SPERRY UNIVAC
COMPUTER SYSTEMS

ATTN: CHARLIE GIBBS

01110
CAV208M45541 UP 8811 R1

UAS

SPERRY UNIVAC
1 - 1818 CORNWALL STREET
VANCOUVER B C
V6J 1C7

PUBLICATIONS REVISION

Operating System/3 (OS/3)

Distributed Data Processing

Concepts and Facilities

UP-8811 Rev. 1

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) Distributed Data Processing Concepts and Facilities", UP-8811 Rev. 1.

This revision for release 8.0 includes information on:

1. The Distributed Data Processing File Access Facility, which enables you to:
   - access and process files residing on remote OS/3 systems; and
   - write application programs that can initiate and communicate with other OS/3 systems to exchange data and control information.
2. Help screens for Distributed Data Processing (DDP) commands
3. The IMS DDP Transaction Facility which enables IMS users to process transactions between systems in the DDP network.
4. New DDP STATUS command displays.
5. Automatic DDP local and remote activity service logging.
6. The DEVICE-CLASS=DISKETTE parameter for the DDP CREATE file command.

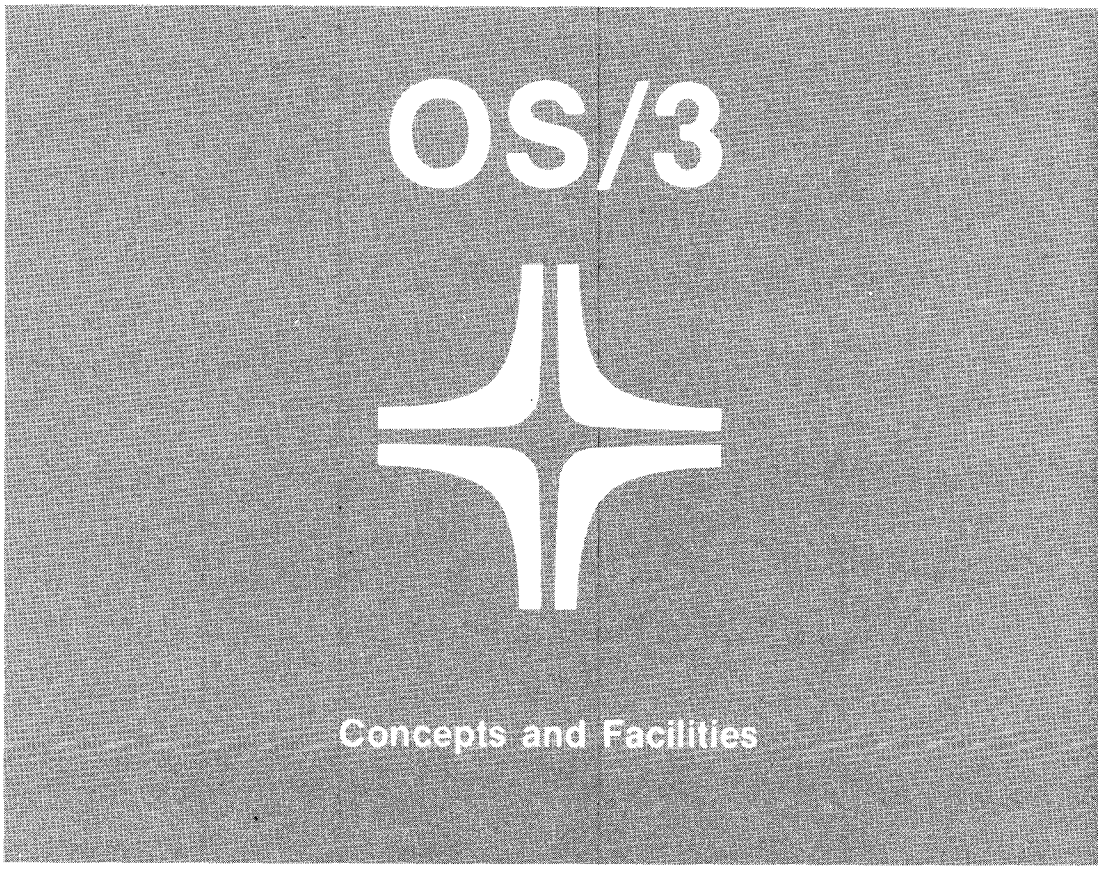
Destruction Notice: If you are going to OS/3 release 8.0, use this revision and destroy all previous copies. If you are not going to OS/3 release 8.0, retain the copy you are now using and store this revision for future use.

Copies of UP-8811 and UP-8811—A will be available for 6 months after the release of 8.0. Should you need additional copies of this edition, you should order them within 90 days of the release of 8.0. When ordering the previous edition of the manual, be sure to identify the exact revision and update packages desired and indicate that they are needed to support an earlier release.

Aditional copies may be ordered by your local Sperry Univac representative.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ,CZ and MZ	Mailing Lists A00, A22, B00, B22, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U. 76 and 76U (Cover and 199 pages)	Library Memo for UP-8811 Rev. 1
		RELEASE DATE: September, 1982

Distributed Data Processing



SPERRY UNIVAC

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-8811 Rev. 1
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Preface	1, 2	
Contents	1 thru 7	
PART 1	Title Page	
1	1 thru 20	
2	1 thru 5	
3	1 thru 3	
4	1 thru 7	
5	1	
PART 2	Title Page	
6	1 thru 3	
7	1 thru 3	
8	1 thru 41	
9	1 thru 44	
PART 3	Title Page	
Appendix A	1 thru 3	
Appendix B	1 thru 4	
Appendix C	1 thru 3	
Appendix D	1 thru 19	
Appendix E	1, 2	
Appendix F	1 thru 3	
Appendix G	1, 2	
Appendix H	1	
Appendix I	1 thru 3	
Glossary	1 thru 12	

Part/Section	Page Number	Update Level
Index	1 thru 5	
User Comment Sheet		

Part/Section	Page Number	Update Level

All the technical changes are denoted by an arrow (➔) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (➔) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Preface

This manual is one of a series designed to instruct and guide you in using the SPERRY UNIVAC Operating System/3 (OS/3). This document describes the distributed data processing (DDP) concepts and facilities.

This manual is divided into the following parts:

- PART 1. DISTRIBUTED DATA PROCESSING CONCEPTS

  Defines DDP, gives background definitions, and discusses the function of current and future DDP products. Part 1 is designed for the site administrator who needs to know how SPERRY UNIVAC DDP products fit together and for the programmer who wants to examine the concepts behind the products.

- PART 2. DISTRIBUTED DATA PROCESSING FACILITIES

  Describes in detail the use of:

  the DDP transfer facility, which sends files and jobs from one computer to another;

  the DDP file access facility, which enables users to access and process files on remote computer systems and write applications programs that can communicate with other applications programs on remote OS/3 systems; and

  the IMS DDP transaction processor facility (IMS DDP), which enables users to process IMS transactions at a remote OS/3 computer.

  Part 2 is designed for the person who uses the DDP facilities.

- PART 3. APPENDIXES

  Explains DDP commands and logging methods; show how to enter DDP messages as a batch stream and how to define your ICAM network for DDP. These appendixes are designed as reference tools. Appendix D is a special appendix summarizing DDP for the system operator; it should be removed from this manual and added to the operations handbook.

- GLOSSARY

  Defines DDP terms.

The following publications are referenced in this manual:

- OS/3 interactive services commands and facilities user guide/programmer reference, UP-8845
- OS/3 general editor user guide/programmer reference, UP-8828
- OS/3 spooling and job accounting concepts and facilities, UP-8869
- OS/3 Assembler user guide, UP-8061
- IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207
- IMS action programming in RPG II user guide, UP-9206
- IMS system support functions user guide, UP-8364

Contents

2. ADVANTAGES OF DISTRIBUTED DATA PROCESSING

3. COMPUTER CONCEPTS BEHIND DISTRIBUTED DATA PROCESSING

4. THE CURRENT PRODUCTS

PART 3. APPENDIXES

FIGURES

TABLES

PART 1. DISTRIBUTED DATA PROCESSING CONCEPTS

1. An Introduction to Distributed Data Processing

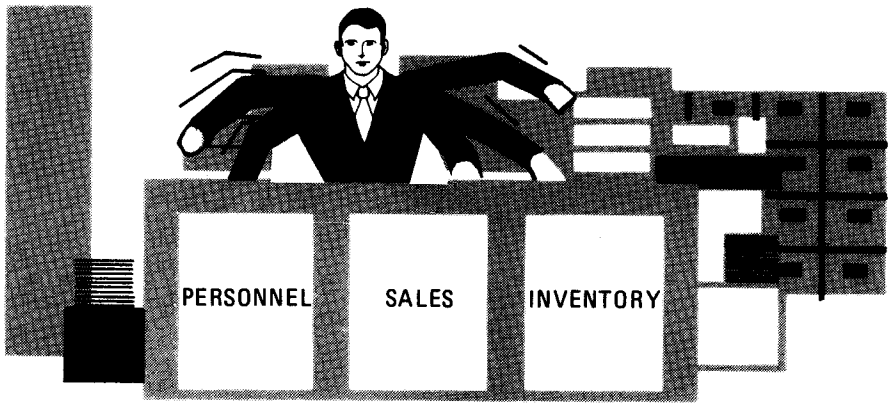
1.1. DISTRIBUTED DATA PROCESSING AS A MANAGEMENT CONCEPT

Distributed data processing (DDP) is not a hardware or software product. It's a way to manage data within an organization. Instead of centralizing all your data at one location, you distribute it to the sites where it's needed most. Then you use electronically linked computers to move data from one site to another or to access or change it from a remote location.

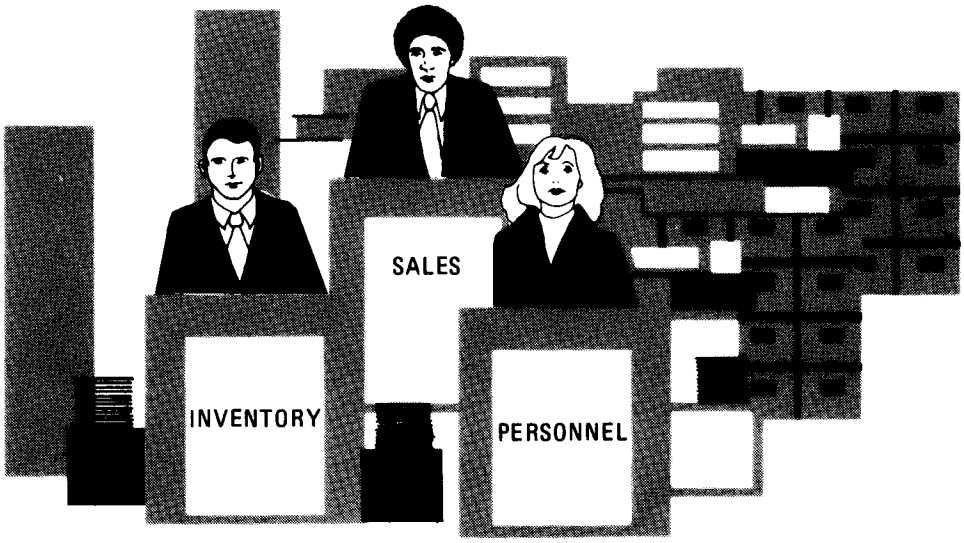
Because the term *distributed data processing* means different things to different people, the first thing we'll do is define what Sperry Univac means by the term.

1.2. WHAT IS DDP?

We're all familiar with how businesses grow. At first, there's just the owner, who makes all the decisions and has all the responsibility. He buys a building, orders supplies, hires some workers, supervises what they do, and markets his product. He's his own personnel manager, inventory supervisor, and salesman all rolled into one.

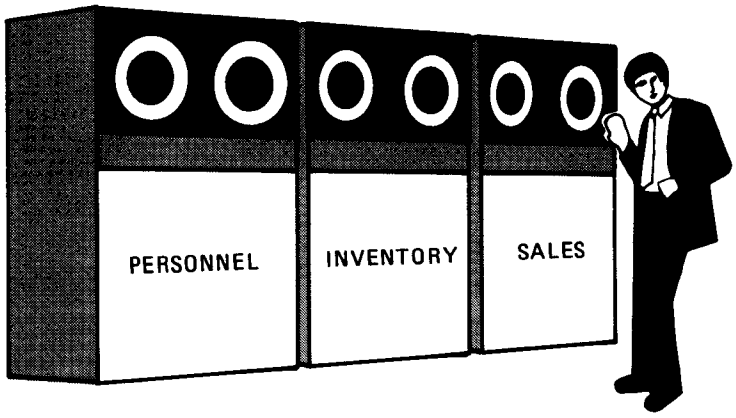


But, as his business grows, he can't keep control of everything himself. He needs help. So he takes partners, and they in turn hire department managers. Maybe there's a new branch with its own managers. All these people are making decisions – responsible, serious business decisions that affect the whole enterprise. The original owner no longer has total control. In other words, decision-making responsibility is *distributed* across many managers.

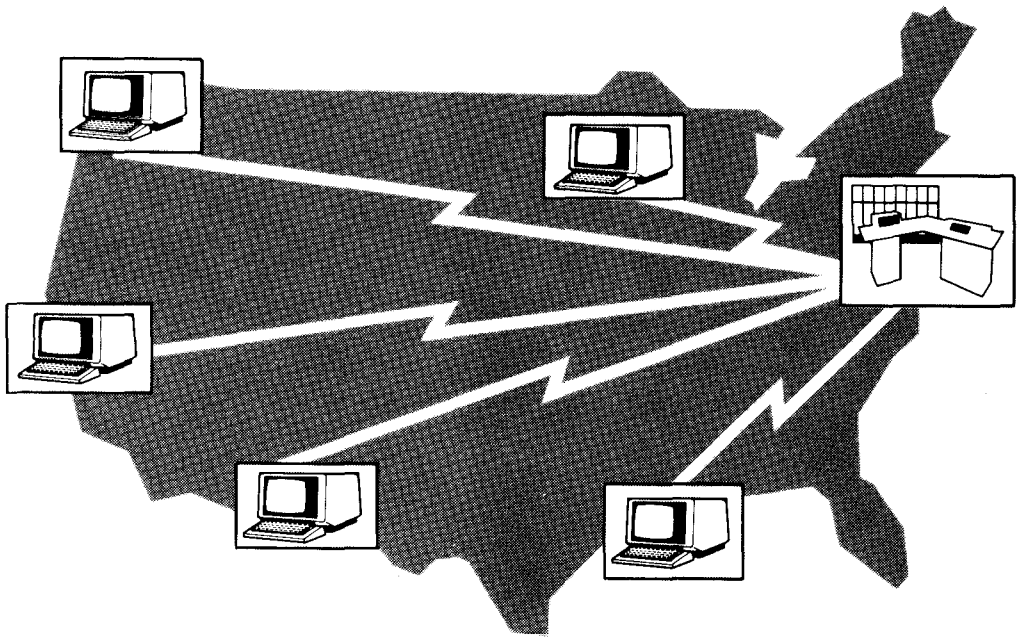


The business is kept on course through communication. Information flows up and down the management ladder. Department managers keep their superiors informed of their actions. People in the president's office exert central control.

The business communication process is a lot easier since the introduction of computers. Managers have rid themselves of mountains of files. Reports that used to take days to research and type now take only minutes.

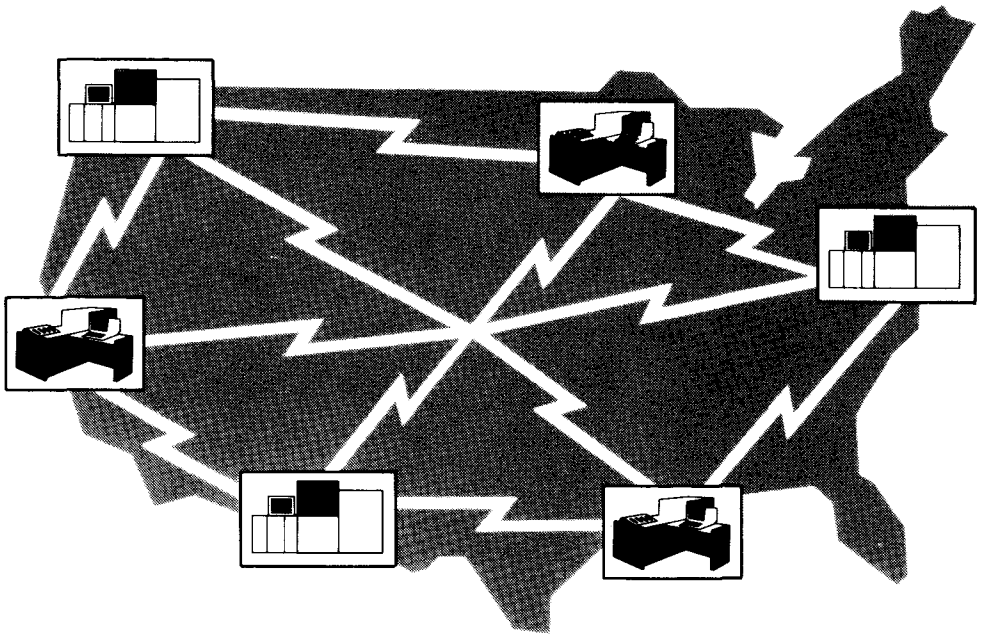


But, until recently, computers had a serious disadvantage. Often, there was only one of them in a company and everybody had to use the same one. That meant that, if you were the branch manager in San Francisco with your headquarters in New York, you had to send all your raw data through a terminal to headquarters, then request reports back through the same terminal. To keep communications costs down, you may have communicated only at night. You may have waited a day or more for reports. And you didn't have control over the most important management tool of all – your own data.



Less expensive computers changed all that. Now you can put small computers in branches so local managers can control their own data. But what about sending information up and down the management ladder? With computers scattered all over the company, central managers might not have easy access to the data they need to keep everything coordinated.

To get the administrative advantages of the local computers while at the same time keeping central control, you need a distributed data processing (DDP) system. In a DDP system, your independent, local computers are linked so that any computer can access another's data or send jobs to it. Just as administrative responsibilities in a large company are distributed over a number of managers, so in a DDP system jobs and data are distributed over a number of computers. And just as those managers coordinate their work through communication, in a DDP system computers coordinate their work to access jobs and data on any other computer in the system.

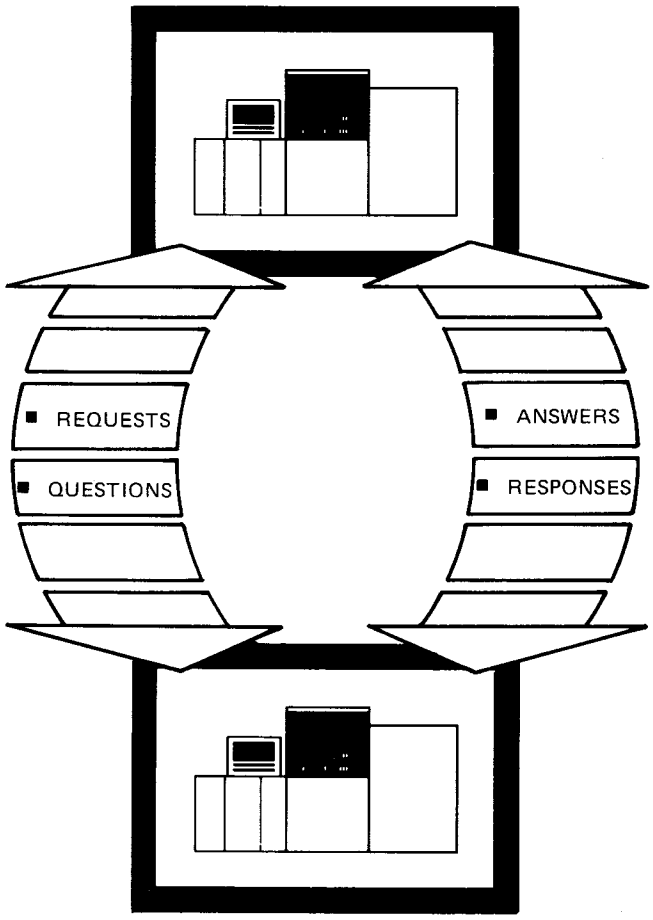


We can look at a DDP system in three different ways. First, it has a physical aspect: DDP links independent computers. Second, it has an administrative aspect: DDP distributes both jobs and data over a number of linked computers. Third, it has a functional aspect: DDP forms a layer between you and the communications network. Let's examine each of these aspects in detail.

