BUF(1024),TAG(20),P(5),DCB(144)

C Call RMPAR to get class number for class get issued
C   by (CALL GET).

      CALL RMPAR (P)

C I/O CLASS number passed in parameter 1.
      CLASS=P(1)

C Now sit and wait for some action.

      CALL GET (CLASS,ERR,FCN,TAG,LEN)
      IF (ERR .NE. 0) STOP 1

C FCN= 1 POPEN
C      2 PREAD
C      3 PWRIT
C      4 PCONT
```

```
C On POPEN, the tag field contains the destination file NAMR.

      GO TO (10,25,25,25)FCN

C Create the destination file.

C Information in the tag field is the result of a call to NAMR
C by the Master program. The routine NAMR and its output is
C described in the Relocatable Library Manual for your system.

C                        Name, Len  , Type , SC   , CRN
10    CALL CREAT(DCB,ERR,TAG,TAG(8),TAG(7),TAG(5),TAG(6))
      IF(ERR .GE. 0) GO TO 30

C Process error and send to master program

25    CALL REJCT (TAG,ERR)
      CALL FINIS
      STOP 3

30    CALL ACEPT (TAG,ERR,BUF)

C   At this point we don't want to accept any more POPEN's
C   from any other master programs until we complete this transfer.

40    CALL GET (CLASS,ERR,FCN,TAG,LEN)
      GO TO (50,50,60,70)FCN

C Reject all POPEN attempts until we exit.

50    CALL REJCT (TAG,ERR)
      GO TO 40

C Process PWRIT.

60    CALL ACEPT (TAG,ERR,BUF)

C Unpack the buffer and WRITE records to destination file.
C BUF(1) is the first word in the buffer and equals the
C number of records in the buffer. LEN is the second word
C in the buffer and equals the record length. IX is set to
C 3 to begin reading the third word in the buffer. After a
C record is read, IX is reset to equal its original value plus
C the length of the record just read, plus 1 to pass the word
C containing the length of the next record.

      IX=3
      DO 65 I=1,BUF(1)
      LEN= BUF (IX-1)
      CALL WRITF (DCB,ERR,BUF(IX),LEN)

65    IX=IX+LEN+1

C Go back to the GET statement and wait for the next buffer.

      GO TO 40

C Handle PCONT and close file when all buffers have been passed.