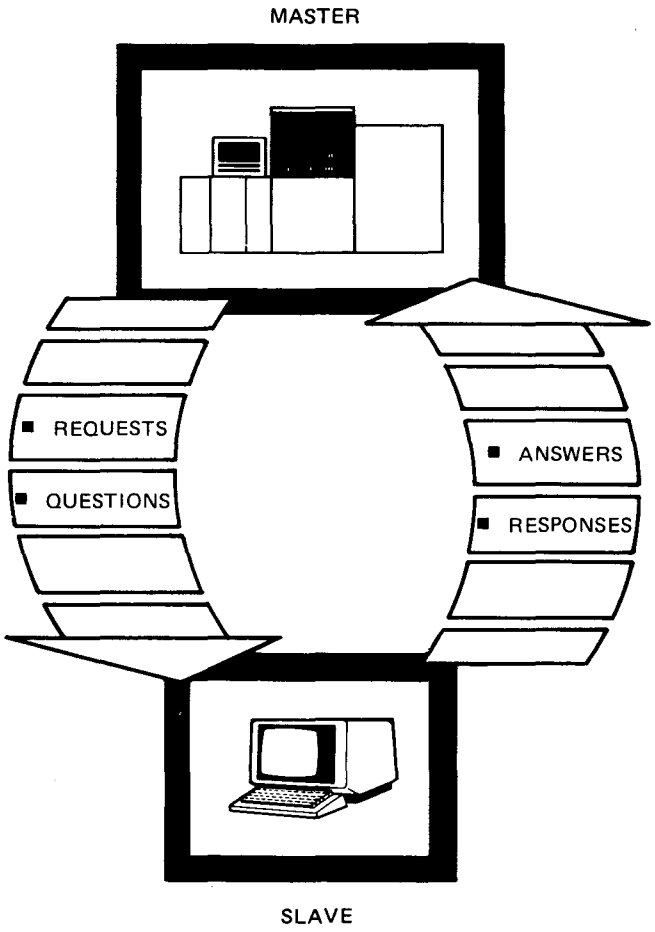
1.2.1. DDP Links Independent Computers

A DDP system contains at least two computers capable of independent processing. They are linked together through telecommunications so that each can use the other's capabilities and data.

Computers work as equals in DDP. That means that any computer in the system can ask for data from the other, and any can send a job to another.



At any time, of course, one computer may be *in charge,* controlling, for instance, the copying of a file. But the computers may switch roles for different jobs. That's different from a master-slave relationship, in which one piece of hardware is always in charge and the other always does its bidding.



The joined computers have independent operating systems. They can communicate and process transactions independently. But they can also be joined as equal hosts – not terminals – in a DDP system, allowing each computer to perform the job it does best. Thus, work is distributed over several computer hosts.

1.2.2. DDP Distributes Both Jobs and Data over a Number of Hosts

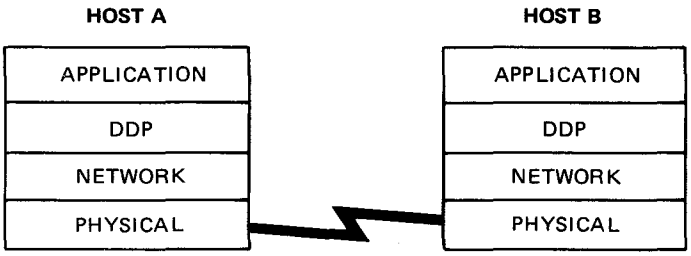
What is *distributed* in a DDP system? First, jobs. You can send a job to any other host in your DDP system and get the results back just as if your own host had done the job. Second, data gets distributed. You can put files on the host where they're needed most, but others can get records from them or copies of the whole file.

1.2.3. DDP Forms a Layer between You and the Communications Network

The third way to view DDP is to look at the function DDP products have in your computing system. DDP is that layer of software products and their associated hardware devices that lies between the user and a communications network. Current products in the layer let you:

- send files and jobs anywhere in your system;
- access and process files anywhere in your system;
- write programs that can communicate with other programs anywhere in your system; and
- use your information management system (IMS) action programs to access records on a remote host.

More products will follow.



1.2.4. DDP Definition Summarized

DDP is a computer management concept giving local managers control over their data and jobs, while at the same time allowing central access to all data. Local computers that normally operate independently are linked together through telecommunications so that an operator on any Series 90 or System 80 computer can access data on any other Series 90 or System 80 computer or send jobs to it. Sperry Univac makes a DDP system work for you by providing DDP products that form a layer of service between you and the communications network.

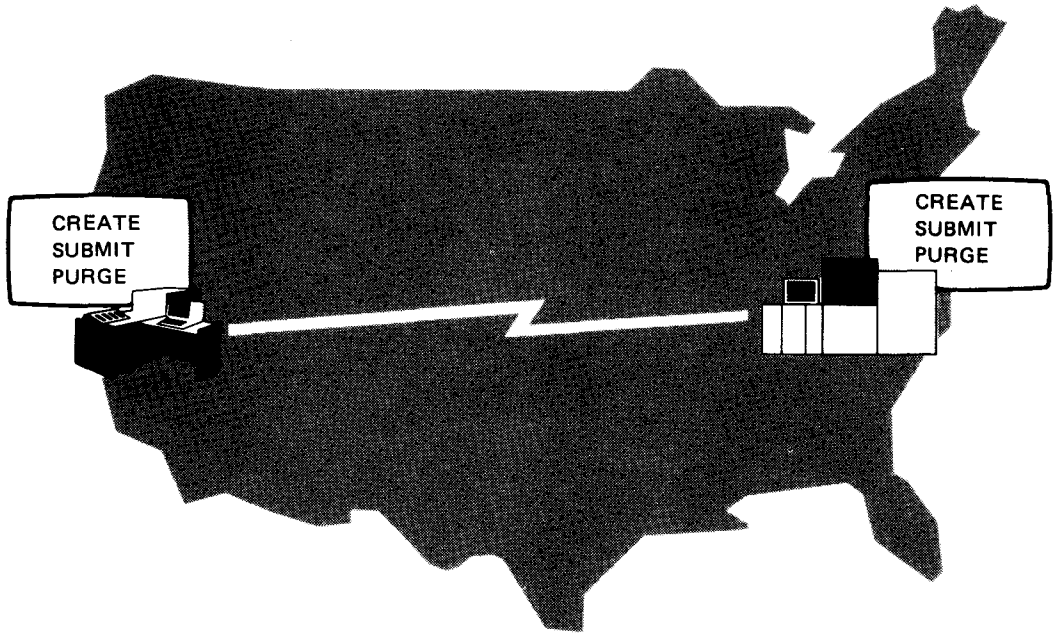
1.3. REQUIREMENTS FOR A DDP SYSTEM

The two requirements for a DDP system are:

1. Two or more computers must be logically separate. Each host must have an operating system so it can function independently.

2. An electronic link must join the hosts so they can work cooperatively on the same project. This link makes the capabilities of each host available at any host. It lets you distribute projects across different hosts.

In addition to these requirements, Sperry Univac gives you a bonus. We're using the same set of commands to perform DDP tasks on different computers. Identical commands achieve identical results anywhere in the system.



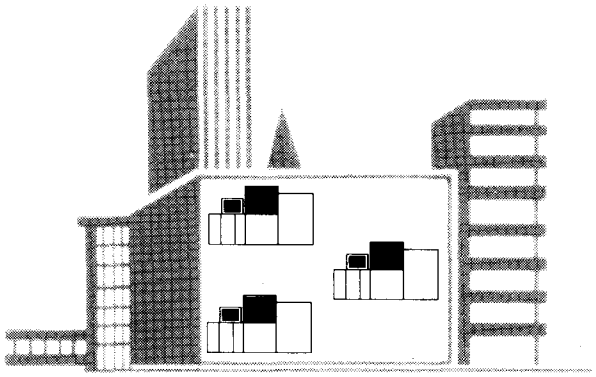
DDP REQUIREMENTS

- **Electronic Link**
- **Geographical Separation**
- **Common Command Language**

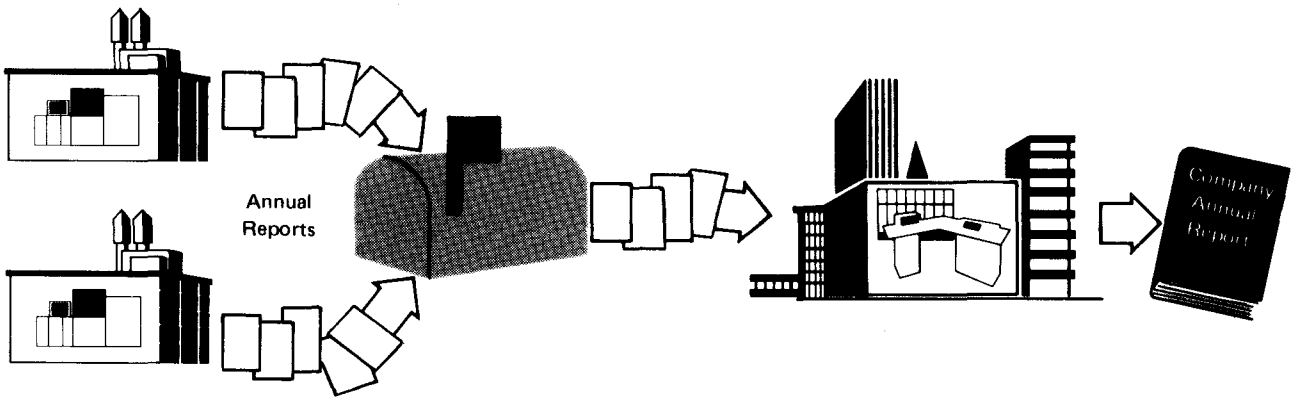
1.4. WAYS TO JOIN COMPUTERS IN A DDP SYSTEM

A DDP system contains at least two computers capable of independent processing. But, some ways to join computers don't meet the DDP requirements.

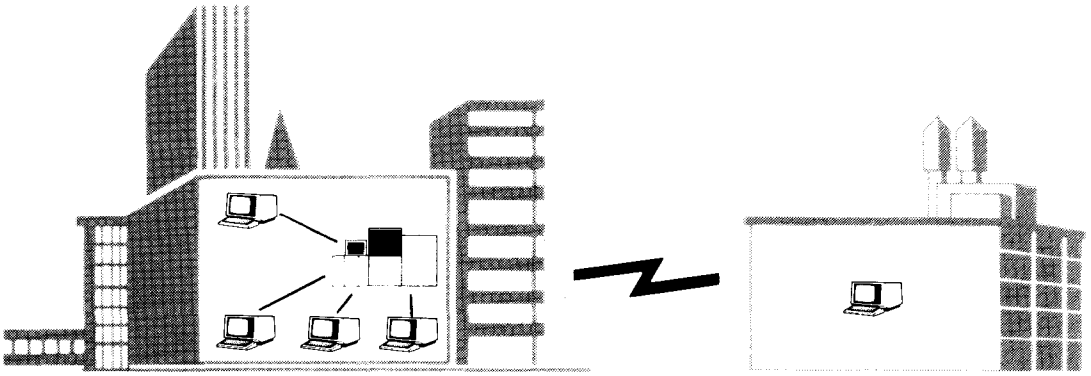
If the computers have no links, if the link isn't electronic, or if only slave terminals are joined, you don't have DDP.



a. Independent computers with no link



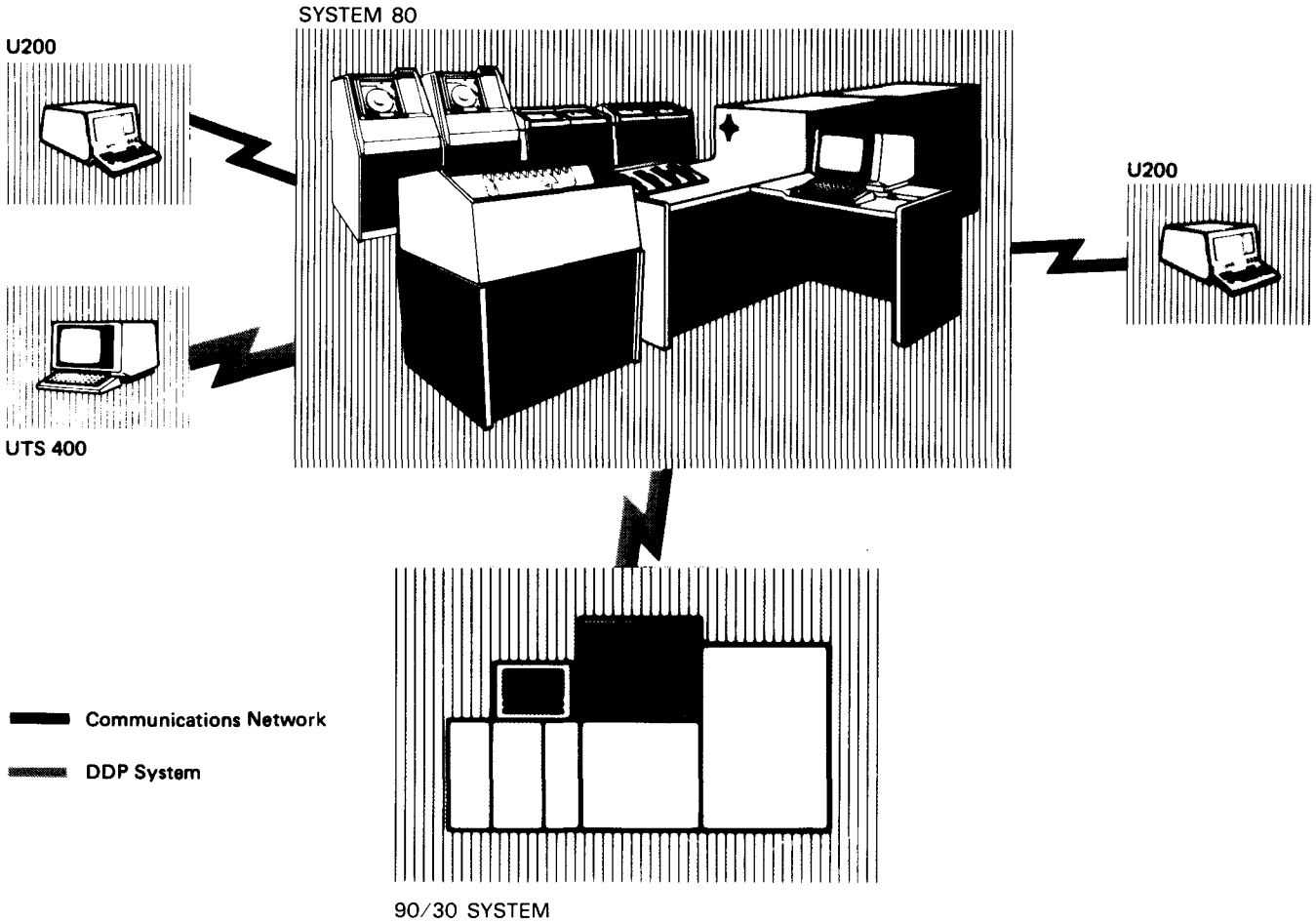
b. Independent computers whose link is not electronic



c. A variety of processing devices linked to a single computer

HARDWARE CONFIGURATIONS - That are not DDP

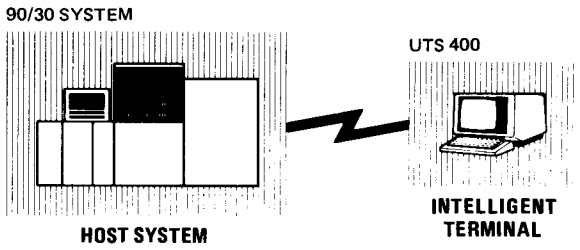
Now let's see what a DDP system looks like. You'll recognize the configurations right away. They're the same forms you find in any communications network that joins nodes. But there is a difference. A communications network joins both hosts and terminals. Since a DDP system is a subset of a communications network, only independent hosts are joined, though of course those hosts may themselves have terminals.



HARDWARE CONFIGURATION USING DDP

So if you already have a communications network, you won't have to make any changes in it when you move to DDP.

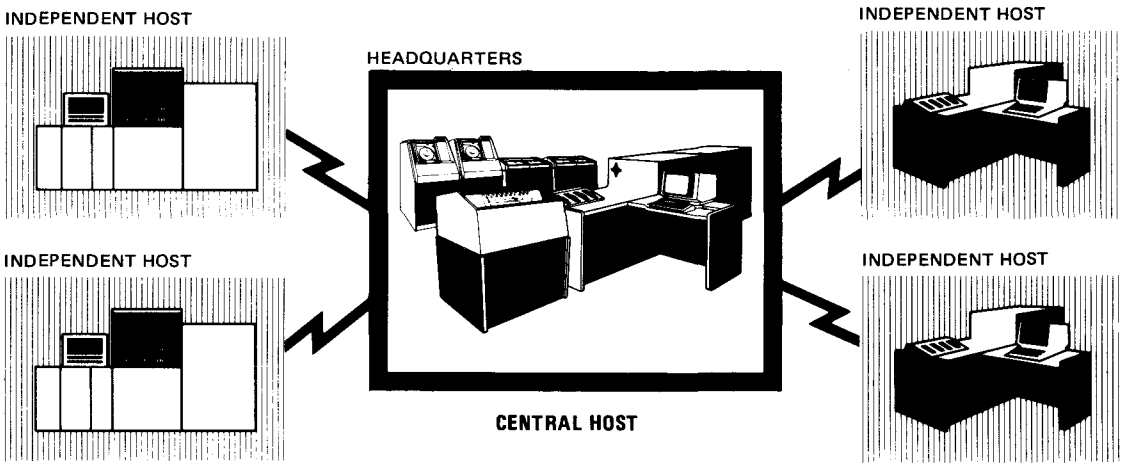
A programmable terminal is the simplest form of independent computer. So, the simplest DDP configuration is a link between a processor and a programmable terminal. For instance, a SPERRY UNIVAC 90/30 System processor and a programmable SPERRY UNIVAC UTS 400 Universal Terminal System form a simple DDP configuration.



A MINIMAL DDP SYSTEM

Although the UTS 400 is normally used as a terminal to a larger computer, it is capable of independent processing. Once its programs are loaded from the 90/30 processor, it can store and call them with no further help.

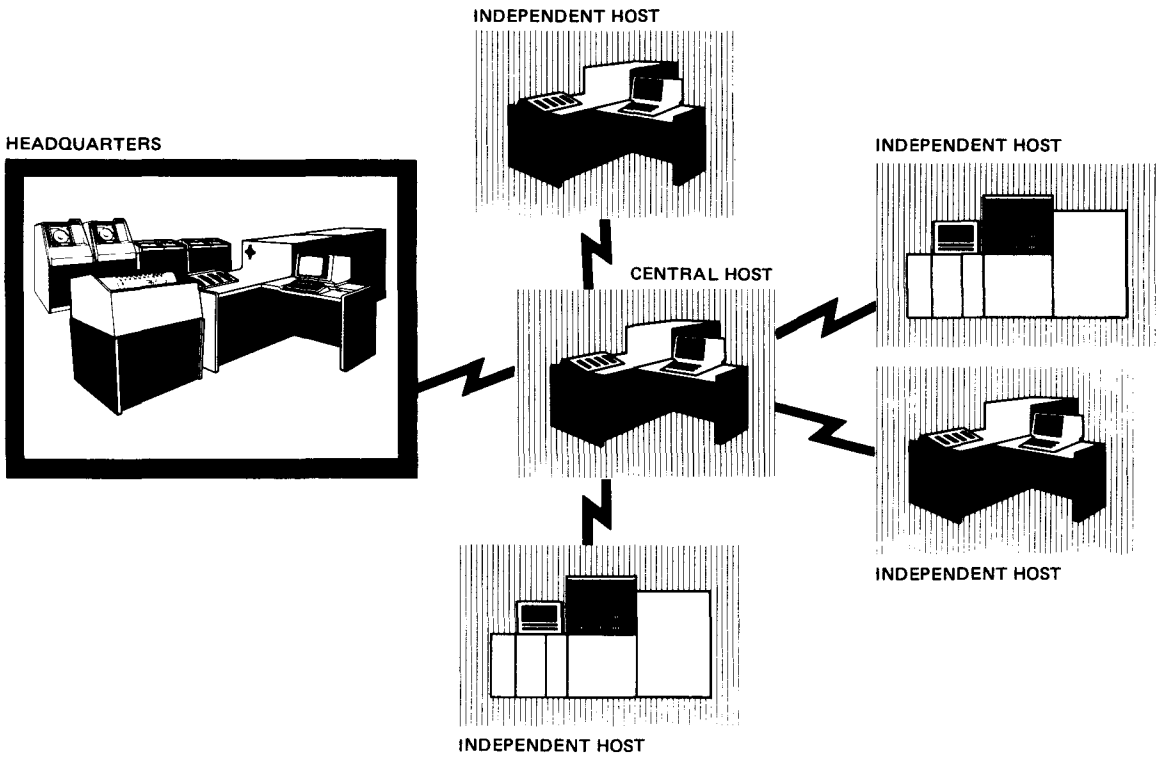
The *star* is one of the structures most used in DDP, just as it is in communications networks.