70    CALL ACEPT (TAG,ERR,BUF)
      CALL CLOSE (DCB)
      CALL FINIS
      END
```

UTILITY ROUTINES

Utility routines provide capabilities not available through conventional program communication calls. Some of these routines consist of the following:

DMESG and **DMESS** — send a message to a remote node's system console or message processor.

FCOPY — copy a file from one node to another.

FLOAD — down-load an absolute program file into an RTE-M or RTE-L node.

GNODE— obtain your local system's node number.

DNODE— access peripheral devices at a remote node.

DLGON— log-on to a remote session account.

DLGOF — log-off a remote session account.

DLGNS — obtain non-session access to a session-monitor node.

You have seen three of these calls used in this manual. When you scheduled the editor with the REMAT command RW,EDITR and edited the file SOURCE, you entered the utility call, DNODE. This utility allowed you to run the program remotely from REMAT and have all the FORTRAN READ and WRITE statements access your local terminal. The other utility calls, DLGON and DLGOF, were used in the remote file access program PROG3 which allowed you to programmatically log-on and log-off a remote session account.

REMOTE I/O MAPPING

Remote I/O Mapping is an optional DS/1000-IV feature providing the capability for I/O requests made to a certain LU (designated as being a mappable LU) at one HP 1000 system to be automatically mapped into I/O requests on LUs on another HP 1000 system.

This consequently allows any input, output or control request directed to an LU at a local node to be re-directed (mapped) to a specific LU pointing to a unit record device on a remote node.

With this feature you could do any one of the following:

- With some applications, HP 1000 systems are placed in harsh environments for the purpose of collecting data on environmental changes and transmitting this data across DS/1000-IV communication lines to another computer system.

- In this type of environment, it is undesirable, for many reasons, to have terminal consoles directly connected to the computer system. With Remote I/O Mapping, however, you can have "console-less" nodes. Once Remote I/O Mapping is established, all console operations may be performed from a remote node. Using the same DS/1000-IV communication link used for transmitting data, you may communicate with this console-less computer system from a console at a remote node located in an entirely different area.

- To expand more on the above application, you may have numerous console-less systems communicating with a console located at a central remote node. To aid in this application, Remote I/O Mapping has an option whereby it will place at the beginning of any message sent to the central console a header identifying the node location where the message originated from. This will identify the node location of any messages received by the central console.

- Remote I/O Mapping may be used to create a pseudo "virtual terminal" to any HP 1000 node in your network supporting Remote I/O Mapping. Once Remote I/O Mapping has been established with a particular remote terminal LU, whether it actually has a terminal or not, you may then log on to an existing remote account and enter commands just as if you were working from a terminal directly connected to the system. This will permit access to interactive programs at remote nodes without requiring that they be modified to issue DEXEC calls for interactive I/O.

Basically, Remote I/O Mapping is very useful in applications where one node requires a unit record device located at another node. This may apply to situations requiring "console-less nodes", virtual terminal capability, or remote printer devices. There are many possibilities.

When working with Remote I/O Mapping, it is important to supply the correct LU references when re-directing one LU to another LU on a different node. Supplying the wrong LU numbers may cause some confusing responses. For this reason, the operation of Remote I/O Mapping will not be discussed here since it is designed for the more advanced user. Refer to your DS/1000 Network Manager's Manual for a more detailed discussion of Remote I/O Mapping.

REVIEW

At this point in the manual you have learned how to use many of the valuable interactive features of DS/1000-IV. With DS/1000-IV's capabilities, you are now able to access resources that exist in computer systems other than your own.

As a review, let us apply the remote program development concepts learned in this chapter to develop programs from an RTE-M node. Let us assume that node 12 in our sample network configuration is an RTE Memory Based computer. Such a system may not have a file system and is therefore very limited in developing programs that you wish to run on your RTE-M system. With DS/1000-IV, however, you are able to use many DS/1000-IV features to access program development tools existing in other computer systems in your network.

From the RTE-M node you would first run REMAT (see Figure 7), issuing the correct parameters so that a communication link is created with a node containing an editor (node 13 in our example). You would then schedule with wait (RW) the editor located at the remote node. From your local node, you may perform all the editing steps needed in modifying or creating a program file.

When finished editing your program, you may now remotely compile and load the program. The first step is to use REMAT and remotely run with wait (RW) the compiler. Make sure you enter a list file namr as one of the parameters. This will insure a list of the compiled program with any errors that may have occurred.

Once the program has compiled successfully, run with wait the DS version of the RTE-M loader (RTMLG). Enter as the scheduling parameter a command file comprised of the file

name containing the snapshot of your RTE-M system (this file, created during system generation, describes the RTE-M system to the relocating loader), a list of libraries the loader is to search, and any other additional information the loader requires.

When the loader has successfully completed with no errors, a type 7 file will exist containing the absolute version of your program to be executed on the RTE-M system.

To run this program on your local RTE-M system, you need to down load the type 7 file into memory of the RTE-M. While in REMAT, enter the LOad command and load the absolute program developed on node 13 into memory of your local RTE-M system. Your program may now be executed on the RTE-M.

This process simplifies developing programs to be executed on an RTE-M computer system. This process need not be limited to just basic program development but may be expanded to include development of system generation files.

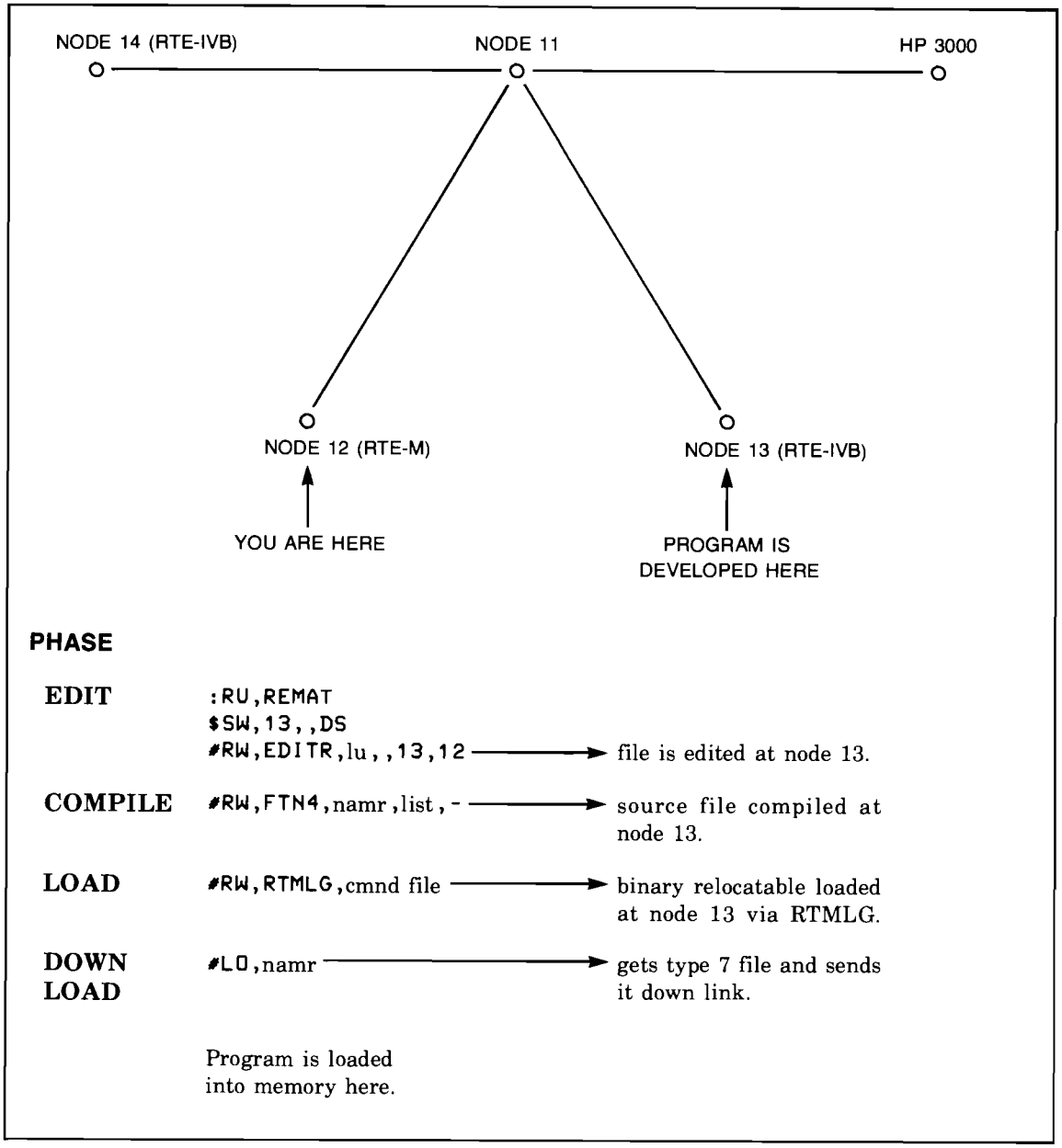


Figure 7. Remote Program Development From An RTE-M

Chapter 4
DSINF

When a DS/1000-IV network has been initialized, a table is created showing how different nodes in a network are connected to one another. DSINF is a program that displays this table and a variety of other information about your network. This table is extremely useful in finding out how your network is designed. This table contains information such as the LU number through which each node in your network communicates with one another, their time-out values and whether the node is running under DS/1000-IV or an older version of DS/1000.

```
:RU,DSINF <-------------------- Enter this command.

/DSInn: FUNCTION? <------------ DSINF will prompt you with this. (nn is your session
                                number).
```

In response to this prompt, type NR to receive a listing of the Nodal Routing Vector (NRV) presently in your system.

```
/DSInn: FUNCTION? NR

NRV SPECIFICATIONS:
LOCAL NODE :     11, NO. OF NODES=      4

     NODE    LU   EQT SUB  T/O(SEC)   TYPE  LEVEL
      11*     0                 15              1
      12*    12    10   1        15     65      1
      13*    13    12             15     66      1
      14*    15    13   1        15     65      1

      (* INDICATES NEIGHBOR)
```

This table lists the HP 1000 nodes your local node is able to communicate with, the LU of each communication link, its EQT and subchannel number, the transaction time-out override value, the type of driver being used and the upgrade level of each node.

As illustrated in the above example, node 11 uses LU 12 as its communication link with node 12, LU 13 as its communication link with node 13 and LU 15 as its communication link with node 14.

The master Time-Out override value is the maximum time your local system will allow a transaction to complete to another node.

If the LEVEL of a node is 1, then the node is operating under DS/1000-IV software. If the LEVEL of a node is 0 then the node contains an older version of DS/1000. An asterisk (*) indicates the node is a neighbor of your local node.

As you recall, the example network has five nodes, four RTE nodes and one HP 3000 node. The HP 3000 node is not included in the NRV table. You can use the VA (DS values) command to determine the LU number used to communicate with the HP 3000. In the following example network, node 11 uses LU 14 for communication with the HP 3000.

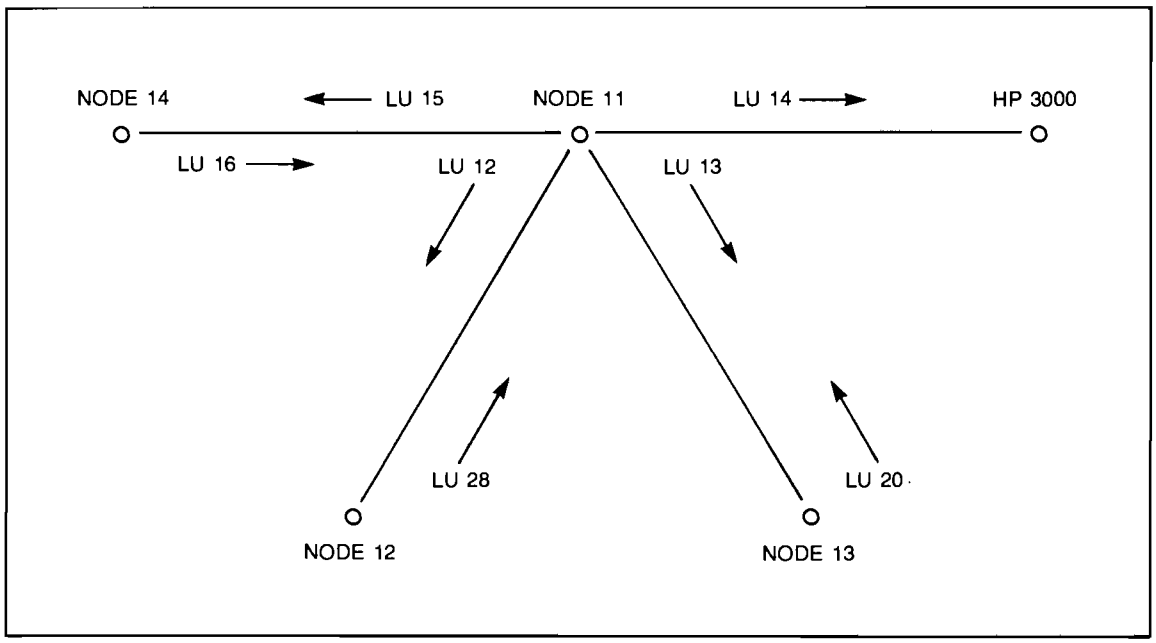


Figure 8. Using Logical Unit Numbers

You may also run DSINF remotely from REMAT to access network information from other nodes in the network. If you wanted to run DSINF at node 12, you would use the following sequence of commands:

```
:RU,REMAT

$SW,12,,DS

#RW,DSINF

/DSInn:  FUNCTION ?
```

DSINF will send you this prompt from node 12. To list the NRV table located at node 12, enter an NR command.

```
NRV SPECIFICATIONS:
LOCAL NODE :      12, NO. OF NODES=       4

     NODE     LU    EQT  SUB   T/O(SEC)    TYPE   LEVEL
      11*     28      9    1        15      65       1
      12       0                    15               1
      13      28      9             15      65       1
      14      28      9    1        15      65       1

     (* INDICATES NEIGHBOR)
```

These two examples of an NRV Table describe the nodes a particular local node knows about (excluding the HP 3000). Therefore, at different nodes the tables may be different. In the two examples shown, the LU numbers, the time-out values, the level of each node, and whether a node is a neighbor are different for the two nodes. To complete the picture as shown in Figure 8, you would have to run DSINF remotely at node 13 and 14. This would tell you that node 13 uses LU 20 as its communication link with nodes 11, 12 and 14, and that node 14 uses LU 16 as its communication link with nodes 11, 12 and 13. To find more on what the program DSINF is able to do, enter two question marks (??) after the DSInn prompt.

```
/DSInn: FUNCTION? ??

 /DSInn: 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
   EQ          DS/1000 EQT ENTRIES
   EQ,N        DS/1000 EQT ENTRY # N
   LU,N        EQT ENTRY FOR LU # N
   MA          MESSAGE ACCOUNTING
   RR          REROUTING
   RS          REMOTE SESSIONS
   EX OR /E    TERMINATE DSInn
```

To find additional information on these commands and what they can do, refer to the DS/1000-IV Network Manager's Manual.

```
/DSInn: FUNCTION? EX <-------- EX ends DSINF execution.