A STAR SYSTEM - Involving a "hierarchical" relationship among hosts

Star structures are hierarchical. A central host controls the system. Subordinate, though still independent, hosts have links to the central computer but not to each other. It's important to realize, though, that the DDP system itself treats all hosts as equals. Each can request data from the others. The central host, in addition to its other functions, switches messages from one host on a point of the star to another.

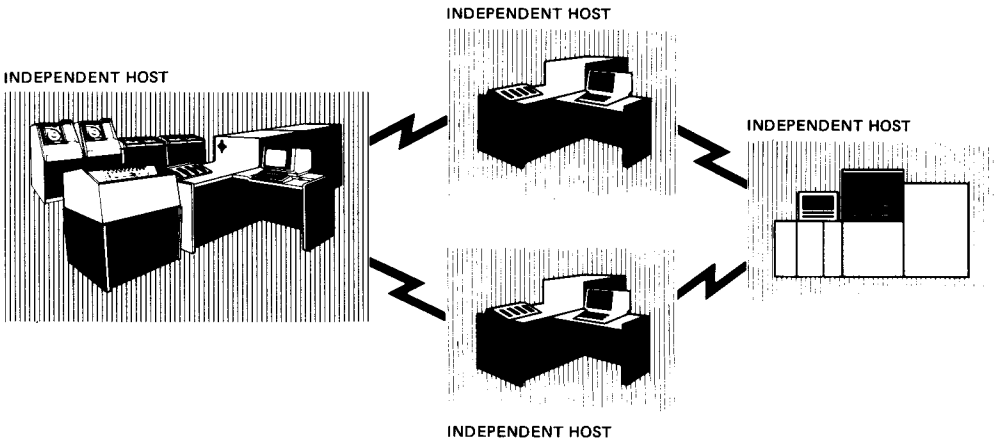
You may make the central host the largest or make it monitor the functions of the other hosts. But you may also decide differently. For instance, your headquarters computer could be on one of the points, and the central host could be a small computer mainly managing traffic. This structure frees the headquarters computer from traffic management.



A STAR SYSTEM - With the largest computer on one of the points

A star structure with many points transfers messages rapidly, since each message gets to its destination with no more than one intervening host. Fast communication makes the star efficient.

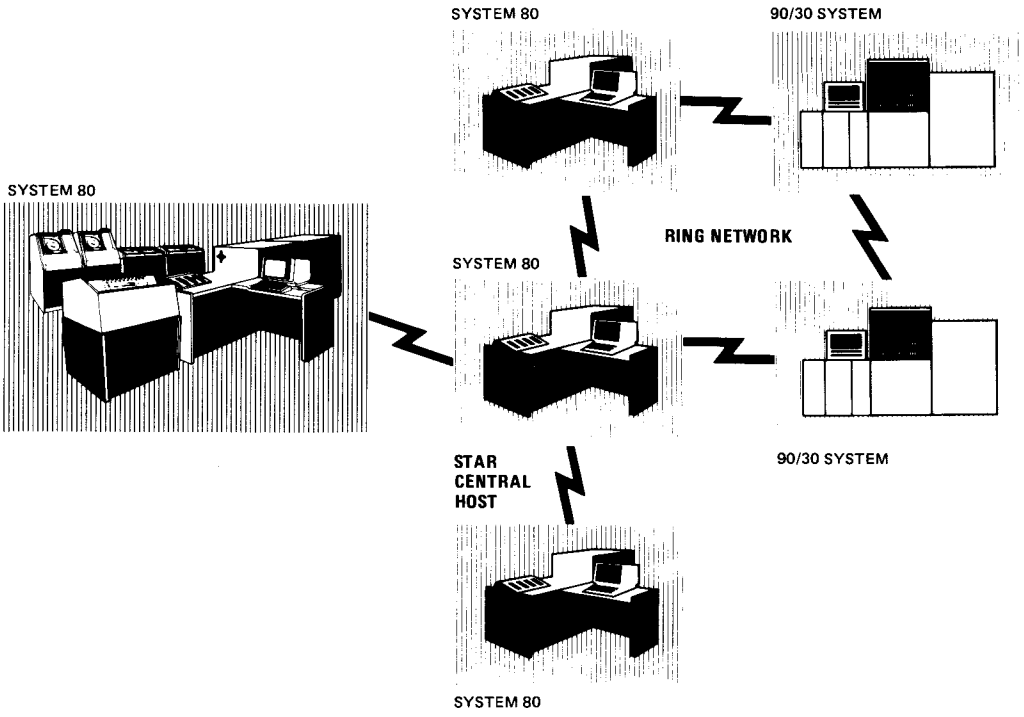
DDP configurations can also use the ring form of communications.



A RING SYSTEM

The ring is nonhierarchical. There's no central traffic manager. Each host communicates directly with its two contiguous hosts. Communications to the more remote hosts go through each intervening host. Thus communications are slower. For this reason, the ring is less efficient than the star. You might want to use it, though, if line costs are less for a ring than for a star structure.

Of course, you can combine these types. And most DDP systems will. A tree structure can combine star and ring structures. A single host may be in star or ring configuration with other hosts in addition to being connected to still other hosts and configurations. The important thing is that DDP treats all hosts in the network as equal.



A TREE SYSTEM

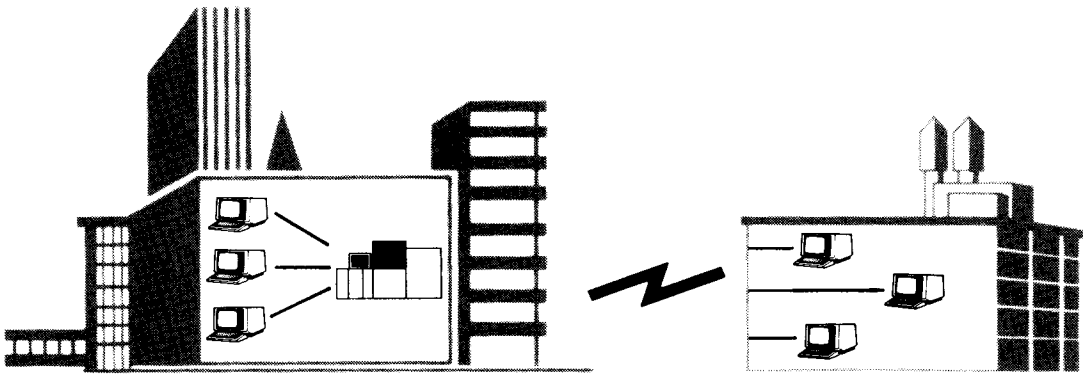
So what do these various DDP structures mean to you? They mean you can have whatever configuration of hosts your company needs to process data efficiently.

1.5. HOW DDP EVOLVED

Data processing systems have evolved from centralized to decentralized to distributed. And distributed processing itself didn't emerge instantly. Instead, it evolved slowly in response to user needs. Distributed systems are well-suited to today's management needs.

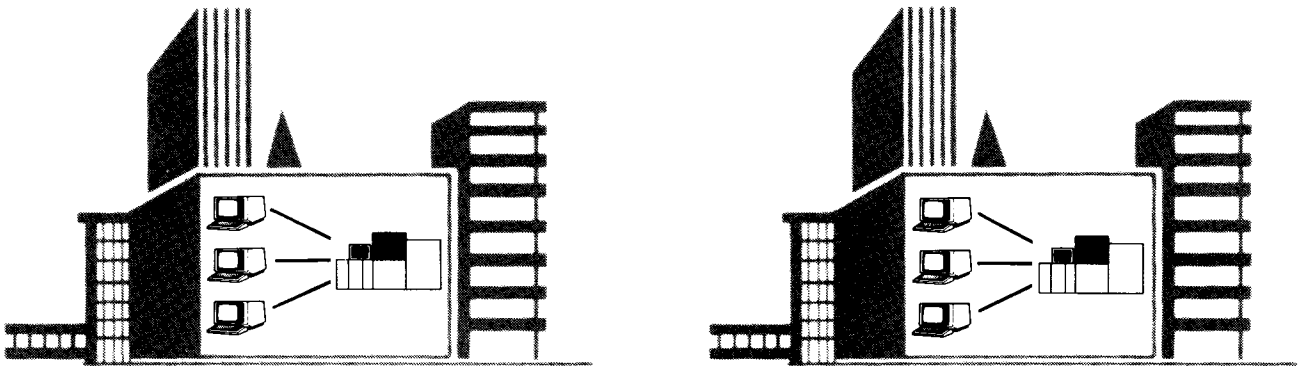
1.5.1. Definitions of Centralized, Decentralized, and Distributed Systems

- A *centralized computer system* is a central processor performing varied tasks for many users. It generally combines batch processing with electronically connected terminals that send and receive data but don't process it themselves.



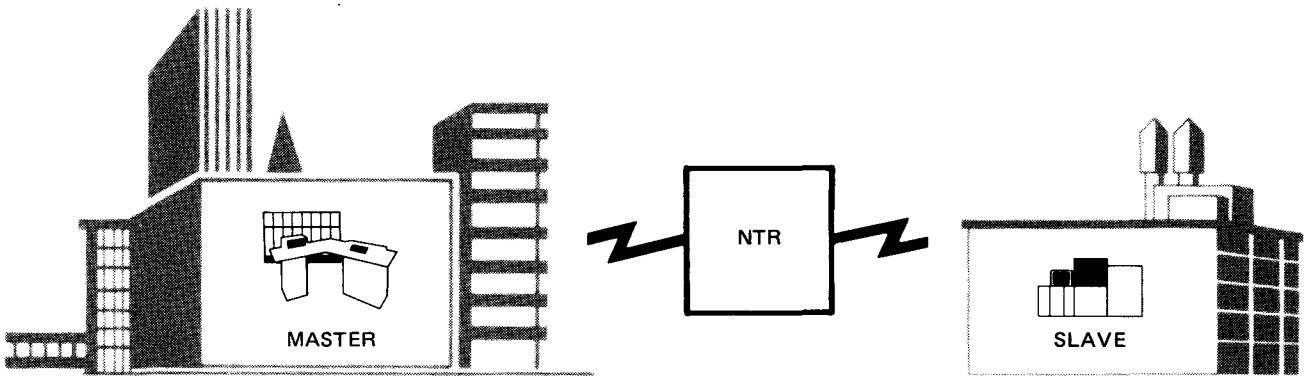
A CENTRALIZED SYSTEM - A central computer and its terminals

- A *decentralized computer system* has several different computers, either large or mini. These computers are not connected as equals, where each seeks data from the other. Instead, they operate independently.



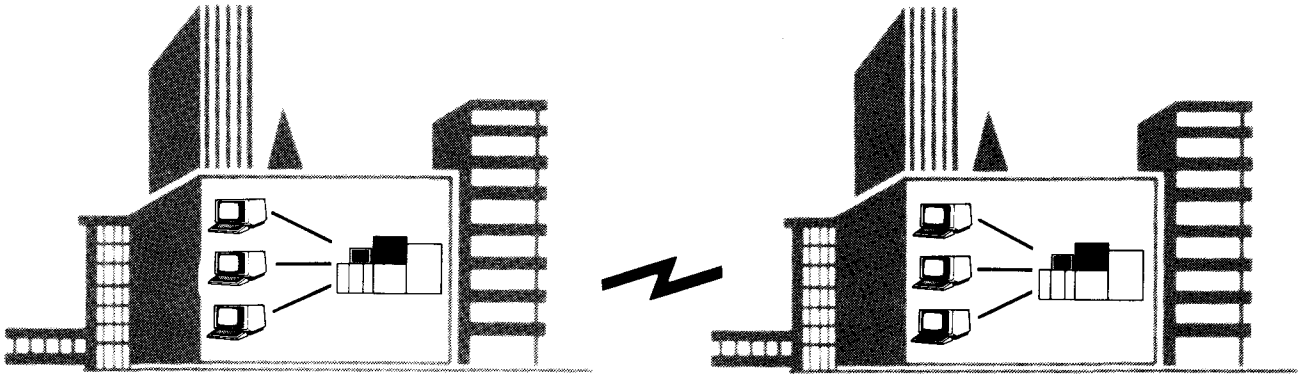
A DECENTRALIZED SYSTEM - Independent computers with no electronic link

Computers in a decentralized system may be temporarily connected to work on the same task. But unless they're connected as equals, they're not in a DDP system. For instance, nine-thousand remote (NTR) connects the SPERRY UNIVAC 1100 System and the SPERRY UNIVAC 90/30 System so that the 1100 system is always the master and the 90/30 system is always the slave. They can't change roles with NTR. So NTR isn't, strictly speaking, a DDP product.



A DECENTRALIZED SYSTEM - Not a distributed system; NTR doesn't join the computers as peers

- A *distributed computer system* combines the electronic link of the centralized system with the independent computers of the decentralized system. Functions within the computers can cooperate on the same task, and any computer in the DDP system can initiate or control a particular task.

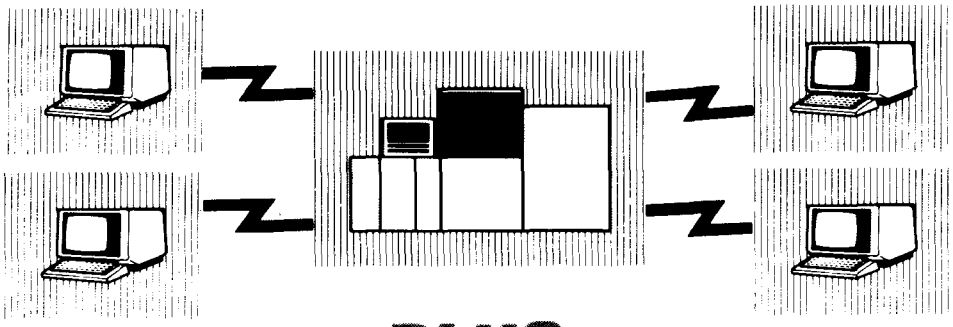


A DISTRIBUTED SYSTEM - With electronic link between two independent computers

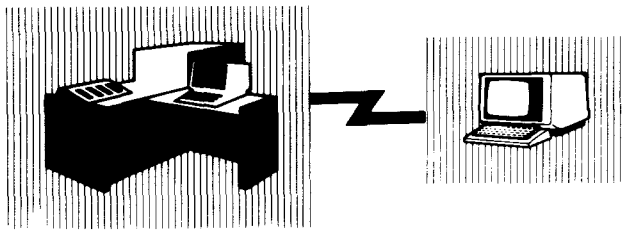
1.5.2. The Evolution from Centralization to Decentralization to Distribution

Because of the cost of the earliest computers, the limited number of people trained to operate them, and specialized site requirements, the earliest data processing systems were centralized. Very few users could justify the cost of multiple systems. Gradually, however, decentralized and then distributed systems evolved from centralized ones.

Decentralized systems evolve from centralized ones when you add an independent computer that's not connected to the old one. This new computer may be small, or it may be as elaborate as the central system.



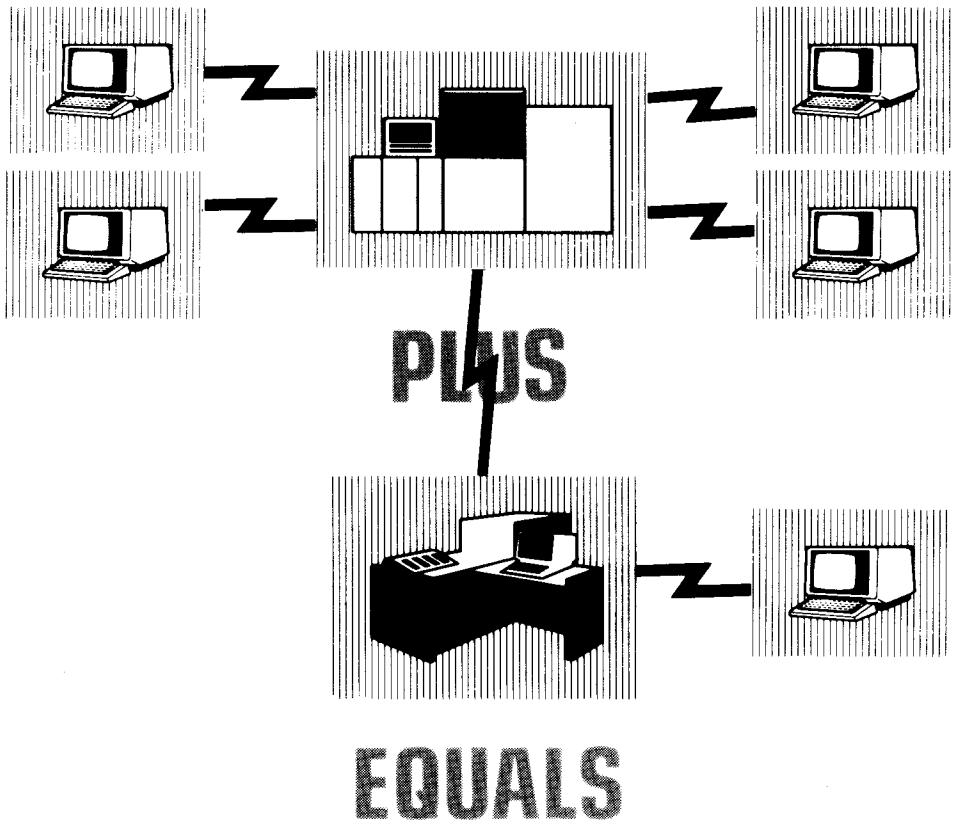
PLUS



EQUALS

A DECENTRALIZED SYSTEM

Decentralized systems become distributed systems when you join the independent computers as equals. Each of the joined computers must be able to access data on any other.