 *** END OF DSInn ***
```

Chapter 5
1000 To 3000 Network

MPE

Before beginning this section you should be familiar with an HP 3000 system. Refer to the appropriate introductory manuals for the HP 3000 or the Documentation Map at the beginning of this manual for a list of HP 3000 manuals that will prove useful in understanding communication with the HP 3000.

A 1000 to 3000 communication link is very similar in operation to a 1000 to 1000 communication link. The major difference is that once communication has been established between an HP 1000 system and an HP 3000 system, the terminal at the HP 1000 end of the link acts as a virtual terminal with the HP 3000 system. This will allow any program to be scheduled on the HP 3000 and interact directly with your terminal on the HP 1000 without including any DEXEC calls or Remote I/O Mapping.

RMOTE

To communicate with an HP 3000, use the program RMOTE. The only limitation is that RMOTE must run in an RTE node directly connected (a neighbor) to the HP 3000 node. If Remote I/O Mapping is used, however, you may be located elsewhere in the network. In our example, node 11 is a neighboring node to the HP 3000 (see Figure 9). Thus, an operator at node 11 uses the program RMOTE to send commands to the HP 3000.

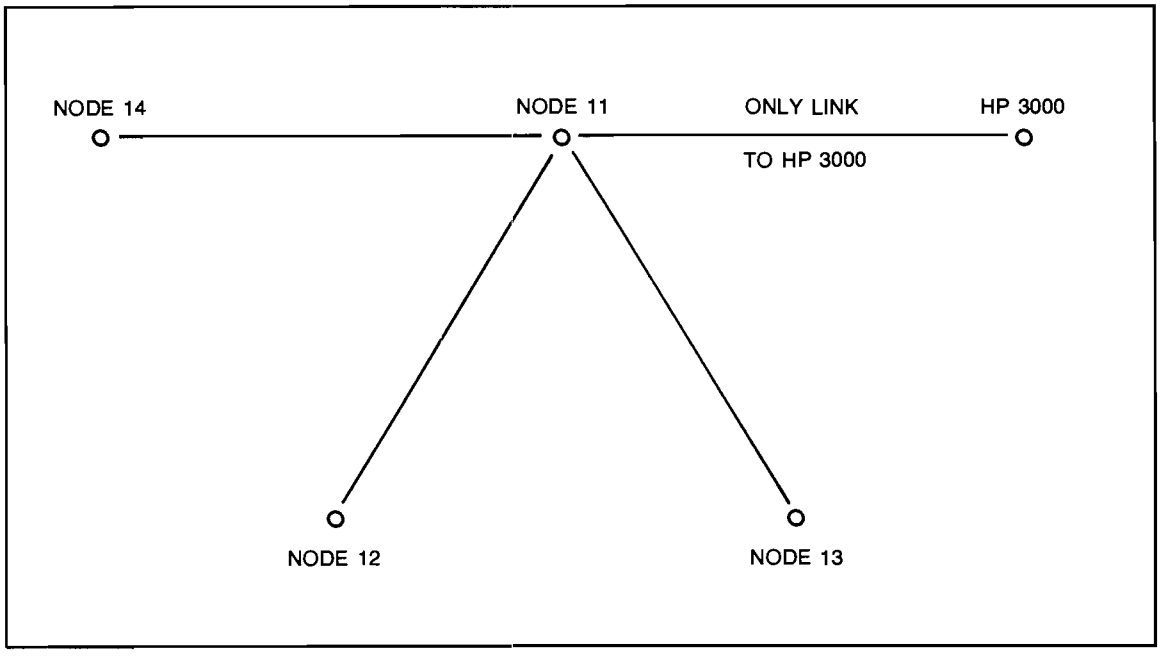


Figure 9. Using the HP 3000 Remotely.

For the following examples, let us assume you are at an RTE session terminal and are running under FMGR. You should have a colon (:) as a prompt.

```
:RU,RMOTE <------------------ Enter this command.

$ <-------------------------- RMOTE will prompt you with a dollar sign ($).
```

At this point, the commands you enter will be sent to your local operating system because you have a dollar sign as a prompt.

```
$TI <------------------------ Enter this command.

1980  11  10  4  30 <-------- You will receive this response.
```

To communicate with the HP 3000 you need to enter the Switch command.

```
$SW,[LU] <-------------------- LU is the Logical Unit Number of an HP 3000 link.
```

If you are not a neighboring node to the HP 3000 or the 3000 link has not been initialized you will receive the following message:

```
NEED TO RUN "DINIT" <--------- If you receive this message, contact your Network
                               Manager.
```

If the 1000 to 3000 communication link has been initialized, you will receive a pound sign (#) as your new prompt.

```
# <------------------------- This prompt is to remind you that you are now com-
                             municating with an HP 3000.
```

Figure 10 diagrams where you are and where commands are being sent to. The first command you must issue is the HELLO command. This signs you on to an account on the HP 3000.

```
#HELLO USER ACCNT <---------- Enter this command. The USER.ACCNT is an
                             example of a user and account name allowing you to
                             sign on to the HP 3000. If you don't know your
                             account name, obtain one from your System
                             Manager.
```

If you receive the following message, your account does not exist on the HP 3000. If just the third line is printed, the HP 1000 to HP 3000 communication link is "DOWN". In either case contact your System Manager.

```
 HELLO USER ACCNT
NON-EXISTENT ACCOUNT.  (CIERR 1437)
HELLO FAILED OR LINE DOWN
```

When the communication link is "UP" and you issued the proper user and account name, you will receive a message similar to:

```
HP3000 / MPE III B.01.00.  MON, JAN 14, 1980,  3:26 PM
```

You may now enter commands to the HP 3000.

```
#SHOWTIME <------------------ Enter this command.