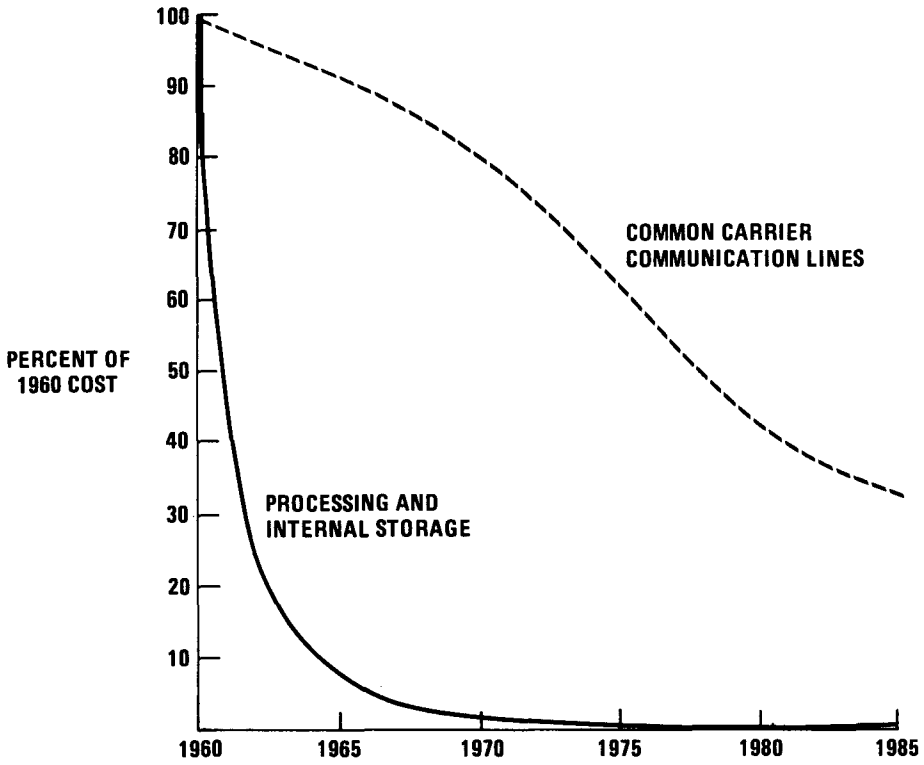


A DISTRIBUTED SYSTEM

1.5.3. Hardware and Communications Developments Leading to DDP

Centralized systems developed for a very good reason: cost. The original computer hardware cost so much that companies could generally justify the cost of only one computer and had to keep it running constantly to realize savings over manual methods. But the history of computer hardware costs is one of continually declining prices. Using almost any standard of measure (units of main storage, throughput), hardware costs only about 10 percent of what it did 10 years ago. And experts project the cost will fall to about one-tenth of its present amount over the next decade.

In the USA, the cost of computer communications has also fallen, but not as much as computer hardware costs.

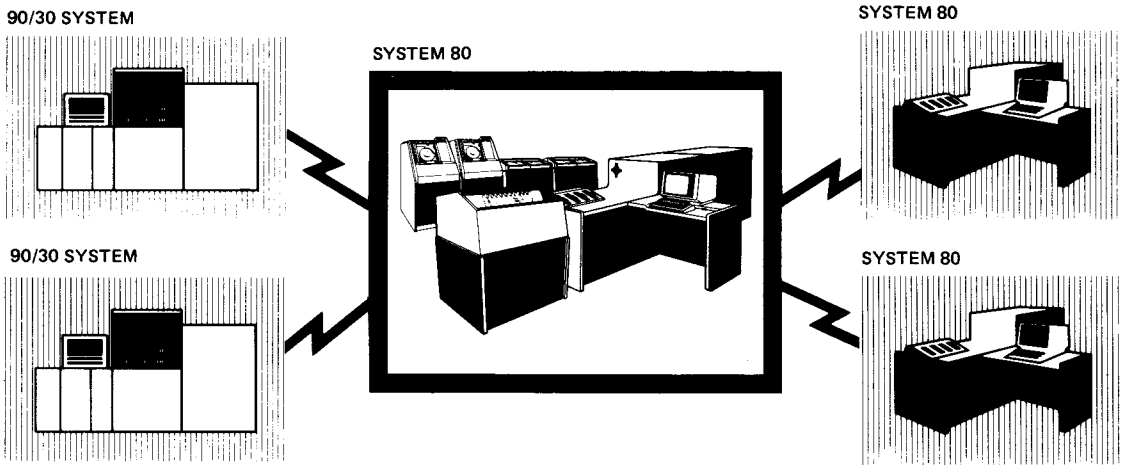


HARDWARE AND COMMUNICATIONS COSTS – 1960 TO 1985

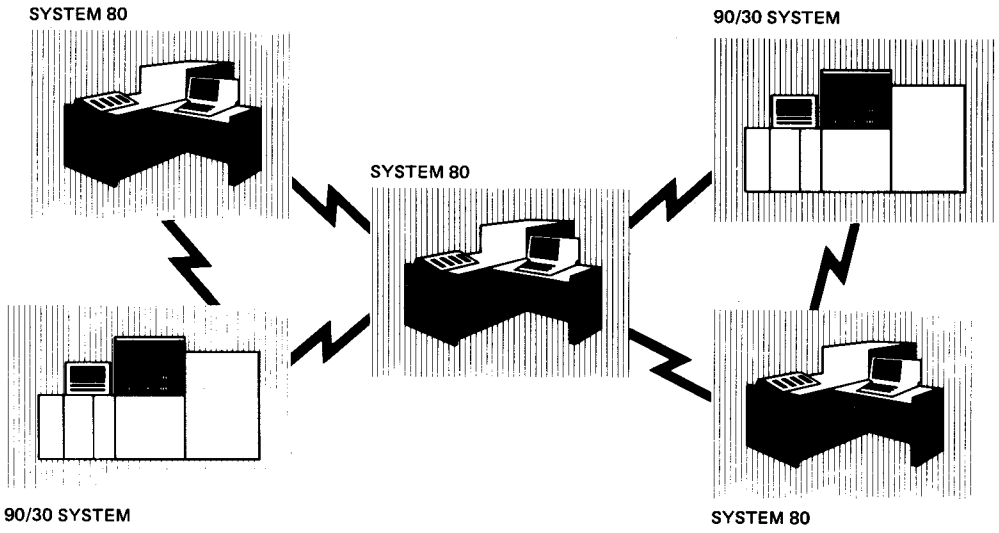
Because communications costs in both the USA and Europe are controlled by the telephone system, future costs are uncertain. Many companies communicate vast amounts of raw data from terminals to a centralized computer. It may be cheaper for them to do some processing at the original site on relatively inexpensive hardware, and then to communicate less data. In many cases, the lowered communications costs more than pay for the new computer.

Many companies already have small computers in the field, sending back reduced data through methods such as NTR. But as hardware costs continue to fall and as the field computers become more powerful, it makes less sense to use the field computers simply as terminals to a central computer. With DDP, computers can send jobs to each other during busy periods and can access facilities they don't have. At the same time, they can continue to send back processed data to a headquarters computer and to perform strictly local functions. Linking hosts saves money because the demand for excess capacity at each site during busy periods disappears.

You may decide to structure your DDP network with a main computer and subsidiary hosts.



Or you may do away altogether with a main computer and rely wholly on the combined capacity of your hosts.



Whichever you choose, Sperry Univac offers you a range of hardware and software products to ensure that your system meets your demands.

1.6. A BRIEF LOOK AT DDP PRODUCTS

The available DDP software products are:

- The DDP TRANSFER FACILITY

  It allows you to send files and jobs between OS/3 computers. This facility enables you to inquire about records on another OS/3 computer or copy its files and update them, using simple English-based commands.

- The DDP FILE ACCESS FACILITY

  It enables you to access and process files on remote hosts and write application programs that can communicate with other application programs on remote OS/3 systems.

- The IMS DDP TRANSACTION PROCESSOR FACILITY

  It enables you to process IMS transactions at a remote OS/3 computer in three different ways:

  1. Directory routing –

     whereby the terminal operator can enter a transaction code at the primary IMS center – that locates the transaction at a particular remote site and processes it.

  2. Operator routing –

     whereby the terminal operator can enter a special character – that locates the transaction at a remote site and process it as in the directory routing manner.

  3. Action program routing –

     whereby the terminal operator can initiate a transaction at a primary IMS center via an action program.

Right now, these products can only be used in OS/3 systems. In the near future, however, enhancements of DDP facilities will enable you to link your OS/3 system with non-OS/3 systems.

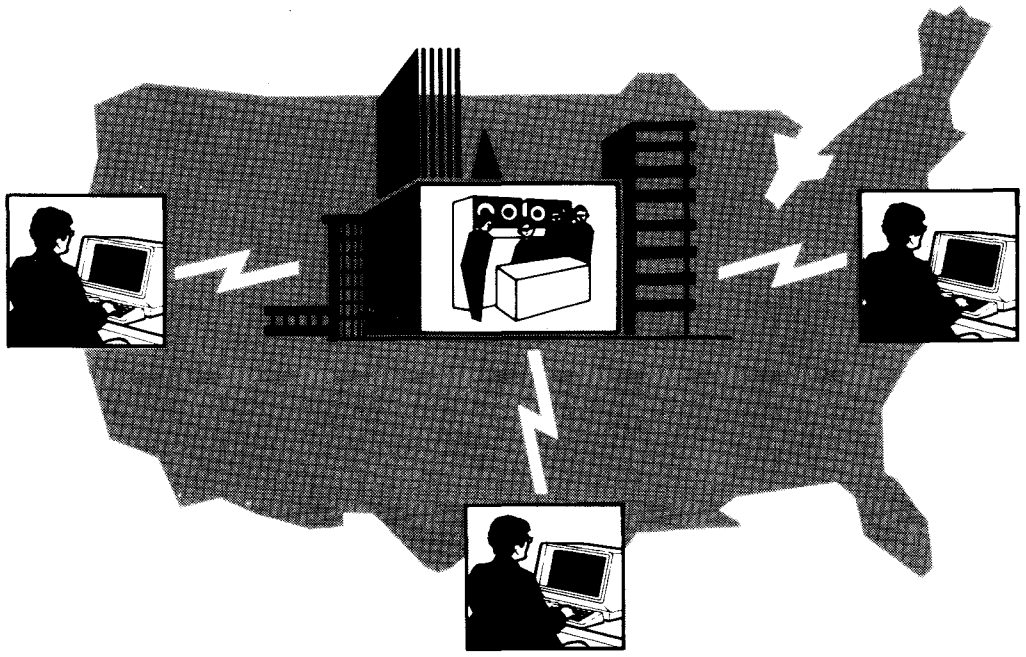
2. Advantages of Distributed Data Processing

2.1. OVERVIEW

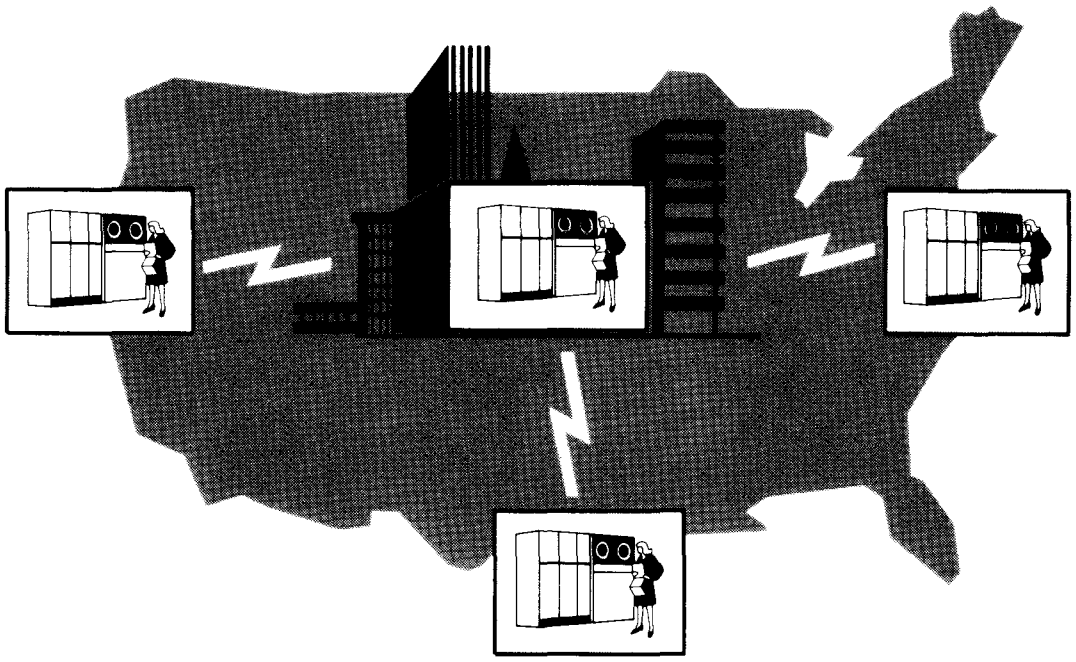
SPERRY UNIVAC DDP products help you take advantage of recent computer developments that increase responsiveness and lower costs. The advantages of DDP deserve close examination.

2.2. DDP LETS MANAGERS GET DATA FASTER AND MORE EASILY

Studies indicate that efficiency increases when a manager has control over all the tools necessary to make decisions. Computerized data is a vital tool. Under a centralized computing system, managers may not have immediate access to their own records, which are compiled and stored elsewhere. Special reports may take longer to complete than they did under older, manual methods. And managers may not welcome new centralized computing functions, since such functions can mean loss of control.



Under a DDP system, however, local managers control the local computing site, the personnel, and the data. Increased communication between managers and data processing people can result in better services for local needs. Thus, local managers find themselves using more of the services the computer can perform.



2.3. DDP SUPPORTS THE ORGANIZATIONAL STRUCTURE OF YOUR COMPANY

Centralized computer systems make sense in a centralized organization. A centralized system, for example, may use remote terminals to collect raw data from branches, and use those same terminals to send back directions from the central, decision-making group. If the only managers with authority are at central headquarters, it makes sense for them to control all computer data.

Similarly, a decentralized system works well in a decentralized organization. If a manager's only responsibility to headquarters is a once-a-year profit report, and if his branch has little contact with other branches, it makes sense for him to have a strictly local computer system.

But most business organizations don't fall into either of these categories. Instead, they use what we might call a *distributed* management system, whereby decisions are made at different management levels. A distributed data processing system offers the ideal compromise: central coordination of, and access to, data processed and located at decentralized sites. Distributed organizations avoid too much central control on the one hand and *every man for himself* lack of coordination on the other. Local managers control local data, but central managers can quickly access it.

And the SPERRY UNIVAC DDP products minimize problems in sending data between hosts whose basic structures are quite different.

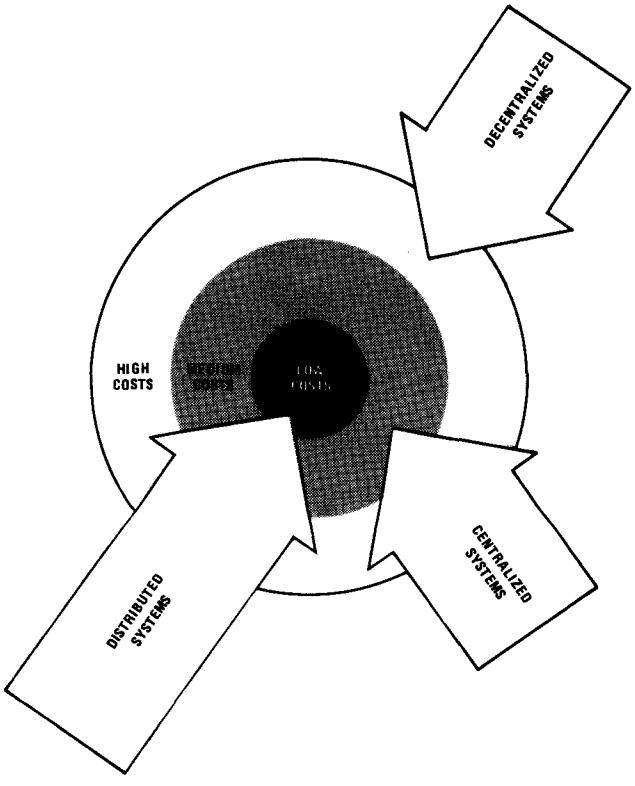
2.4. DDP LOWERS COSTS

As we mentioned earlier, the price of computer hardware has fallen over the past 10 years to about 10 percent of its original price. And future dramatic reductions are anticipated. Communications costs, on the other hand, have decreased much less rapidly than hardware costs, and future trends are unpredictable. If you install a central computer with many remote terminals that send raw data to it, communications costs may be very high. But if, instead, you process that raw data on a small computer at the remote site and then send just a summary, communications costs are bound to be less.

DDP may also decrease the load on your central computer and increase throughput. The computer needs to spend less time in switching among competing jobs and can devote more resources to accomplishing tasks. This may allow you to retain your present computer rather than having to purchase a larger one.

DDP also lets your organization share resources. A department that can gain access to a large computer in another branch may not need a large computer of its own. Through DDP, occasional users can access large systems promptly and efficiently as needed, avoiding demands for a larger local system.

DDP may also help you control personnel costs. Let's say you're planning to put independent computers at five different sites in your company. In the past, that might have meant a 500 percent increase in data processing personnel. But since DDP makes communications and coordination among those hosts faster and easier, you may be able to centralize skilled personnel and have smaller staffs of less skilled persons at the remote sites. Using DDP products, the central staff may be able to do a great deal of the program development and testing that otherwise would have been done at the remote sites.



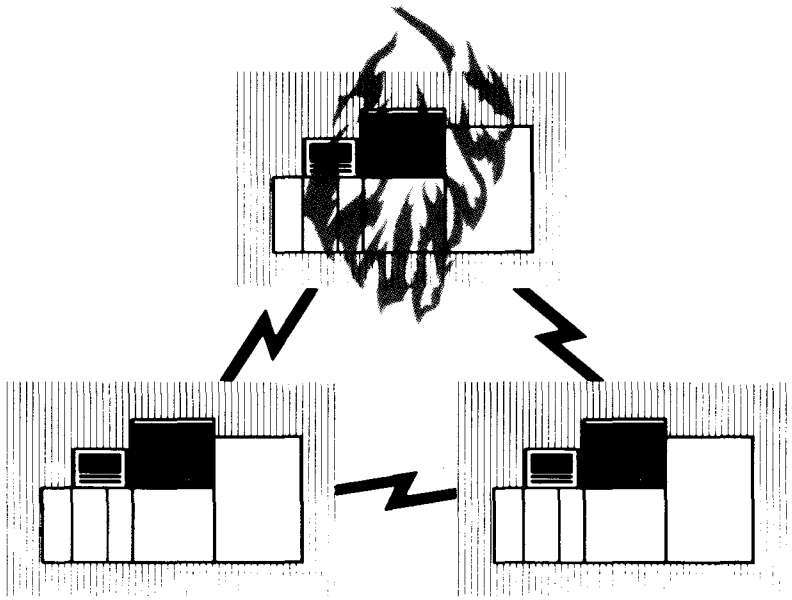
2.5. DDP COMMANDS ARE THE SAME THROUGHOUT THE SYSTEM

Sperry Univac is developing a common DDP command language for use with all Series 90 and System 80 systems. This command language lets operators on all systems use an identical set of commands to achieve identical results, no matter what computer they use. These commands are discussed in Part 2 of this manual.

The common DDP command language is easy to learn and use. It gives operators and programmers a tool that's portable from one computer to another. The common set of commands doesn't eliminate job control language, but it does mean that the commands the operator uses on the remote host are the same as the commands on the local host.

2.6. DDP HELPS YOU SAFEGUARD YOUR DATA

Some companies now make copies of files to send to a remote site to guard against the total destruction of records at one site. DDP can make this task easier or make the files more readily accessible. You can use DDP to make copies of your files on a remote host. Then, if files are destroyed, they can be accessed immediately on the remote host or recopied from the remote host. Thus, an accident or failure at one host doesn't destroy vital company data.



2.7. DDP HELPS YOU CONSERVE YOUR STORAGE FACILITIES

By being able to access and process remotely located files, DDP helps you conserve your storage facilities. A file that is seldom used at a particular site need not occupy file space at that site. Instead, whenever that file is needed, it is simply accessed through the DDP file access facility.

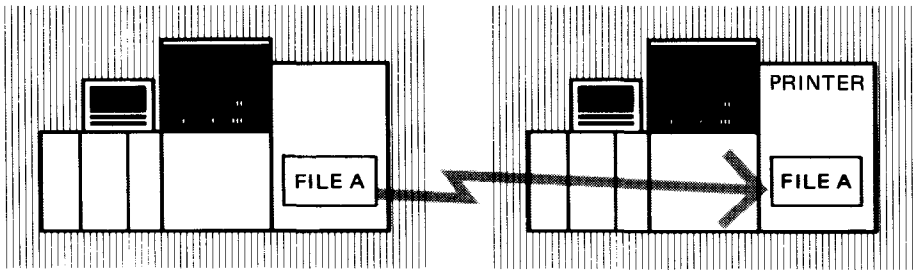
2.8. DDP HELPS YOU MAINTAIN PROGRAM CONTROL BETWEEN REMOTE SYSTEMS

By providing the ability to have programs initiate and communicate with one another, DDP allows you to write programs that control the execution of other programs at remote sites. Thus, an application that requires several programs to be executed in a sequence determined by processing events, can now be programmed to function automatically. That is, after the first program is initiated, it initiates the second, which in turn could initiate a third and fourth program, and so on. The communicating programs can be located on the same host or distributed among all the host systems in the DDP network.