MON, JAN 14, 1980, 3:26 PM <- You will receive a response similar to this.
```

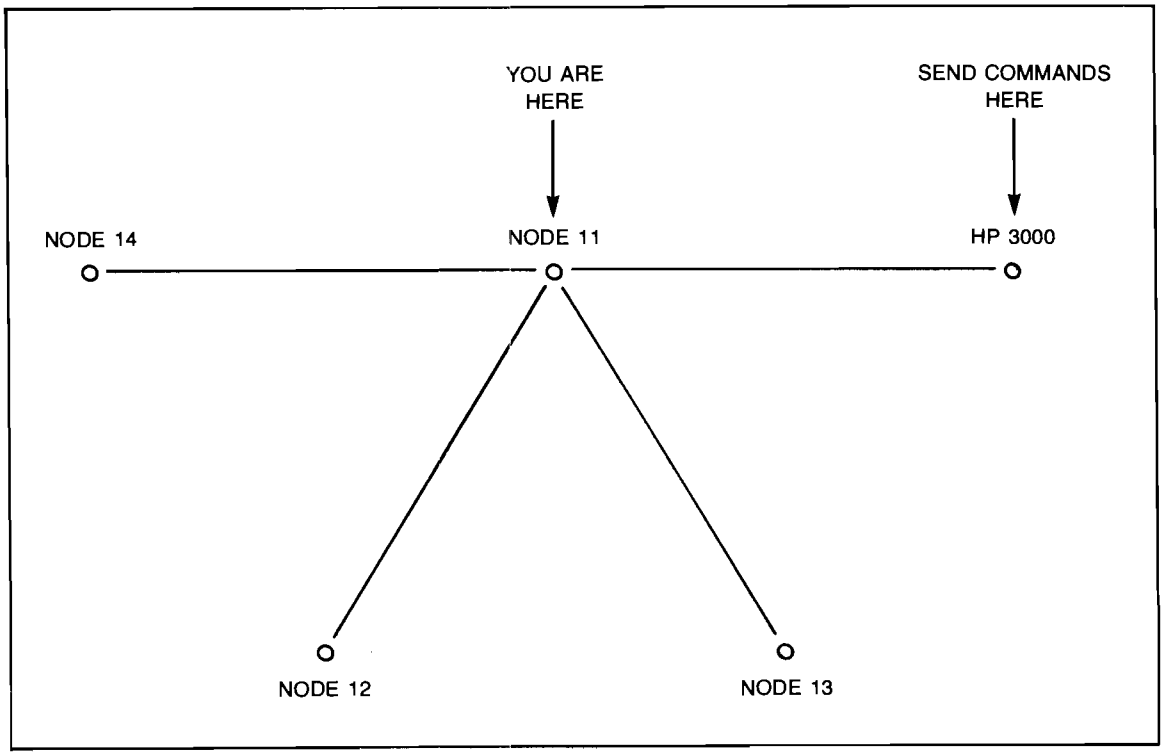


Figure 10. Using RMOTE

To see a list of files in your account, try the following:

```
:LISTF,2
```

Enter this command. The parameter 2 allows additional details to be listed.

Account name

Group name

```
ACCOUNT= ACCNT      GROUP= GRP

FILENAME  CODE  -----------LOGICAL RECORD----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS  X MX

APPDXA             88B  FA         258        258  23      104 13 13
APPDXB             88B  FA         352        352  23      136  9  9
APPDXC             80B  FA         173        173   3       59  1  1
APPDXD             80B  FA         408        408  16      135 14 14
GRAPDX1            88B  FA         451        451  23      168 11 11
GRAPDX2            88B  FA         226        226  23       88 11 11
GRAPDX3            88B  FA         435        435  23      160 10 10
GRAPDX4            88B  FA         616        616  23      224 14 14
GRAPH1             88B  FA         697        697  23      256 16 16
GRAPH2             88B  FA        1245       1245  23      448 14 14
GRAPH2A            88B  FA        1976       1976  23      696 15 15
```

End-of-file location

ASCII/Binary option

Size of records

Record format (Fixed length)

The GROUP name shown above is a name the HP 3000 associates with your user name to separate different files within an account into various groups. The list of information under SECTORS identifies the number of sectors in use, extents allocated and the maximum number of extents allowed for a particular file.

Suppose you want to send a system command to your local RTE system without exiting the HP 3000 account and then store a file from your local RTE system to the HP 3000.

```
#SW <-------------------------- Enter this command to SWitch back to the local RTE
                                system

$ <---------------------------- The $ prompt indicates you are back at the local
                                system. Your session on the HP 3000, however, is
                                still active.

$TI <-------------------------- Send this command to the local RTE system.

$1980  16  15  26  47 <-------- You will receive a response similar to this.
```

To store a file from your local RTE system to the HP 3000 you enter the MOve command (this command is an option to RMOTE, and therefore may not be included in your system). Since you are now communicating with your local RTE system, indicated by the dollar sign ($) prompt, the direction of your file transfer will be from your local RTE system to the HP 3000.

```
$MO,URFILE,URFILE,UN <--------- Enter this command. The first file name is the name
                                of the file on your local RTE system that you wish to
                                move. The second file name is what you wish this file
                                to be called in your account on the HP 3000. The
                                parameter UN causes the file to be transferred as
                                fixed length 40 word ASCII records instead of the
                                default as variable binary records.

NO SLAVE AT 3000 <------------- If you receive this error when you attempt a move,
                                contact your System Manager. This error indicates a
                                slave program required for the MO command does
                                not exist on the 3000.

13 RECORDS READ, 13 RECORDS WRITTEN

                                You will receive this response indicating how many
                                records were successfully transferred.
```

Whenever you use the MO command, you must first sign on to an account on the HP 3000 with the HELLO command. This of course insures your files are moved to the correct account on the 3000.

```
$SW <-------------------------- Resume sending commands to the HP 3000.

# <---------------------------- You will again receive a pound sign (#) as your
                                prompt without having to enter the HELLO com-
                                mand again.