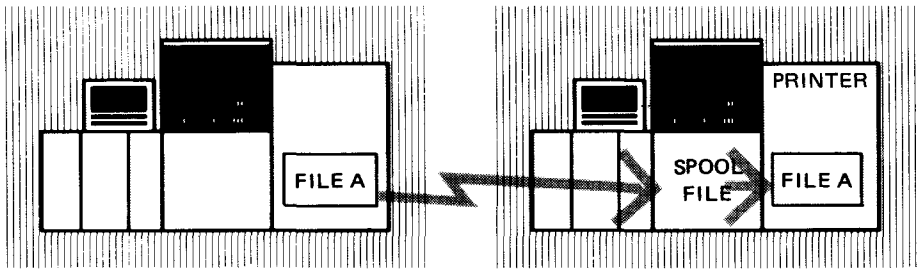
3. Computer Concepts Behind Distributed Data Processing

3.1. USING SPOOLING IN DDP

Spooling is vital to a DDP system because it lets hosts store DDP data or output until devices are available at the destination host. Without spooling, you'd have to make sure that, for instance, any time you expect output back from a remote host, your printer or punch must be free. With spooling, output from a remote host goes to your spool file to wait until the device is available.



DIRECT COPY

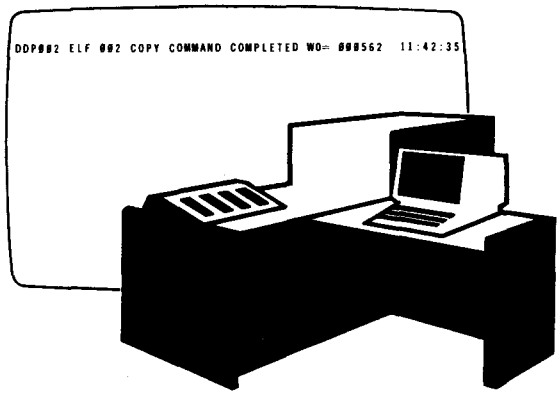


INDIRECT COPY

3.2. DISPLAYING MESSAGES

Display messages tell you about your job. They may tell you that your job has been successfully submitted to the DDP network or that the remote host has completed it. The DDP software displays messages at:

- the terminal, workstation, or system console where you entered the command;
- the log of the host originating the command;
- the system console of the host originating the command;
- the log of the receiving host; or
- the system console of the receiving host.



3.3. AUTOMATIC LOGGING AND ACCOUNTING METHODS WITH DDP

All DDP local and remote activity is automatically logged and sent to the central site printer for your reference. See 7.1.1.

Logging and job accounting for DDP is described in spooling and job accounting concepts and facilities, UP-8869 (current version).

3.4. HOMOGENEOUS VERSUS HETEROGENEOUS SYSTEMS

Computer hosts form a homogeneous DDP system if they use the same operating system. Thus a 90/30 computer and a System 80 computer form a homogeneous system, since both use OS/3. A 90/30 computer and an IBM System 34 computer, however, form a heterogeneous system, since the first uses OS/3 while the second uses a non-OS/3 operating system.

Current SPERRY UNIVAC DDP systems are only homogeneous. Heterogeneous systems will come later. Then you'll be able to connect Series 90 and System 80 computers with computers from other manufacturers.

3.5. COMMUNICATIONS REQUIREMENTS FOR DDP HOSTS

DDP requires electronic communications and Sperry Univac is using its distributed communications architecture (DCA) to implement DDP. In its present form, DDP requires the following communications software products:

- integrated communications access method (ICAM); and
- DCA termination system, or public data network (PDN).

3.6. ENTERING COMMANDS TO A REMOTE HOST OR TO YOUR LOCAL HOST

You can enter DDP commands at the system console, a terminal, or a workstation. Normally, you'll use a terminal or workstation that is not dedicated to DDP.

Some DDP functions take a long time to execute. For instance, you may want to copy a very large file. Or you may want to start a program on a busy host and then have to wait to establish a connection. Fortunately, your terminal is not reserved for DDP during the entire DDP operation. Once you have entered the DDP commands to start your function, you may perform other tasks after the DDP command you entered has been accepted.

DDP commands aren't reserved just for remote tasks. You can also use them to perform local tasks as well. You simply use your own host's identification number for the receiving host. You may not want to do that often, since only a limited number of local tasks can be performed through DDP commands. But if you suddenly need to perform a task such as a job start or a file copy while working with DDP, you can simply use DDP commands to do the job.

4. The Current Products

4.1. OVERVIEW

Three DDP products are available:

- SPERRY UNIVAC DDP transfer facility
- SPERRY UNIVAC DDP file access facility
- SPERRY UNIVAC IMS DDP transaction processor facility

This section presents an overview of these products, describing what they do for you. Part 2 of this manual explains how you can use these facilities.

4.2. DDP TRANSFER FACILITY

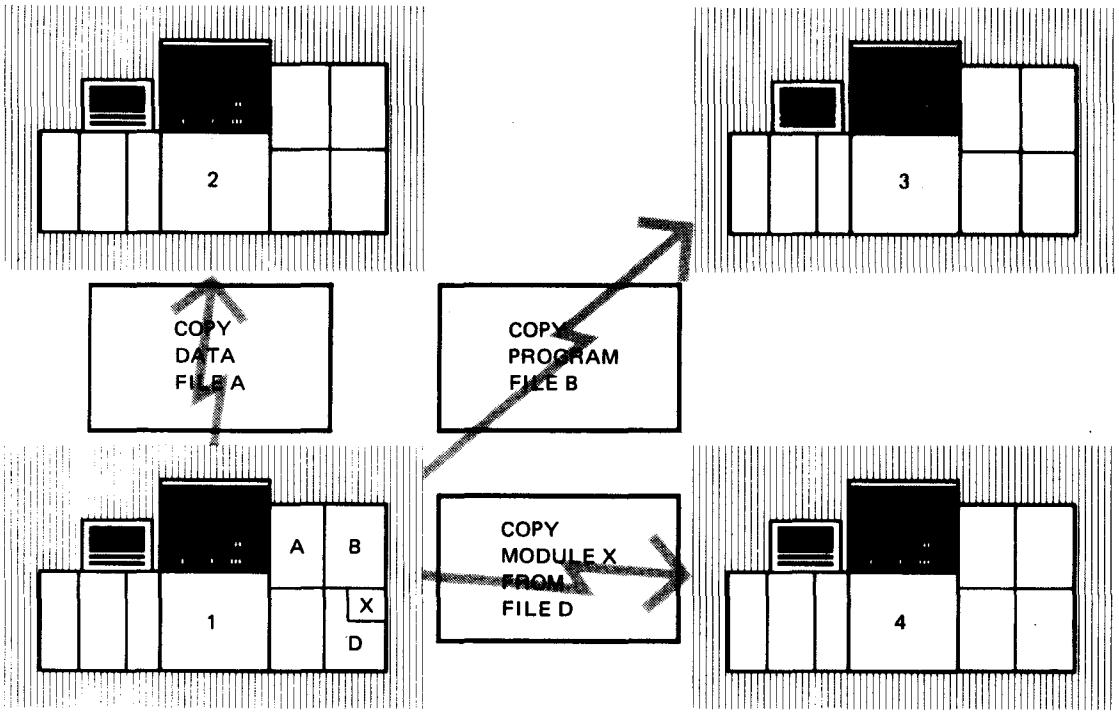
The DDP transfer facility allows you to transfer files and jobs between OS/3 computers. You can also inquire about the availability of a file or the status of a job on a remote OS/3 system. A simple set of English-based commands is all you need to use the file transfer facility.

4.2.1. Sending Files to a Remote Host

You can copy data files, library files, or individual modules from library files and send these files to another host.

You can transfer files three ways:

- From a local system to a remote system
- From a remote system to a local system
- From a remote system to a remote system



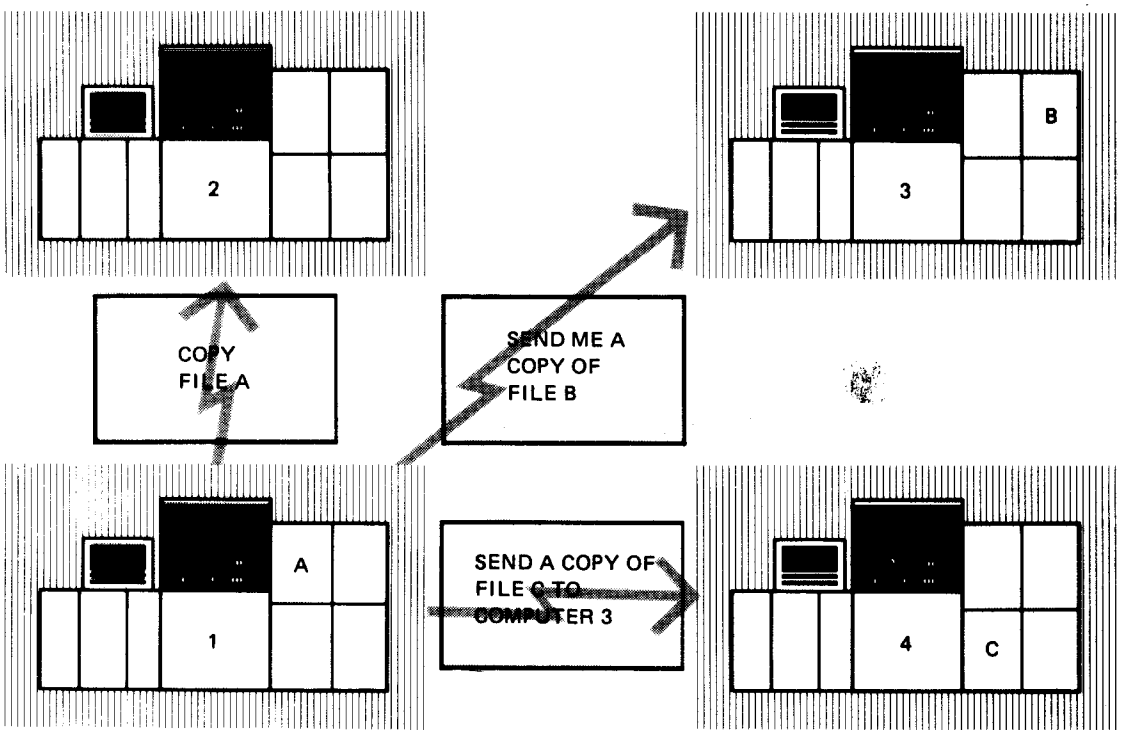
You can:

- duplicate a data base by sending files to a remote host;
- send a library file of programs to a remote host and run the programs later;
- make a new copy of a file that has been updated at another host; and
- send files to a remote host for a special project. For instance, an annual report may require one-time access to data on several hosts. If many separate points of access to those files are needed, the most efficient way to do the job may be to copy all the files onto one host and run the report program there.

4.2.2. Sending Jobs to a Remote Host

In addition to the file copying described in 4.2, you can also:

- send job control streams (a form of program module) from one host to another;
- start a job on a remote host;
- send, receive, and respond to messages from a job on a remote host;
- inquire about the status of your job on a remote host; and
- communicate with the operator of a remote host.



4.2.3. Job Control Language and Device Requirements for Remote Jobs

When you send job control streams to a remote host, you must know details about the capacities and devices of your remote host so you can write your job control stream to conform to its limits.

Job streams must be complete to be sent. And once you send them, they may be performed in any time sequence – not just in the one you originally intended. So don't send jobs to different hosts if the successful completion of one job depends on the prior execution of another. If, for instance, job A sends data to job B, execute both jobs on the same host.

4.2.4. What Happens to Output from Remote Jobs?

What happens to completed output? Normally, it returns automatically to the sending host's spool file. Then it's printed or punched, according to your instructions in the job control stream. In the future, in addition to this automatic return, you will also be able to send the output to a third host or send multiple copies throughout the DDP system.

4.2.5. Functions You Perform with Remote Jobs

Jobs are sent to remote hosts for various reasons. For example you can:

- move a job that's too large for one computer to a more powerful one; or
- you can move a job needing a particular file to the host where the file resides.

For example, you may have a centralized inventory file comprised of data from 10 different sites. Each remote site needs to access or update the file only about once a month. But the site where the file resides updates it daily. It wouldn't make sense to keep copies of the file at each site and have to update them constantly. Instead, it's more efficient to keep just one file. When the remote sites need to access the file, they can submit their jobs through DDP to the site where the file resides.

Section 8 provides more detailed information on this facility.

4.3. DDP FILE ACCESS FACILITY

The DDP file access facility enables you to access and process files residing on remote OS/3 systems and write application programs that can initiate and communicate with one another. The programs can reside on the same or different hosts.

4.3.1. Remote File Processing

You can now access and process remote disk files simply by adding a host identification parameter to the device assignment set for that file. DDP will then connect your program with the remotely located file. No changes are required in your program. Remote file processing eliminates the need for you to copy a remote file on your system before you can access and process it.

4.3.2. Program-To-Program Communication

You can now write BAL programs that can communicate with one another on different OS/3 hosts in your DDP network. You can write primary/surrogate programs that you can use to transfer data from one host to another host, elaborate on the data, and then send it back to the original host.

The program that initiates the conversation is called the *primary* program and the initiated program is called the *surrogate* program. However, the primary and surrogate can reverse roles. There is no limit to the number of status changes that can be requested between a pair of programs. In addition, up to 255 surrogate programs can be initiated by a primary program.

All program to program conversations are initiated and controlled by a set of consolidated data management macroinstructions embedded in both primary and surrogate programs. These macroinstructions enable a program to:

- establish a conversation with another program (DOPEN);
- direct a communication to a particular surrogate program (DMSEL);
- transfer data between programs (DMOUT and DMINP);
- transfer primary/surrogate control between programs (DMCTL); and
- terminate a conversation with a program (DCLOSE).

4.4. IMS DDP TRANSACTION FACILITY

The IMS DDP transaction facility lets you process IMS transactions at a remote OS/3 computer. To use this facility, each OS/3 system must:

1. include the IMS transaction facility in their operating system;
2. define a global ICAM network that supports distributed data processing, including a LOCAP macroinstruction identifying each IMS system that will send or receive remote transactions;
3. configure a multithread IMS; and
4. include a LOCAP section in the configuration for each remote IMS system.

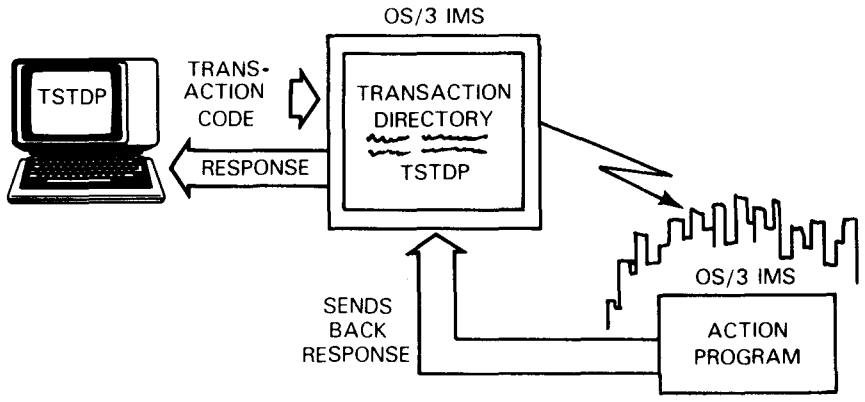
In a DDP transaction, a terminal operator at one IMS system, called the primary IMS, initiates the transaction. The primary IMS, through the transaction facility, routes the transaction to a remote system where a secondary IMS processes the transaction and sends back a response. The remote transaction may be processed by user action programs (written in COBOL, RPG II, basic assembly language) or by UNIQUE (inquiry language). There is little difference between the way action programs process a remote transaction and the way they process a local transaction. Most IMS features are available, including the use of screen format services.

Continuous output, use of auxiliary devices, and switch messages (SEND function) features at the source terminals are unavailable at present.

The three different ways in which IMS can route a transaction to a remote system are explained in the following subsections.

4.4.1. Directory Routing

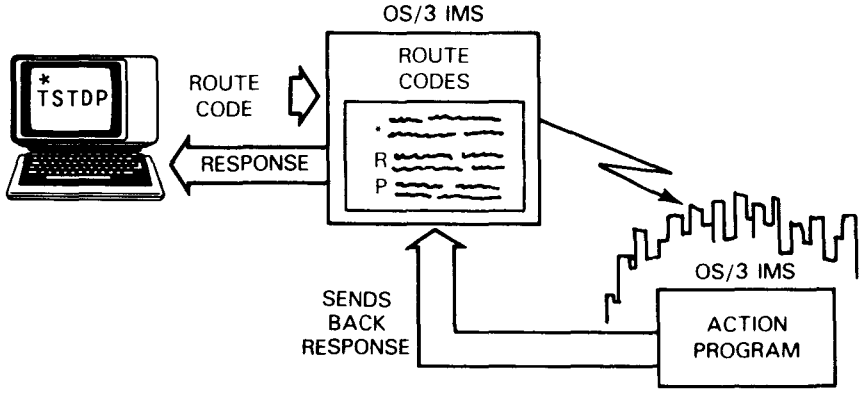
A terminal operator at the primary IMS enters a transaction code that identifies a transaction at a particular remote system. The transaction code and the remote system associated with it are defined to IMS in the configuration. The primary IMS routes the message to the secondary IMS, where action programs or UNIQUE process the transaction. Once the transaction begins, a communications link is established between the terminal operator and the remote system. This allows a dialog transaction consisting of multiple input and output messages.



Directory Routing

4.4.2. Operator Routing

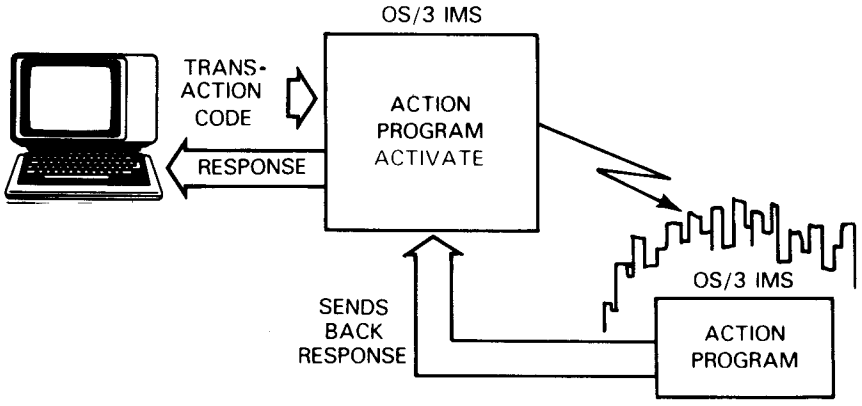
Operator routing is similar to directory routing. In this case, a special character is associated with the remote system, rather than a transaction code. The special character is defined in the IMS configuration or at IMS start-up. When the terminal operator enters a transaction code prefixed by the special character, IMS routes the message to the secondary IMS, where the transaction is processed in the same manner as in directory routing.



Operator Routing

4.4.3. Action Program Routing

The terminal operator enters a transaction code that initiates a transaction at the primary IMS system. A COBOL or basic assembly language (BAL) action program at the primary IMS issues an ACTIVATE function call to IMS, identifies the remote system in its output message header, and generates a message containing a transaction code. IMS routes this message to the remote IMS, where action programs process the transaction and return a message to the originating action program or its successor. The action program at the primary IMS can then return a message to the terminal operator or can issue another ACTIVATE call to initiate another remote transaction.



Action Program Routing

For specific network definition, configurator, and start-up requirements for IMS DDP processing, refer to:

- IMS 90 system support functions user guide and programmer reference, UP-8364 (current version)
- IMS action programming in RPG II user guide, UP-9206 (current version)
- IMS action programming in COBOL and basic assembly language (BAL) user guide, UP-9207 (current version).

5. The Future for DDP

5.1. OVERVIEW

SPERRY UNIVAC DDP products will come to you in stages. You'll have time to use individual products before you need to learn about the next one, and you'll be able to try out products before committing yourself to more.

The following projections are not final product descriptions or commitments to a timetable. They are simply an outline of some current Sperry Univac plans. And they're not the end. DDP will keep growing to bring you products with capabilities beyond these projections.

5.2. LINKS BETWEEN COMPUTERS WITH DIFFERENT OPERATING SYSTEMS

In the future, you'll be able to hook up non-SPERRY UNIVAC computers, such as the IBM System 34, using DDP products. Other systems will also be considered.

5.3. EXPANSION OF ALLOWABLE FILE CODING

DDP currently permits copying only for character-oriented files (EBCDIC and ASCII). Later, this restriction will disappear. You may now use nonencoded characters in your system. (Nonencoded characters are bit configurations within ASCII or EBCDIC that have no standard meaning. You may use these characters to represent special data in your system.) If the file is in ASCII and you're copying it to another computer that uses ASCII, there's no problem. But if either the originating or the destination file, or both, are in EBCDIC, DDP translates all nonencoded characters as blanks. However, later, you may be able to alter the software to transfer them.

PART 2. DISTRIBUTED DATA PROCESSING FACILITIES

6. Statement Conventions for DDP Commands

This manual uses the following conventions to illustrate commands and control statements:

- Code capital letters, parentheses ( ), and punctuation marks (except braces, brackets, and ellipses) exactly as shown. An ellipsis, which is a series of three periods (...), indicates the omission of a number of obvious entries.
- Lowercase letters and terms in commands represent information that you must supply. These lowercase terms may contain hyphens for readability. Lowercase letters and terms in system response messages represent variables the system supplies to you.
- Information within braces { } represents necessary entries. You must choose one.
- Code positional subparameters in the order indicated. Code nonpositional subparameters in any order.
- Insert commas after each positional parameter except the last. When you omit a positional parameter (or a positional subparameter within a series of parameters), retain the comma to indicate the omission. When you omit a trailing positional parameter, omit its associated comma also, even when a keyword parameter follows. Code positional parameters or positional subparameters in the order shown.
- Separate keyword parameters from each other and from positional parameters by spaces. This is different from normal OS/3 format.
- Use underlines as indicated in commands. DDP uses underlines, not hyphens, to join two words in a parameter. This is different from normal OS/3 format.
- The shaded parameter is selected automatically when you omit a keyword parameter or subparameter. In this example, the default for the MODE parameter is DIRECT:

```
MODE={DIRECT  }
     {WAIT    }
     {INDIRECT}
```

- White print on a black background indicates entries in a dialog that the user makes.

- Keyword parameters (nonpositional parameters) are normally entered in the format: keyword=value. This manual shows optional keyword parameters in alphabetical order, for easy reference. However, you may enter them in any order.

- Code zero or more spaces before a DDP command. Conclude each command with one or more spaces except for the last command on a line, which does not require the concluding space.

- Commands may continue on subsequent lines if the last character on the line to be continued is an ampersand (&). In a command, the ampersand may appear only where a space may appear, and DDP treats it as the last character on the line. The ampersand may follow zero or more spaces and must be followed by at least one space before a comment begins.

- The exclamation mark (!) is the comment character and terminates line scanning. It may appear anywhere in a command that a space may appear; it does not have to be used only at the end of a command. If you use it in the middle of a command, however, be sure to put a continuation character (&) before it so that your command may continue on the next line.

  Comments following ampersands do not need the preceding exclamation mark, since the ampersand terminates scanning of that line. After an ampersand or an exclamation mark, scanning begins again on the next line.

- Character strings begin and end with apostrophes. Apostrophes (') within strings must be doubled. Parameter values require string format if they contain a comma (,), space (△), semicolon (;), exclamation mark (!), apostrophe ('), quote (''), pound sign (#), equal sign (=), or ampersand (&). Note that the exclamation mark and ampersand within a quoted string are simply characters within that string. They do not have their usual functions as the comment and the continuation characters.

- When you use a character string as a keyword parameter value, you may continue it on the next line by using the pound sign (#) as a concatenation character. The procedure is:

  1. Break the character string into two parts. For instance, if you have this character string:

     ```
     TO=H001::'MOD1,PERSONNELFILE(READ/WRITE),,S'
     ```

     break it into the two strings:

     ```
     'MOD1,PERSONNELFILE(READ/
     WRITE),,S'
     ```

2. Add the concatenation character (#) and the continuation character (&) to the end of the first part of the string:

```
'MOD1,PERSONNELFILE(READ/'#&
'WRITE),,S'
```

The concatenation character may come immediately after the final character of the first part of the string or immediately before the first character of the second part of the string. For instance, you might have a long statement in a DDP SUBMIT REQUEST command that won't fit on one line. You can continue the command on the next line in either of two ways:

```
DDP SUBMIT REQUEST='RU CGV,,O=VSN999,N=186309,'#&
'T=30' HOST=N825
```

or

```
DDP SUBMIT REQUEST='RU CGV,,O=VSN999,N=186309,'&
#'T=30 HOST=N825
```

Note that:

- the concatenation character (#) may follow zero or more spaces, and may be followed by zero or more spaces; and
- to continue a character string on the next line, you must use both the concatenation character (#) and the ampersand (&). They may be surrounded by any number of blanks and may appear in any order. However, if the concatenation character follows the ampersand, it must be on the next line, since scanning of the first line terminates at the ampersand.

7. DDP Network Requirements

7.1. THE DDP SYSTEM ENVIRONMENT

To use DDP, all OS/3 hosts in your system must have:

- a minimum main storage capacity of 512K bytes (with no resident shared code);
- spooling;
- ICAM generated with the demand mode interface (DMI);
- consolidated data management (CDM); and
- interactive services.

Normally, you also have a terminal or workstation. However, you may enter DDP commands at the system console.

DDP also uses dynamic buffers, but you don't have to worry about their number or size. DDP takes care of that for you automatically. However, some DDP STATUS response messages (8.5.1) show you the number and size of buffers being used. The number varies with the DDP commands being run.

7.1.1. DDP Activity Logging

All DDP local and remote activity is logged in the interactive services log files on the systems involved with the DDP activity. This information is provided at the system console printer after interactive services shutdown time.

Figure 7–1 is a typical log printout of a session showing a series of commands that were entered by the OPERATOR. At the end of the activity, a DDP STATUS COMMAND was entered to provide the status summary shown. Accounting information is given at the end of the log printout.

```
)LOGON                                                                                                    TIME OF EVENT
  LOGON    RWK,BU=NO
  OB IS19 LOGON ACCEPTED AT 13:32:47  ON  82/05/24, REV 08.0.053                                          L 13:32:46
                                                                                                          W 13:32:47
ACTIVITY DURING THIS SESSION
  DDP TALK USER=OPERATOR MESSAGE='START OF DDP TEST'
  0C DDP002 CA1 002 TALK     COMMAND ACCEPTED  WO=000002 13:34:14                                        W 13:34:09
  0D DDP022 FTR 022 TALK     COMMAND COMPLETED WO=000002 13:34:19                                        W 13:34:15
  DDP CREATE FILE=',TEST.FILE,RES' REG=VTOC                                                             W 13:34:20
  0E DDP002 CA1 002 CREATE   COMMAND ACCEPTED  WO=000003 13:35:01                                        W 13:34:57
  0F DDP022 FTR 022 CREATE   COMMAND COMPLETED WO=000003 13:35:05                                        W 13:35:02
  DDP COPY FROM='DDP$$TRM,SY$SCLOD,RES,L' &                                                             W 13:35:05
  0G CONTINUE CURRENT DDP COMMAND                                                                       W 13:35:38
  0H?                                                                                                   W 13:35:42
   0H TO'DDP$$TRM,TEST.FILE,RES,L'                                                                      W 13:35:43
  0J DDP021 CA2 008 COPY     COMMAND ERROR     WO=000C04 13:36:09                                       W 13:36:08
  0K KEYWORD VALUE TO'DDP$$TRM,TEST.FIL IS INVALID                                                      W 13:36:09
  0L DDP021 CA2 020 COPY     COMMAND ERROR     WO=000004 13:36:10                                       W 13:36:10
  0M MISSING TO                   IDENTIFICATION                                                        W 13:36:10
  0N DDP021 CA1 003 COPY     COMMAND ERROR     WO=000C04 13:36:11                                       W 13:36:11
  0P COMMAND REJECTED BECAUSE OF ERROR(S) LISTED ABOVE                                                  W 13:36:11
  DDP COPY FROM='DDP$$TRM,SY$SCLOD,RES,L' &                                                             W 13:36:11
  0Q CONTINUE CURRENT DDP COMMAND                                                                       W 13:36:41
  0A?                                                                                                   W 13:36:45
   0A TO='DDP$$TRM,TEST.FILE,RES,L'                                                                     W 13:36:45
  0B DDP002 CA1 002 COPY     COMMAND ACCEPTED  WO=000005 13:37:07                                       W 13:37:07
  0C DDP022 FTR 022 COPY     COMMAND COMPLETED WO=000005 13:37:27                                       W 13:37:07
  0D PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                           W 13:37:28
  DDP SUBMIT REQUEST='FS ,TEST.FILE,RES,L L0'                                                           W 13:37:30
  0E DDP002 CA1 002 SUBMIT   COMMAND ACCEPTED  WO=000006 13:38:01                                       W 13:37:59
  0F    L-DDP$$TRM DDP COMMON TERMINATION RTNE    81/11/24  02:51                                     W 13:38:02
  0G IS03 FSTATUS FINISHED, 00001 ELEMENTS WERE DISPLAYED                                               W 13:38:11
  0H DDP022 FTR 022 SUBMIT   COMMAND COMPLETED WO=000C06 13:38:14                                       W 13:38:12
  0J PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                           W 13:38:15
  DDP PURGE FILE=',TEST.FILE,RES'                                                                       W 13:38:21
  0K DDP002 CA1 002 PURGE    COMMAND ACCEPTED  WO=000007 13:38:44                                       W 13:38:41
  0L?IS51 ERASING ENTIRE FILE, PROCEED? (Y,N)                                                           W 13:38:45
   0L Y                                                                                                 W 13:38:57
  0M DDP022 FTR 022 PURGE    COMMAND COMPLETED WO=000007 13:39:03                                       W 13:38:59
  0N PREVIOUS USER FUNCTION KEY NOT PROCESSED                                                           W 13:39:03
                                                                                                        W 13:39:06
STATUS OF COMMANDS ISSUED IS REQUESTED
  DDP STATUS USER=RWK
  0P DDP002 CA1 002 STATUS   COMMAND ACCEPTED  WO=000008 13:39:22                                       W 13:39:19
  0Q  USERID   BUFFERS   BUF SIZ   W.O. COUNT                                                          W 13:39:22
  0A  RWK        00002    0009176    000001                                                           W 13:39:25
  0B W.C.#  FUNCTION PRIM SECN STARTED COMPLETE      STATUS                                            W 13:39:25
  0C 000002 TALK     NOD4 NOD4 13:34:17 13:34:19   TERM NORMALLY                                      W 13:39:25
  0D 000003 CREATE   NOD4 NOD4 13:35:02 13:35:05   TERM NORMALLY                                      W 13:39:26
  0E 000005 COPY     NOD4 NOD4 13:37:09 13:37:27   TERM NORMALLY                                      W 13:39:26
  0F 000006 SUBMIT   NOD4 NOD4 13:38:03 13:38:14   TERM NORMALLY                                      W 13:39:27
  0G 000007 PURGE    NOD4 NOD4 13:38:46 13:39:03   TERM NORMALLY                                      W 13:39:27
  0H 000008 STATUS   NOD4 NOD4 13:39:22   :  :     IN PROGRESS                                       W 13:39:27
  0J DDP022 FTR 022 STATUS   COMMAND COMPLETED WO=000008 13:39:28                                       W 13:39:28
LOGOFF
  LOGOFF                                                                                                W 13:39:37
  0K IS73 LOGOFF ACCEPTED AT 13:39:38 ON 82/05/24                                                       W 13:39:38
ACCOUNTING INFORMATION
  AC50  USER-ID=RWK      ACCT NO=          LOGON AT 13:32:46.990  LOGOFF AT 32:46.990  LOGOFF AT 13:39:38.983  CONNECT TIME 00:06:51.993   A 13:39:39
  AC51  CPU TIME USED=00:00:28.211   TASK PRIORITY=01   DATE=82/05/24        DATE=82/05/24        NUMBER OF EXCP'S=00003438                  A 13:39:39
  AC52  NUMBER OF:   COMMANDS=00009 FILES ACCESSED=00004  SVC  CALLS=0C007114  SVC  CALLS=0C007114  TRANSIENT CALLS=00000209                  A 13:39:39
```

Figure 7–1. DDP Activity Log Printout Sample

7.1.2. Host Identification Requirements

Each host in your DDP network has a host identification (host-id). This is the same identification as the label of your DMI LOCAP macroinstruction in your ICAM network definition. You must know the host-id to send files to, or otherwise communicate with, a remote host.

The host-id is one to four alphanumeric characters long. The first character must be alphabetic.

7.1.3. Using the DDP Network

You must always have ICAM and interactive services running to use DDP. So, the first things to do to prepare your system for DDP are:

1. At the system console, enter Cn or Mn to load the appropriate ICAM symbiont,

   where:

   `Cn or Mn`
   Is the name specified on the MCPNAME parameter in the COMMCT phase of SYSGEN, and n is a numeric digit 1 to 9 that identifies the network to be loaded.

   The message ICAM READY should appear.

2. Enter IS at the system console to load interactive services.

3. You are now ready to use the DDP facilities described in the remainder of this part.

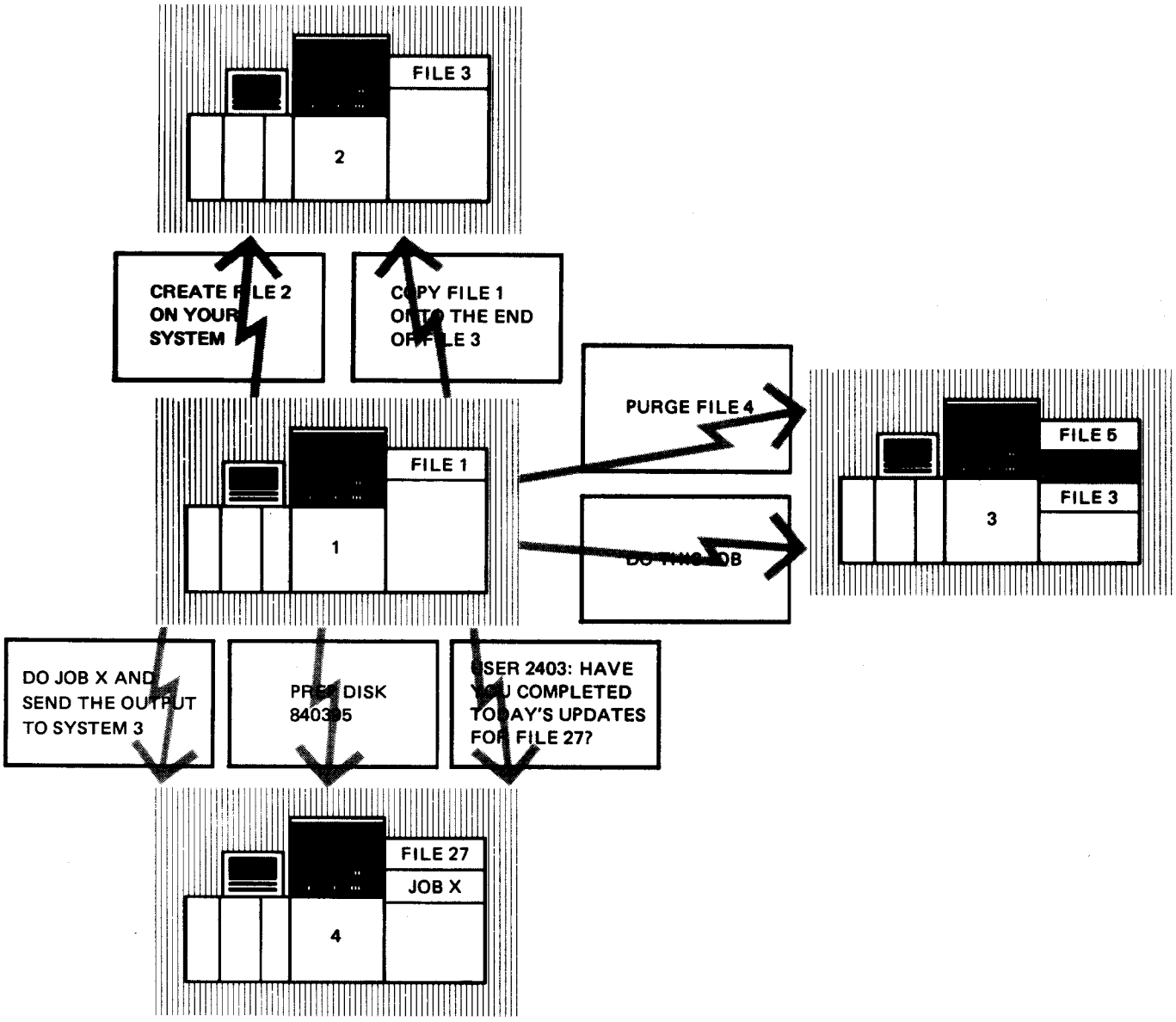
8. The DDP Transfer Facility

8.1. WHAT THE DDP TRANSFER FACILITY DOES FOR YOU

The DDP transfer facility lets you enter commands at your workstation or terminal to:

- create a file on a host;
- copy a file from one host in your system to another;
- remove a file from a host;
- send a job control stream to a host;
- run a job control stream on a host;
- receive the output from your executed job or send the output to another host;
- find out the status of a command, host, job, file, or user in the system;
- send a request (such as an operator command) to a host; and
- send a message to a remote operator or user.

So, you can perform many tasks on another computer using DDP.



But, before you start entering commands, you need some background information.

8.2. USING THE DDP TRANSFER FACILITY

If you're going to enter DDP commands from a terminal, go to 8.2.3. If you're using a workstation, go to 8.2.4. If you're entering DDP commands from the system console go to 8.2.5.

8.2.1. File Identification Requirements

The DDP transfer facility uses a specific form of file identification (file-id). It contains not only the name of the file, but also the name of the volume that contains the file (if the file isn't cataloged), the module name and type, and the read and write passwords. It is a maximum of 74 characters long plus required punctuation marks such as commas, slashes, and parentheses for a total of 81 characters. It is similar to the file-id used in other OS/3 products such as interactive services, and is always expressed as a character string (in apostrophes).

The file-id format is:

```
'[module-name],filename[([read-passwrd]/[write-passwrd])],[vol],[element]'
```

where:

`module-name`
: Is used only when you're sending just one module from a program library file. It is the one to eight alphanumeric characters identifying the module. It must begin with an alphabetic character.

`filename`
: Is always required. It is the same as the name on the // LBL job control statement for the program. For disk files, it is 1 to 44 alphanumeric characters long. For tape files and logical files (spool files), it is 1 to 17 alphanumeric characters long.

`read-passwrd`
: Is a password (one to six alphanumeric characters) enabling you to read the file. If omitted, the system assumes there is no read password on the DDP command. If the file does have a read password that you omit, you won't be able to read the file.

`write-passwrd`
Is a password (one to six alphanumeric characters) enabling you to write to the file. If omitted, the system assumes there is no write password on the DDP command. If the file does have a write password that you omit, you won't be able to write to the file.

`vol`
Is the volume serial number (one to six alphanumeric characters) of the volume on which your file resides. Specify this when the file is not cataloged.

`element`
Is the type of code one to four alphanumeric characters contained in the program module you're sending. In OS/3, we'd normally call this *type* or *module-type*. But since DDP is designed eventually to connect you with non-OS/3 hosts, we're using the word *element*.