#LISTF,2 <---------------------- Enter this command to confirm your file was
                                transferred.
```

```
ACCOUNT= ACCNT       GROUP= GRP

FILENAME  CODE  ----------LOGICAL RECORD----------  ----SPACE----
                  SIZE  TYP        EOF      LIMIT R/B  SECTORS  X MX

APPDXA             88B  FA         258        258  23      104 13 13
APPDXB             88B  FA         352        352  23      136  9  9
APPDXC             88B  FA         181        181  14       70  1  1
APPDXD             80B  FA         110        110   3       38  1  1
GRAPDX1            88B  FA         451        451  23      168 11 11
GRAPDX2            88B  FA         517        517  23      192 12 12
GRAPDX3            88B  FA         412        412  23      152 10 10
GRAPDX4            88B  FA         572        572  23      208 13 13
GRAPH1             88B  FA         697        697  23      256 16 16
GRAPH2             88B  FA        1245       1245  23      448 14 14
URFILE             80B  FA          13         13   3        6  1  1
```

Transferring files from the HP 3000 to the HP 1000 is very similar to transferring files in the opposite direction. The same parameters apply in either case. The first parameter is the file you wish to transfer, the second parameter is the name you wish to give the transferred file and the third parameter should, in many cases, be an UN. The direction of the transfer depends on what system you presently have RMOTE sending commands to (the "$" prompt indicates commands are being sent to the HP 1000; the "#" prompt indicates commands are being sent to the HP 3000). For more information on the MOve command, refer to your User's Manual.

Another effective way to use the MO command is to list files located on the 3000 to your terminal. To do this, enter the following command from RMOTE:

```
#MO,filename,1
```

where filename is the file you wish to list. This feature eliminates the time consuming process of using the HP 3000 editor, texting a file, and then listing the file to your terminal.

If at any time you wish to see what jobs and sessions are running on the HP 3000 try the following command:

```
#SHOWJOB <---------- Enter this command.

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

S1      EXEC        20  20       SAT 11:14A  OPERATOR.SYS
S4      EXEC        44  44       MON  7:33A  OC1,COSOPER.INFOSYS
S5      EXEC        26  26       MON  7:39A  OS3,COSOPER.INFOSYS
S27     EXEC        32  32       MON  9:37A  SOSUSER.STATS
J62     EXEC         6S LP       MON  9:25A  COS5,COSRPT.INFOSYS
S10     EXEC        27  27       MON  8:01A  OS1,COSOPER.INFOSYS
S11     EXEC        24  24       MON  8:03A  MGR.MRN
S28     EXEC        22  22       MON  9:42A  STODG.INFOSYS
S16     EXEC        42  42       MON  8:27A  JANE.MRN
S20     EXEC        28  28       MON  9:07A  OS2,COSOPER.INFOSYS
S29     EXEC        66  66       MON  9:46A  USER.ACCNT
J63     WAIT:1   8   6S LP       MON  9:41A  DIST,USRAMIGO.MAILS

12 JOBS:
    0 INTRO
    1 WAIT; INCL 0 DEFERRED
   11 EXEC; INCL 10 SESSIONS
    0 SUSP
JOBFENCE= 7; JLIMIT= 1; SLIMIT= 32
```

To exit from RMOTE, the only command you need to enter is EX. RMOTE will issue a BYE command for you, terminating the session at the 3000 while, at the same time, ending RMOTE's execution.

```
#EX <-------------------- Enter this command.

CPU=1. CONNECT=1. THU, JAN 17, 1980, 10:51 AM
END RMOTE
```

EDITOR

To run the MPE editor, run RMOTE, enter the SW command, enter the HELLO command and then type EDITOR. No node number is needed since only a neighboring node can communicate with an HP 3000 and no special run parameters need to be given since your terminal acts as a virtual terminal to the HP 3000.

```
:RU,RMOTE <-------------- Enter this command.

$SW <-------------------- SWitch so you can now send commands to the
                          HP 3000.

#HELLO USER ACCNT <------- Give the HELLO command with your correct ac-
                          count name.

#EDITOR <---------------- This command schedules the editor at the HP 3000.
```

You will receive a response similar to the following indicating you are running the HP 3000 Editor:

```
HP32201A.7.05 EDIT/3000  THU, JAN 17, 1980,  3:01 PM
(C) HEWLETT-PACKARD CO. 1979
```

The HP 3000 editor does not ask you for a source file. If you have a file that you want to edit, the first command to give is:

```
TEXT filename
```

If you want to create a new file, enter the command ADD and then enter your lines of text.

```
/ADD <------------------- The / is the editor's prompt. We will create a new
                          file.

1      $CONTROL USLINIT <- The editor gives the line numbers

2            DO 10 I=1,3
3            B=I
4            A=SIN(B)+COS(B)
5            WRITE(6,15) A
6         15 FORMAT ("VALUE OF A = ",F5.2)
7         10 CONTINUE
8            STOP
9            END

10     // <---------------- When you are done adding lines enter two slashes.

... <-------------------- Three dots indicate you are out of the ADD mode of
                          the editor.
```

```
/LIST ALL <-------------------- This command will list the complete file.

   1     $CONTROL USLINIT
   2           DO 10 I=1,3
   3           B=I
   4           A=SIN(B)+COS(B)
   5           WRITE(6,15) A
   6        15 FORMAT ("VALUE OF A = ",F5.2)
   7        10 CONTINUE
   8           STOP
   9           END

/KEEP FILE4 <---------------- Create a file called FILE4 and keep the new text.

/EXIT <---------------------- Exit the editor

END OF SUBSYSTEM <----------- This is an HP 3000 message telling you that you are
                              no longer in the editor.

#EXIT <---------------------- Exit RMOTE.
```

When you give the EXit command, RMOTE will automatically give a BYE command for you.

```
CPU=3. CONNECT=10. THU, JAN 17, 1980,  3:10 PM

END RMOTE
```

PROGRAM DEVELOPMENT

Program development on an HP 3000 is very similar to program development on an RTE system. You write the program with the editor, compile, prepare and then run the program. There is no loader as with the RTE system. The only difference in developing a program on an HP 3000 rather than on an RTE system is that you do not have to run the editor, the compiler or the program with any special parameters other than what they normally require. Your terminal acts as a virtual terminal to the HP 3000. It's as if your terminal is directly connected with the HP 3000 rather than going through a DS communication link.

The steps in HP 3000 program development are the same whether you are directly connected to the HP 3000 or using RMOTE at an RTE system to talk with the HP 3000.