The element types are the same ones you're familiar with for all OS/3 library files. SAT files may have the following types:

- S (source code)
- M (macro)
- P (procedure)
- L (load code)
- O (object code)

MIRAM files may have the following types:

- F and FC (screen format modules)
- J (saved job control stream)
- MENU (menu modules)
- HELP (help screen modules)

The default is S (source).

For MIRAM files, you may create your own element type and identify it with one to four characters. (For further information on MIRAM files, see consolidated data management concepts and facilities, UP–8825 (current version).)

Table 8–1 gives some examples of complete file-ids.

Table 8–1. Examples of File-ids

Type of File	Example
A cataloged file with no read and write passwords	',SITE/3'
A cataloged file with read and write passwords	',SITE/3(XX2674/BENNIS)'
A source module from a cataloged file with read and write passwords	'''PROG14,JOBFIL19(XX4982/RILEY)'#& ',,S'''
An uncataloged file with a write password but no read password	',SITE/A(/SMPRO),VSN149'
An uncataloged file with a read password but no write password	',SITE/A&B(READ/),296410'

8.2.2. Requirements for Using DDP Commands within Your Local Host

You may use DDP commands to work with files and jobs entirely within your local host. The DDP transfer facility accepts your local host-id as a valid host-id for all commands. In addition, DDP commands use your local host as a default if you fail to provide a host-id. Thus, if you're working with DDP and must perform a local task, you can perform the task with a DDP command.

8.2.3. Preparing Your Terminal to Enter DDP Commands

If you're entering DDP commands from a terminal rather than from a workstation or system console, you must follow these steps:

1. Run the GUST program (ML$$G1).

2. Sign on the terminal.

Continue with 8.2.4.

8.2.4. Logging On (LOGON)

If you're using a workstation or terminal, enter the LOGON command with your user identification. (You don't need the LOGON statement if you're entering commands from a system console.) The format is:

```
LOGON user-id
```

where:

`user-id`
Is the one to six alphanumeric characters identifying you as a user to the host.

Example:

```
LOGON CHAR
```

If the system accepts your LOGON command, it returns a message stating so.

8.2.5. Entering the DDP Software (DDP)

If your terminal, workstation, or system console is connected to a DDP network (see 1.3 and 3.5), you enter DDP software whenever you issue a DDP command.

8.2.6. Services DDP Performs Automatically

8.2.6.1. DDP Scans and Forwards Your Commands

DDP scans your commands and informs you of all detectable syntax errors. (See the system messages manual, UP–8076 (current version), for a complete list of all DDP messages.) Once the command is correct, DDP accepts it and forwards it to the DDP system.

8.2.6.2. DDP Gives You a Work Order Number to Reference Your Commands

Once DDP accepts your command, it returns a work order number to you using the format:

```
DDP002 CAI 002 ccccccccc COMMAND ACCEPTED WO=nnnnnn time
```

where:

cccccccccc
: Is the name of the command you've just entered.

nnnnnn
: Is the work order number of the command.

For instance, let's say you've just entered the DDP COPY command. DDP responds:

```
DDPØØ2 CAI ØØ2 COPY COMMAND ACCEPTED WO=A48ØØ7 13:46:Ø2
```

Write down the work order number, since error messages use it for identification. If, for example, you enter a command that cannot be completed at the remote host, DDP informs you of the problem by sending you a message such as:

```
DDPØØ1 ELH Ø23 COPY COMMAND ABORTED WO=A48ØØ7 13:48:39
```

The WO=A48107 is the work order number that DDP returned to you at the time you entered your copy command. You may have entered several copy commands; but each has a different work order number, so you can tell them apart.

8.2.6.3. DDP Gives You a Job Name to Reference Your Jobs

When you enter a DDP SUBMIT FILE command, DDP returns a job name to you with which you can cancel the job if you later need to.

DDP renames all jobs submitted using the DDP SUBMIT FILE= command. The job name is eight characters long with the format xxxxyyyy.

where:

xxxx
: Is the host-id of the initiated job.

yyyy
: Is a unique 4-digit number.

The format for the job name message is:

```
DDPØ44 EJS Ø44 jobname JOB SUBMITTED FOR WO=work-order-number time
```

As explained in 8.2.6.2, the work order number identifies the command you've entered. (In this case, it identifies the DDP SUBMIT FILE command.)

8.2.6.4. DDP Sends Messages to You from the Remote Host

As your command or job is executing on the remote host, it produces messages informing you of its status and of any problems encountered. DDP sends you these messages at your terminal or workstation. If you are still logged on the system at the time the message is produced, it is displayed immediately. If you've logged off, DDP displays your messages on the system console.

For a complete list of DDP messages, see the system messages manual, UP–8076 (current version).

8.2.7. No Need to Terminate DDP

You can log off or go on to another task as soon as your command is accepted by DDP.

8.2.8. Entering DDP as a Batch Operation

DDP is designed as an interactive product. But you can use DDP commands in a job control stream. For instance, if it's 4:30 p.m. and you're leaving at 5:00 p.m., you can punch cards with DDP commands on them and submit them in the same way you'd submit any other remote batch job. Your results and messages are printed at your local host.

Appendix F gives an example of DDP commands entered as a remote batch job.

8.3. REMOTE FILE COMMANDS: DDP CREATE, DDP COPY, AND DDP PURGE

There are three remote file commands: DDP CREATE, DDP COPY, and DDP PURGE.

- The DDP CREATE command establishes a file on a host.
- The DDP COPY command copies the contents of a file or modules from a file to an established file on a host.
- The DDP PURGE command removes the file or module and all references to it from the host.

8.3.1. Creating a File (DDP CREATE)

Function and Requirements:

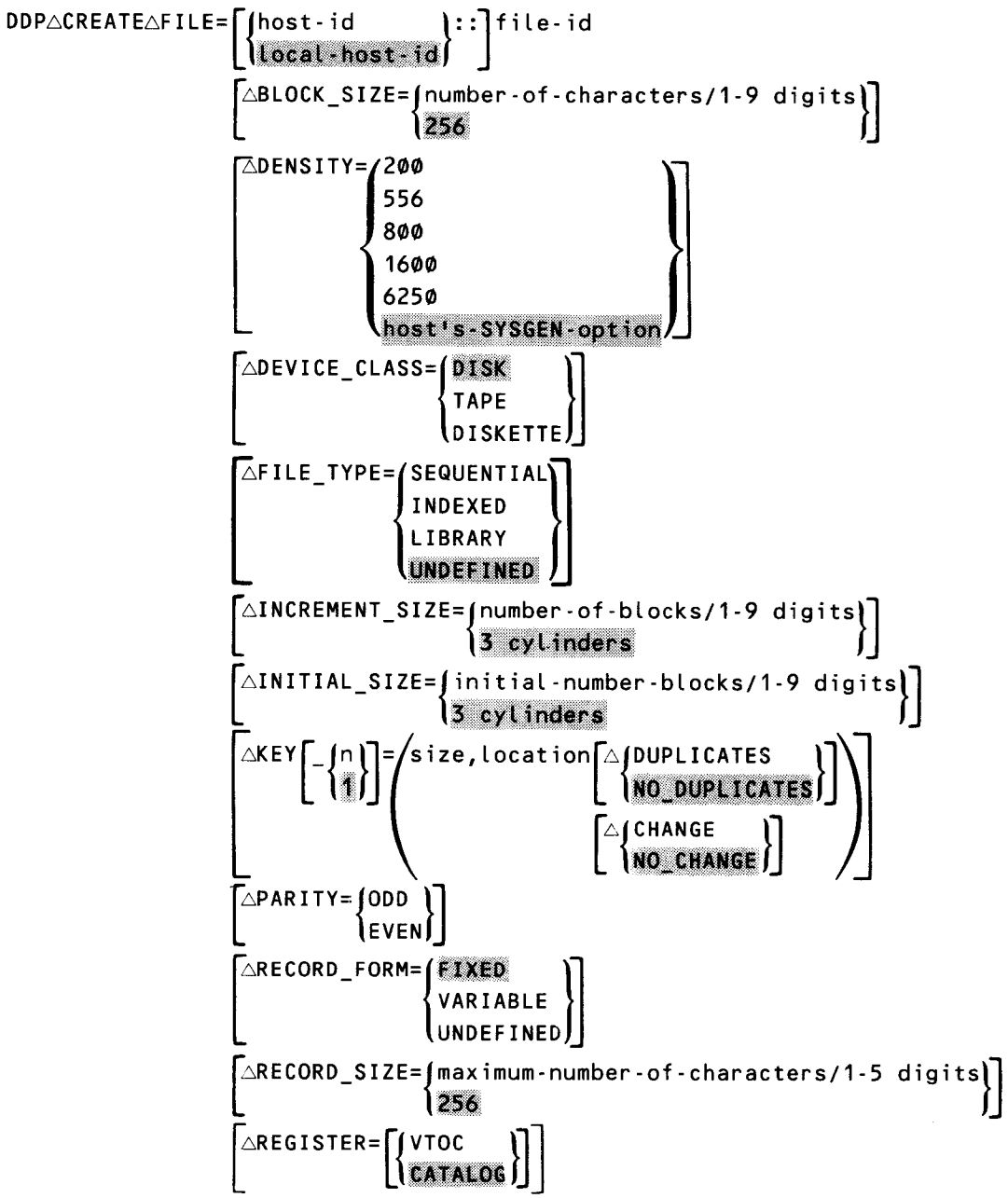
The DDP CREATE command establishes a file on a receiving host. It allocates space for the file and either catalogs the file in your online system catalog or records it in your volume table of contents (VTOC).

The DDP CREATE command is the most complex DDP command. But, once you have the file established, copying and using it are easy.

Follow these requirements for correct file creation:

- The file name cannot already exist on the receiving host. If you try to use a file name already being used, you get an error message. For more information on file names, see 8.2.1.
- Defaults on the file specifications are those of the receiving host. If you do not want these defaults, and if your requirements cannot be met through the DDP CREATE command parameters, create your file by submitting a batch job control stream to the host. (See 8.4.1, the DDP SUBMIT FILE command.)
- The only required parameter is file-id.

Format:



Parameters:

`host-id`
Is one to four alphanumeric characters that name the host you create the file on. If you omit the host-id, the command creates a file on your local system. For more information, see 7.1.2.

`file-id`
Is 1 to 74 alphanumeric characters identifying your file. For more information, see 8.2.1.

`BLOCK_SIZE=`{`number-of-characters/1-9 digits` | **`256`**}
Is the number of characters in a block. If omitted, the default value is 256 characters. If DEVICE_CLASS=DISKETTE, your maximum block size is 256 characters. You must specify BLOCK_SIZE if you specify INITIAL_SIZE.

`DENSITY=`{**`200`** | `556` | `800` | `1600` | `6250` | **`host's-SYSGEN-option`**}
Is the tape density in bytes per inch (bpi). If omitted, the default value depends on the SYSGEN option selected on the receiving host.

`DEVICE_CLASS=`{**`DISK`** | `TAPE` | `DISKETTE`}
Is the device class that will contain your file. The default is DISK.

If you specify DEVICE_CLASS=DISKETTE

- For Series 90

  The diskettes are data set label diskettes and can be used only for MIRAM data files. Diskette library files are not supported. Also, INITIAL_SIZE must be specified, BLOCK_SIZE must be 256, and RECORD_SIZE cannot exceed 512 characters.

- For S/80

  The diskettes can be either data set label diskettes or format label diskettes. Either type of diskette can be used for MIRAM data files. If you intend to use the diskette for library files, then the diskette must be prepped as a format label diskette with the following parameters in the prep job stream.

  FORM=FLB DNSTY=2 RECSZ=256

  The diskette must be double sided and double density.

  The INITIAL_SIZE parameter of the DDP CREATE command must be specified.

If you specify DEVICE_CLASS=TAPE

Tapes can be used for MIRAM data files only.

FILE_TYPE={SEQUENTIAL / INDEXED / LIBRARY / **UNDEFINED**}

Is the type of file being created. The default is UNDEFINED.

A SEQUENTIAL file is one that you access record-by-record, according to the order of the records in the file.

An INDEXED file, such as an IRAM, ISAM, or MIRAM file, is accessed according to one or more index keys.

A LIBRARY file is a collection of program modules.

An UNDEFINED file may be any type of file. When you create a file as UNDEFINED, then copy a file to it, it takes on the characteristics of the originating file. There is one restriction, however: if you're copying a source library file to an UNDEFINED file that doesn't have anything in it, POSITION must equal SOF. (See 8.3.2.)

INCREMENT_SIZE={number-of-blocks/1-9 digits / **3 cylinders**}

Is the number of blocks of storage to be added to the file whenever you extend its size. The number must be greater than zero. The default is 3 cylinders.

If you specify INCREMENT_SIZE, you must also specify BLOCK_SIZE.

INITIAL_SIZE={initial-number-of-blocks/1-9 digits / **3 cylinders**}

Is the number of blocks (1 to 999,999,999) initially allocated to the file. Omit this parameter for tapes. The default is 3 cylinders.

KEY[_{n / **1**}]=(size,location[{**DUPLICATES** / **NO_DUPLICATES**}][{**CHANGE** / **NO_CHANGE**}])

Is used to create an index for the file on the receiving host. You need one of these parameters for each index key (to a maximum of five) that you create. Naturally, you omit this parameter for any nonindexed file.

n

Is the number of the key being created. If omitted, 1 is assumed. You must start with 1 and continue sequentially through 5. Specify all the keys, each in its own KEY_n parameter.

size

Is the number of character positions in the key being created. The maximum size is 80 characters per key.

location

Is the number of character positions into the record where the key begins.

DUPLICATES permits identical values for different keys in the file. NO_DUPLICATES prohibits keys with identical values.

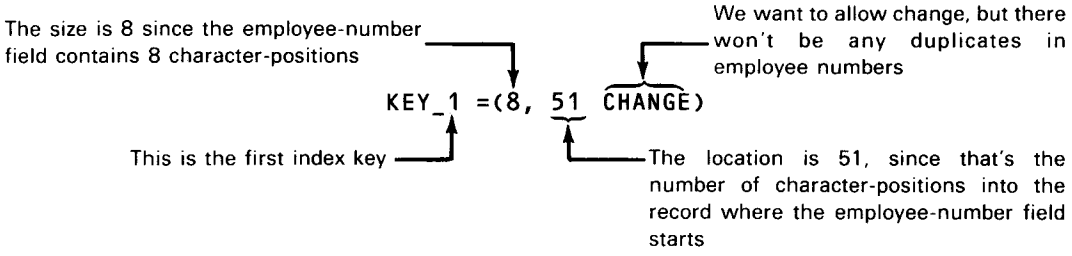
CHANGE permits future file update programs to change the index. NO_CHANGE prohibits change.

Example:

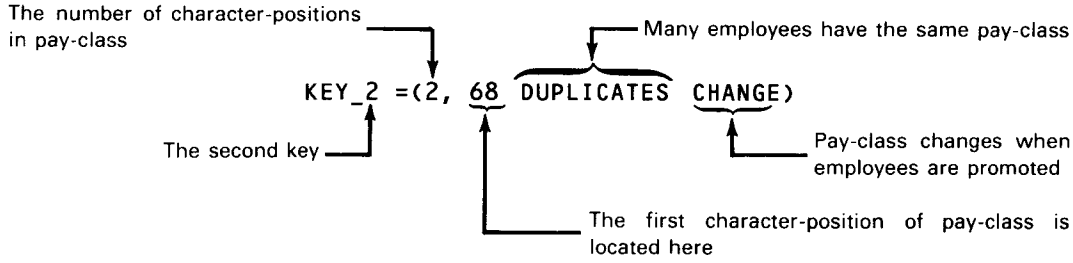
You're creating a file on a remote system and you want to create a 2-key index. Each record in your file has the following fields:

	employee-name	employee-number	social-security-number	pay-class
Number of characters:	50	8	9	2

You want to use employee-number as the first index key and pay-class as the second key. Therefore, in the DDP CREATE command, you first specify:



Following this plan, the other key you need to specify is:



PARITY={ODD / EVEN}

Is the parity system for tape files. Omit for disk and diskette files. If omitted and DEVICE_CLASS=TAPE, the default depends on the SYSGEN option selected at the receiving host.

RECORD_FORM={FIXED / VARIABLE / UNDEFINED}

Specifies whether record length is fixed or variable. The default is FIXED. You may use the UNDEFINED specification only for copying tape files in which the length is immaterial.

RECORD_SIZE={maximum-number-of-characters/1-5 digits / 256}

Is the maximum number of characters in a record. Use this parameter only if FILE_TYPE=SEQUENTIAL, RELATIVE, or INDEXED. If DEVICE_CLASS=DISKETTE, your maximum record size is 256 characters. If omitted, a record size of 256 characters is assumed.

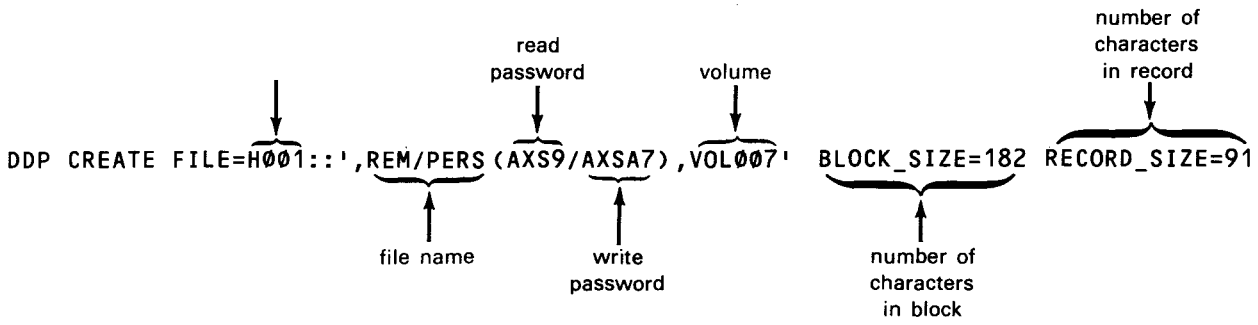
REGISTER={VTOC / CATALOG}

Specifies that the file you create is either cataloged in the catalog file (specify CATALOG) or simply registered in the volume table of contents of the volume you're using (specify VTOC). If omitted, CATALOG is assumed.

Registering the file only in the VTOC takes less time and therefore costs less than cataloging it. However, there are two disadvantages to registering it in the VTOC only. One is that every time you refer to the file, you have to specify the volume name as part of the file-id. The other is that you cannot perform an indirect copy to an uncataloged file. (For additional information on indirect copies, see the DDP COPY command format and parameters, MODE= parameter, 8.3.2.)

Example:

You want to create a cataloged file on host H001, disk volume VOL007, that has read and write passwords, a block size of 182, and a record size of 91. Records are fixed. The file is smaller than three cylinders. You're planning to copy a file into the one you're creating, so you can use the FILE_TYPE default of UNDEFINED. The command is:



8.3.2. Copying Files or Modules (DDP COPY)

Function and Requirements:

With the DDP COPY command function, you can duplicate either an entire file or a module from a file on another system. Follow these requirements to ensure correct copying:

- You can copy only to a file that has already been established. Thus, you might first issue a DDP CREATE command to establish the file, then a DDP COPY command to fill it. However, you can copy to any file on a remote host to which you have access; the file doesn't have to have been established through a DDP command.
- You must know the locations (host-ids) of all files you access.
- See Table 8–2 for the ways records are stored in files of various types.

- See Tables 8–3 and 8–4 for the limitations on originating and destination files during a DDP COPY.
- The two files may be on the same or different hosts. For instance, you may copy from file X on host A to file Y on host A. Or you may copy from file X on host A to file Y on host B.
- You may copy from a remote file to another remote file. In other words, your local host doesn't have to be either the originating or the destination host.
- The record form of the originating and destination files do not have to be the same.
- You may not use a DDP COPY command to alter the element-type of a module. Its element type remains the same in the destination file as it was in the originating file.
- You can use a DDP COPY command to alter a user-specified MIRAM library element type.