```
:RU,RMOTE <------------------ Run RMOTE.

$SW <------------------------ Give the SWitch command.

#HELLO ACCNTS.USER <--------- Enter the HELLO command with your correct ac-
                              count name.

HP3000 / MPE III B.01.00.  MON, JAN 21, 1980, 10:26 AM

#EDITOR <-------------------- Schedule the editor

HP32201A.7.05 EDIT/3000  MON, JAN 21, 1980, 10:27 AM
(C) HEWLETT-PACKARD CO. 1979

/ADD <----------------------- Create a new file. (This file is actually the same as
                              FILE4 created in the Editor section of this chapter.)
```

```
    1      $CONTROL USLINIT
    2            DO 10 I=1,3
    3            B=I
    4            A=SIN(B)+COS(B)
    5            WRITE(6,15) A
    6         15 FORMAT ("VALUE OF A = ",F5.2)
    7         10 CONTINUE
    8            STOP
    9            END
   10      //
...
```

Line 5 in the above program contains a WRITE statement that writes to unit number 6. Unit numbers in an HP 3000 are allocated so that programs executed from a terminal use unit 5 as the terminal keyboard and unit 6 as the terminal's output device (display or terminal). Therefore, you read from unit 5 and write to unit 6.

If necessary, you may edit this file. When you are satisfied with it, save the program in a disk file with the KEEP command, and then terminate the editor.

```
/KEEP TRY1

/EXIT

END OF SUBSYSTEM
```

COMPILE AND EXECUTE THE PROGRAM

To compile and execute a program on an HP 3000 you only need to enter one command. The command FORTGO followed by the file name of your program (TRY1) will first compile your program and then, if your program compiled without any errors, run your program.

```
#FORTGO TRY1 <----------- Enter this command. Remember the # is your
                          prompt on the HP 3000.
```

Each line of the source program is printed as it is compiled.

```
PAGE 0001   HP32102B.01.02  (C) HEWLETT-PACKARD CO. 1979

00001000  $CONTROL USLINIT
00002000        DO 10 I=1,3
00003000        B=I
00004000        A=SIN(B)+COS(B)
00005000        WRITE(6,15) A
00006000     15 FORMAT ("VALUE OF A = ",F5.2)
00007000     10 CONTINUE
00008000        STOP
00009000        END

PROGRAM UNIT MAIN' COMPILED

****       GLOBAL STATISTICS       ****
****   NO ERRORS,   NO WARNINGS    ****
TOTAL COMPILATION TIME   0:00:01
TOTAL ELAPSED TIME       0:00:03
```

If an error occurs when compiling or running your program, run the editor again and list this file. Make sure your version of the program matches the example listed above and make any corrections if needed.

```
END OF COMPILE <-------- The compiler has successfully completed.

END OF PREPARE <-------- The program is ready to run.

VALUE OF A =  1.38 <---- The results of executing the program.
VALUE OF A =   .49
VALUE OF A =  -.85

END OF PROGRAM <-------- The program has completed execution.
```

SAVING THE EXECUTABLE PROGRAM

After executing the FORTGO command, the executable version of the program ("prepared" program) will be in a temporary system file called $OLDPASS. If you log off of the HP 3000, this file will be removed. To save this file permanently in your account, enter the following:

```
#SAVE $OLDPASS,NEWNAME
       ↑        ↑
   temporary  permanent
```

The file name of the permanent file is an arbitrary name. We used NEWNAME in this example. Like all system file names on the HP 3000, a file must begin with an alphabetic letter, must be no longer than eight characters long and must not already exist in the account you are logged on under.

To run this program later, all you need to do is the following:

```
:RU,RMOTE

$SW

#HELLO ACCNTS.USER

#RUN NEWNAME

VALUE OF A =  1.38
VALUE OF A =   .49
VALUE OF A =  -.85

END OF PROGRAM
```

The command FORTGO can be broken into several steps using other commands to give you more control of the program development process. By entering one command you can first compile a program and then by entering another command, you can run the program. These steps are not shown here but are discussed fully in the HP 3000 FORTRAN Reference Manual.

HP 3000 PROGRAM CALLS

HP 3000 Program Calls may be divided into three groups:

1. HP 3000 Remote File Access Calls
2. HP 3000 Utility Calls
3. HP 3000 Program-To-Program Calls

HP 3000 REMOTE FILE ACCESS

You can issue calls from your RTE application program to a set of HP 3000 Remote File Access intrinsics using a set of DS/1000-IV intrinsic calls. These calls, which are similar to the HP 3000 File System (FS/3000) calls, provide you with HP 3000 file access from an HP 1000 node. The names of the HP 1000 calls vary slightly, but all the parameters are the same. For instance, the DS/1000-IV call FRNAM, which renames a file, is equivalent to the FS/3000 call FRENAME.

DS/1000-IV intrinsic call –

CALL FRNAM(filenum,newfilereference)

FS/3000 intrinsic call –

FRENAME(filenum,newfilereference);

HP 3000 UTILITY CALLS

Utility calls are available which allow you to utilize the resources at an HP 3000 node from a program running at a neighboring RTE node. These utility calls provide you with capabilities that are not available through conventional program communication calls. The HELLO utility call, for instance, initiates an HP 3000 session and must be given before any other call to the HP 3000. The BYE utility call is used to terminate the session.

HP 3000 PROGRAM-TO-PROGRAM CALLS

PTOP calls are used between two programs, one on the HP 3000 and one at a neighboring RTE node. These calls provide a data exchange interface between two programs. The two programs are referred to as the MASTER program and the SLAVE program. The SLAVE program simply responds to requests given by the MASTER program which is always in control of the data exchange.

These calls are similar to the RTE-to-RTE PTOP calls.

HP 1000-TO-HP 3000 PROGRAM APPLICATIONS

Many applications exist for developing programs using DS/1000-IV to access an HP 3000 system. One of them may be the following hypothetical case.

The HP 1000 is ideally suited for performing real time control operations necessary in a manufacturing process control environment. The HP 3000, however, is better suited for processing the kind of management information needed in a manufacturing environment. Often, information needed in the 3000 database to provide the management reports is generated on the HP 1000 when performing the real time control operations and data acquisitions.

DS/1000-IV provides a solution to this problem by providing the communication link between the HP 1000 and HP 3000 computers. With DS/1000-IV, programs on the HP 3000 can communicate with programs on the HP 1000 and vice versa.

The data gathered by process control programs in the HP 1000 can be made available to programs in the HP 3000 via Remote File Access calls (RFA) to the HP 3000. This method of data transfer between the two computers is fine for low volume transfers.

For a more efficient means of data transfer for higher volumes of data, program-to-program communication calls will prove more useful. With this feature, the program in the HP 1000 can gather large buffers of data, unformatted, and initiate a PTOP transfer to a program on the HP 3000. The data is transferred in its most compact form with the greatest efficiency and speed. The program in the HP 3000 can unpack the buffer and store the data either in a database or in files, making it available for management or engineering reports where necessary.

Chapter 6
DS/1000-IV ERROR CODES AND PROGRAM STATUS

DS/1000-IV ERROR CODES

While performing some of the examples in this manual and trying some other DS/1000-IV features available, you may occasionally receive an error. Since DS/1000-IV utilizes many different programs, many types of errors exist.

You may receive a different error in each one of these cases:

- Issuing REMAT or RMOTE commands.
- Issuing System Commands from REMAT or RMOTE.
- Scheduling a local or remote program from REMAT or RMOTE.

You may also receive:

- RTE IO, SC and FM errors.
- Errors generated from running the LOADER or COMPILER.
- Numeric errors returned in the IERR parameter within a program using DS calls.

The DS/1000-IV User's Manual lists most of these errors and describes how to recover from them. In most cases the recovery is simple, for instance a -10 error indicates a missing or illegal parameter was issued. However, some of the recovery procedures are beyond the normal user's capability and your Network Manager should be contacted in these instances.

If the error you received is not in the DS/1000-IV User's Manual you should check either the Operating Manuals or Language Reference Manuals for your system. Most likely you will be able to determine what kind of error you received and whether it was generated from DS/1000-IV, the Operating System or from developing and running a program from DS.

DETERMINING PROGRAM STATUS

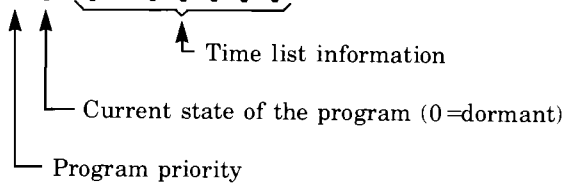
There will be times while running a program locally or remotely from DS/1000-IV when the action you expect to take place doesn't, and you receive no error message to explain why. In these cases the problem may be that the System may be busy and has not executed your program as quickly as expected, a device such as a printer may be down or you may simply be expecting the wrong action to take place.

In any case, there are a few ways to determine the status of your program. One of these ways is the RTE STatus command.

If your program is running locally, you may issue the ST command directly from FMGR.

```
:SYST,PROG2

99 0  0   0 0 0 0 0
```



If you scheduled the program at a remote node, you use this sequence of commands:

```
:RU,REMAT

$SW,12,,DS <-------- The number 12 is the number of the node at which
                     the program is located.

#ST,PROG2

99 0 0  0 0 0 0 0
```

The second parameter (0) indicates that PROG2 is dormant.

A program could be in any one of the following states:

0 = Dormant, not scheduled

If it is your own program that is dormant you may want to check the source code to make sure there are no errors. If, while the program is running, it makes an abortive error, RTE will terminate it abnormally and it will become dormant.

1 = Scheduled but not necessarily executing yet

If you feel something is not working properly, terminate the program and try scheduling it again making sure all parameters are accurate and all devices it uses are operating and ready. Otherwise, wait a little longer and see if any action begins to take place.

2 = Suspended, waiting for I/O to complete

If your program is in IO suspend, check the device to see if it is operating and ready for use. Sometimes you will have to wait for another program to finish using the device before your program gets access to it.

3 = General Wait

When your program is in this state, you can either wait or make the resources the program is waiting for available (you may use WHZAT, which is described next, to determine which resources). User programs will be in this state while waiting for a DS reply. REMAT will be in this state when an RW command is given and the "son" program is successfully scheduled.

4 = Suspended, waiting for memory

You should consult with your Network Manager if your program is in this state.

5 = Suspended, waiting for disk space

You should consult with your Network Manager.

6 = Operator or program suspended

If your program was suspended by an operator or a program, type the GO command to continue execution.

If your program is in states 1,2,or 3, you will find it helpful to use the WHZAT utility.

WHZAT

WHZAT describes the current system environment. It can be used to display:

- All scheduled and suspended programs and their status.
- The status of all partitions in numeric sequence.
- Only those programs associated with your session (default mode).

WHZAT may be locally run either by entering

```
:RU,WHZAT,lu,option
```

or just by entering

```
:WH
```

You can also run WHZAT at a remote RTE-IVB node and have the output listed at your terminal. To do this you need to enter the following commands:

```
:RU,REMAT

$SW,remote node,,security code

#RW,WHZAT,,AL,your node
```

Remember, any program scheduled from REMAT using the RW command has a 20 minute time limit to complete.

For more information on WHZAT refer to the appropriate System Operating Manual for your system.

READER COMMENT SHEET

Getting Started with DS/1000-IV

91750-90004 **April 1982**

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. 141 CUPERTINO, CA.

— POSTAGE WILL BE PAID BY —

Hewlett-Packard Company
19420 Homestead Road
Cupertino, California 95014

FOLD FOLD