Format:

```
DDP COPY△FROM=[{originating-host-id}::]originating-file-id
                {local-host-id     }
        △TO=[{destination-host-id}::]destination-file-id
             {local-host-id     }
            [△ELEMENT_TYPE={SYMBOLIC    }]
                           {RELOCATABLE }
                           {ABSOLUTE    }
                           {MACRO       }
                           {PROC        }
                           {COMPILED_JOB}
                           {SCREEN_FORMAT}
            [△KEY[-{n}]=(size,location[△{DUPLICATES   }][△{CHANGE   }])
                   {1}                 {NO_DUPLICATES}   {NO_CHANGE}
            [△MODE={DIRECT  }]
                   {WAIT    }
                   {INDIRECT}
            [△POSITION={EOF}]
                       {SOF}
            [△TRANSLATE={ASCII }]
                        {EBCDIC}
                        {NONE  }
```

Parameters:

`originating-host-id`
Is one to four alphanumeric characters naming the originating host. If omitted, DDP assumes the originating file is on your local host. For more information, see 7.1.2.

`originating-file-id`
Is 1 to 74 alphanumeric characters identifying the input file. For more information, see 8.2.1.

If your originating file is a LIBRARY file, its element type must be the same as the element type of the destination file.

NOTE:

Table 8–3 shows restrictions on the DDP COPY command for some types of originating files.

`destination-host-id`
Is one to four alphanumeric characters naming the host to receive the data. If omitted, DDP assumes the destination file is on your local host. For more information, see 7.1.2.

`destination-file-id`
Is 1 to 74 alphanumeric characters identifying your file. For more information, see 8.2.1.

If the destination file is a LIBRARY file, its element type must be the same as the element type of the originating file.

NOTE:

Table 8–4 shows some restrictions on the DDP COPY command for various types of destination files.

```
ELEMENT_TYPE=(SYMBOLIC      )
              |RELOCATABLE  |
              |ABSOLUTE     |
              |MACRO        |
              |PROC         |
              |COMPILED_JOB |
              |SCREEN_FORMAT|
```

Is the element type of the module. Specify this only when you are copying a module from a library rather than the entire file. You may omit this parameter if your file-id contains the element type. If you decide to specify the element type using this parameter, you must use this special vocabulary for element types:

SYMBOLIC: source code

RELOCATABLE: object code

ABSOLUTE: an executable program (load code)

MACRO: parts of assembler programs in which values may be substituted (macros)

PROC: parts of programs in any language in which values may be substituted (jproc or source format)

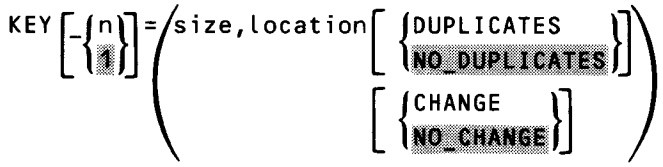
COMPILED_JOB: compiled job streams with expanded jprocs

SCREEN_FORMAT: screen formats

The reason DDP uses this vocabulary is that we've designed it eventually to link OS/3 with other systems.

In library files, the element types of the originating and destination files must match.

If this parameter is omitted, SYMBOLIC is assumed.



Is used to change an index for the file on the receiving host. You need one of these parameters for each index key in the record (to a maximum of five), even if you change only one. Naturally, you omit this parameter for any nonindexed file.

n

Is the number of the key being changed. If omitted, 1 is assumed. You must start with 1 and continue sequentially through 5. Specify all the keys, each in its own KEY_n parameter, even though you may be changing only one of them.

size

Is the number of character positions in the key being changed. The maximum size is 80 characters per key.

location
Is the number of character positions into the record where the key begins.

DUPLICATES permits identical values for different keys in the file. NO_DUPLICATES prohibits keys with identical values.

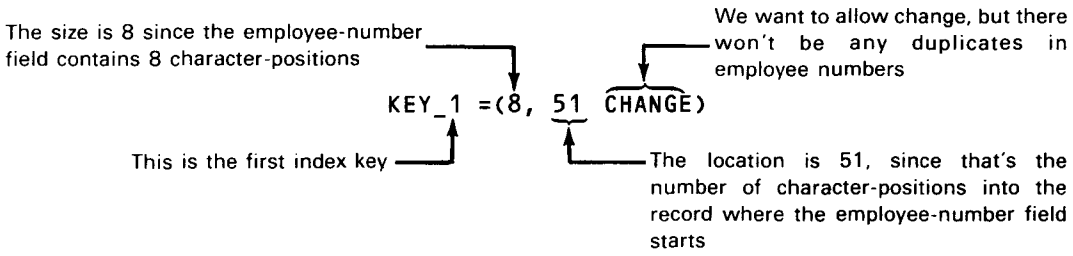
CHANGE permits future file update programs to change the index. NO_CHANGE prohibits change.

Example:

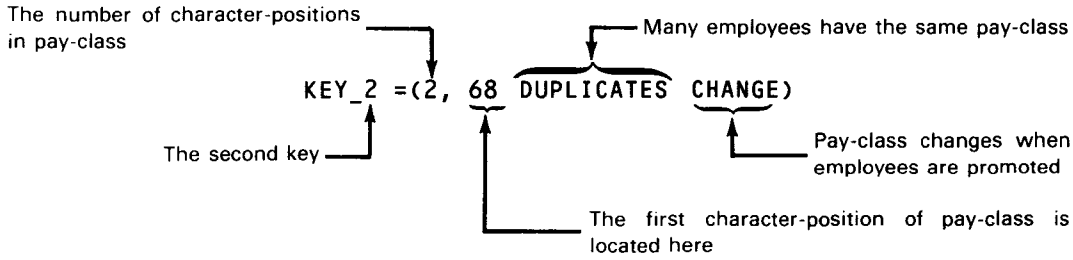
You're copying a nonindexed file to a remote system and you want to create a 2-key index. Each record in your file has the following fields:

	employee-name	employee-number	social-security-number	pay-class
Number of characters:	50	8	9	2

You want to use employee-number as the first index key and pay-class as the second key. Therefore, in the CHANGE command, you first specify:



Following this same plan, the other key you need to specify is:



MODE={DIRECT / WAIT / INDIRECT}

MODE=DIRECT means that the device or medium needed to copy the file at the destination host must be immediately available. If not, the command aborts. DIRECT is the default.

MODE=WAIT means that either the destination file or the originating file is not immediately available, the system should hold the command until both the originating and destination file are free and then execute the DDP COPY. In this case, of course, the COMMAND COMPLETED message won't appear until the devices have been freed and the copy performed. Note, however, that each host has a time limit for which it will hold any command. If this time limit elapses and the devices are still not free, DDP aborts your command.

MODE=INDIRECT permits the facility to build a temporary file at the destination host, if necessary, to store the file copy for later automatic transfer to the proper device. If it is possible to perform a DIRECT copy, however, the system will do so. You may not specify INDIRECT unless your destination file is cataloged. MODE=INDIRECT is only available for SAT or MIRAM library files.

POSITION=$\left\{\begin{matrix}\underline{\text{EOF}} \\ \text{SOF}\end{matrix}\right\}$

Specifies overwriting or extending of the destination file. POSITION=EOF (end of file) means that a copy of the originating file is appended to the end of the destination file. POSITION=SOF (start of file) means a copy of the originating file overwrites the destination file, and all previous contents of the destination file are lost. The default is EOF.

If you specify EOF, then:

- The destination file can't be an UNDEFINED file that's empty.
- The block and record sizes of the originating and destination files must match.
- Record storage is as shown in Table 8–2.

Table 8–2. How Files Store Additional Records at the End of File

If the destination file is this type:	Then records are added to the destination file:
INDEXED	by record key
LIBRARY	with new modules overwriting those with the same name and additional modules added to the end of the file
SEQUENTIAL	in the order transferred, following the last record of the destination file.

If you specify SOF, the destination file may have any file type, because it's completely overwritten with the originating file, whose type it adopts. RELATIVE files store records in successive relative record positions, beginning with the first.

Tables 8–3 and 8–4 give additional information on the use of the POSITION parameter.

TRANSLATE=$\left\{\begin{matrix}\text{ASCII}\\ \text{EBCDIC}\\ \text{NONE}\end{matrix}\right\}$

Indicates the character code you want the file translated to as it arrives at the destination host. If omitted, EBCDIC is assumed.

If you send a file to a host using a different encoding system but you do not want the file to be translated to the host's encoding system, specify NONE. NONE means that the destination host holds the file in the original code. It can transfer the file but cannot access it to perform useful work.

Table 8–3. DDP COPY Restrictions for Originating Files

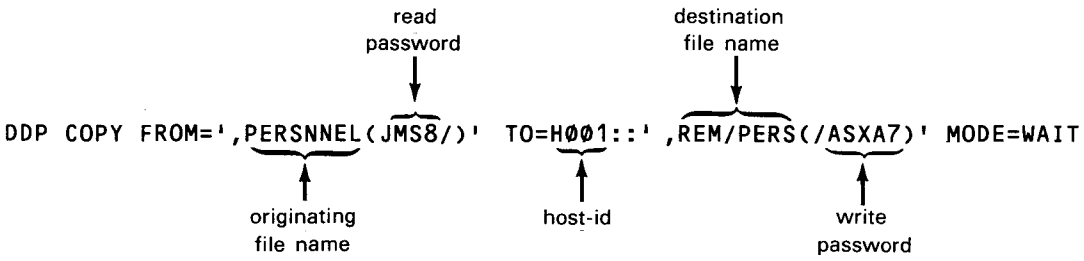
If the originating file type is:	And POSITION=	And you also specify:	Then:
INDEXED	EOF	an INDEXED destination file	keys of both originating and destination files must be identical
LIBRARY	EOF	that the complete library is to be copied, or the file-id contains an element type but no module name	destination file must be LIBRARY and the TO= parameter must not contain a module name
LIBRARY	EOF	module name	destination file may be LIBRARY (with or without module name specified), SEQUENTIAL, or INDEXED
SEQUENTIAL or INDEXED	EOF		destination file may be LIBRARY, SEQUENTIAL, or INDEXED
UNDEFINED	either		the file is treated as a data file
UNDEFINED with nothing in it	either		DDP COPY command is aborted

Table 8–4. DDP COPY Restrictions for Destination Files

If the destination file type is:	And POSITION=	And you also specify:	Then:
INDEXED	EOF	an INDEXED originating file	keys of both files must be identical
LIBRARY	either	module name	originating file must be LIBRARY
LIBRARY	either		POSITION parameter is ignored. Modules from the originating file with the same name as destination-file modules overwrite those modules. Other modules are added to the end of the file.
LIBRARY	EOF	module name and an originating file that is not a LIBRARY file	the resulting module at the destination file is source (symbolic)
UNDEFINED with nothing in it			POSITION must be SOF. If POSITION=EOF, the command is aborted.

Example:

You want to copy file PERSNNEL into the blank cataloged file REM/PERS on host H001, which uses EBCDIC. If a direct connection isn't possible, you want the system to hold the command until it is. The command is:



8.3.3. Purging Files or Modules (DDP PURGE)

Function and Requirements:

The DDP PURGE command physically removes a file or module and all references to it from a host. Follow these instructions to ensure proper purging of your file:

- You may purge only existing files or modules. A DDP PURGE command for a nonexistent file results in an error.
- If the file or module has passwords, you must know them to purge it. If the file has both read and write passwords, you must specify both to purge the file.
- If the file is cataloged, the DDP PURGE command will decatalog the file.

Format:

```
DDP△PURGE△FILE=[{host-id      }::]file-id
               [{local-host-id}   ]
```

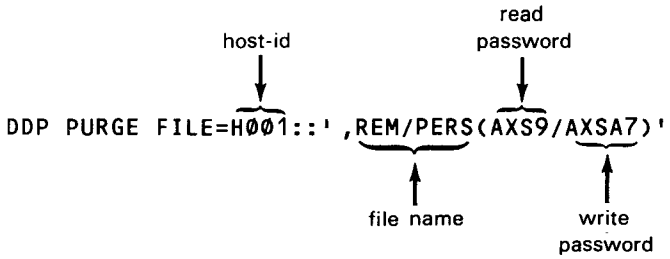
Parameters:

`host-id`
Is one to four alphanumeric characters naming the host you are purging the file or module from. If omitted, DDP assumes the file is on your local system. For more information, see 7.1.2.

`file-id`
Is 1 to 74 alphanumeric characters identifying the file to be purged. For more information, see 8.2.1.

Example:

You want to purge cataloged file REM/PERS on host H001. The command is:



8.4. REMOTE JOB COMMANDS: DDP SUBMIT FILE, DDP CANCEL, AND DDP SUBMIT REQUEST

There are three DDP job commands: DDP SUBMIT FILE, DDP CANCEL, and DDP SUBMIT REQUEST.

- The DDP SUBMIT FILE command sends a file of job control streams to a host for execution. You also use it to initiate a file of job control streams already at the host or to bring a job control stream to your local host for execution.
- The DDP CANCEL command terminates a job that you've submitted.
- The DDP SUBMIT REQUEST command sends an operator command to a remote host.

8.4.1. Submitting Files and Modules for Execution (DDP SUBMIT FILE)

Function and Requirements:

The DDP SUBMIT FILE command sends a file of job control streams to a host for execution. You also use it to initiate a file of job control streams already at the host or to bring a job control stream to your local host for execution. The receiving host returns the output to you, sends it to another host, or retains it.

Follow these instructions to ensure proper functioning of the DDP SUBMIT FILE command:

- The module or file you submit must contain complete job control streams.
- You may submit only SYMBOLIC (source code) or COMPILED_JOB (compiled jobs with expanded jprocs) files and modules.
- Output can be routed to any host in the network, but there must be a direct connection from the host on which the jobs were executed and the host to where the output is directed.
- The destination host must have all the resources required by your job control stream.
- The job stream file may be anywhere in your DDP network. It does not have to be at the destination host or at your local host. Specify its location in the FILE= parameter. The DDP SUBMIT FILE command sends the file of job control streams to the receiving host.

Format:

```
DDP△SUBMIT△FILE=[{originating-host-id}::]file-id
                [{local-host-id     }]
        [△ELEMENT_TYPE={SYMBOLIC    }]
                       {COMPILED_JOB}
        [△HOST={destination-host-id}]
               {local-host-id     }
```

Parameters:

originating-host-id
: Is one to four alphanumeric characters naming the job control stream file location. If omitted, the system assumes the file is on your local system. For more information, see 7.1.2.

file-id
: Is 1 to 74 alphanumeric characters identifying the input file. For more information, see 8.2.1.

  The file-id in the DDP SUBMIT FILE command may be either a sequential data file, which has no module name, or a module in a library file, in which case the module name must be present. (Normally you won't be submitting your entire library for execution.) In either case, the sequential file or the module may contain more than one job name. The destination host runs all jobs submitted in this file or module.

```
ELEMENT_TYPE={SYMBOLIC    }
             {COMPILED_JOB}
```

: Is the element (module) type of the submitted file. SYMBOLIC indicates source code. COMPILED_JOB indicates compiled job streams with expanded jprocs. You may omit this parameter if the element type is SYMBOLIC (the default), or if the element type is specified in the file name. Normally, you won't be using COMPILED_JOB.

```
HOST={destination-host-id}
     {local-host-id     }
```

: Is one to four alphanumeric characters naming the host you submit the file to. If omitted, the command assumes the file is to be submitted to your local system. For more information, see 7.1.2.

NOTE:

DDP renames jobs with the format xxxxyyyy

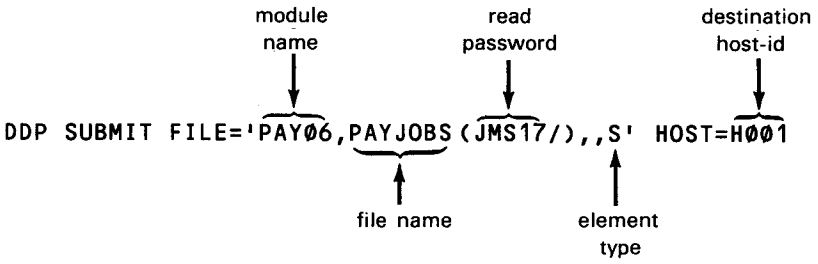
where:

xxxx
Is the node that initiated the DDP SUBMIT FILE command.

yyyy
Is a unique 4-digit number.

Example:

You want to submit a job to remote host H001. You've already stored the job at your local host in module PAY06 of library file PAYJOBS. The command is:



Alternate forms of this command are:

```
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/)' ELEMENT_TYPE=SYMBOLIC HOST=H001
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/)' HOST=H001
```

Response Messages:

As explained in 8.2.6.3. when DDP accepts your DDP SUBMIT FILE command, it returns a job name to you as part of the message:

```
DDP044 JNR 044 jobname JOB SUBMITTED FOR WO=nnnnnn time
```

For example:

```
DDP044 JNR 044 AAAA0001 JOB SUBMITTED FOR WO=000012 13:47:56
```

Write down the job name and its work order number. (You may want to use the log sheet in Appendix G to do this.) You need your job name to cancel the job or to inquire as to its status.

8.4.2. Cancelling a Job or Command (DDP CANCEL)

Function and Requirements:

The DDP CANCEL command terminates an executing or backlogged job or a command executing on a host. Follow these instructions to ensure proper cancellation of your job or command:

- You must specify the host on which you are running the job or command unless it is your own local host.
- You may cancel only jobs or commands you have submitted under your user-id.

Format:

```
DDP△CANCEL△JOB=[{host-id       }::]jobname
                {local-host-id }
      [△OUTPUT={DISCARD}]
               {DELIVER}
      [△COMMAND={host-id       }::]work-order-number
                {local-host-id }
```

Parameters:

`host-id`

Is one to four alphanumeric characters naming the host that the job is executing on. If the host-id is omitted, the system assumes the job is executing on your local system. For more information, see 7.1.2.

`jobname`

Is eight alphanumeric characters naming the job that the system returns to you on the successful completion of your DDP SUBMIT command. (See 8.2.6.3 and 8.5.1 for more information on the jobname parameter.)

```
OUTPUT={DISCARD}
       {DELIVER}
```

Specifies what you want done with the spooled output from the job you cancel. DISCARD is the default and means the system erases the spooled, completed output files. DISCARD does not apply to job output files in process. DELIVER means that the spooled output files go to the host that would have received the completed output.

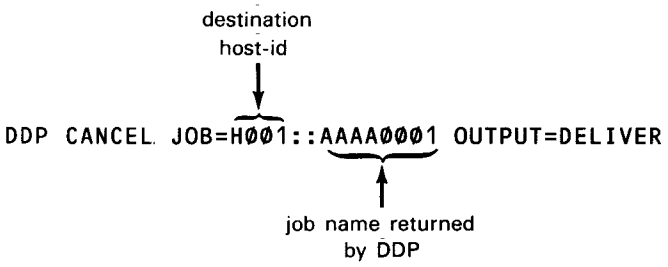
```
COMMAND={host-id       }
        {local-host-id }
```

Specifies the command that you entered but now wish to cancel.

work-order-number
Is the one to six alphanumeric work-order-number that was displayed to you when command was accepted.

Example:

You want to cancel job AAAA0001 on host H001 because there's an error in the program. If there's any output, you want it sent to you, since that might help you diagnose the problem. The command is:



8.4.3. Submitting a Statement for Execution (DDP SUBMIT REQUEST)

Function and Requirements:

The DDP SUBMIT REQUEST command lets you send a statement to a remote host. The statement is an instruction for the remote host to perform some task, such as an operator command to run a utility program. Like anything else you submit to a remote host, it must conform to the configuration of the remote host.

When you use DDP SUBMIT REQUEST, DDP doesn't check the contents of the statement at all. You are responsible for sending a correct, executable statement to the remote host. Also, DDP can't inform you whether or not your statement was executed at the remote host. If, for instance, you send a request to prep a disk, DDP can tell you that your statement was delivered successfully; but it can't tell you if the disk was actually prepped. However, you do receive any messages generated by the command submitted.

Format:

```
DDP△SUBMIT△REQUEST=statement [HOST={host-id      }]
                              {local-host-id}
```

Parameters:

`statement`

Is the system command or message being submitted. The statement must be enclosed in apostrophes (') if it contains any of the following: comma (,), space (△), semicolon (;), exclamation mark (!), apostrophe ('), quote (''), pound sign (#), equal sign (=), or ampersand (&). In addition, you must double any apostrophes within statements enclosed in apostrophes.

Most statements you'll be sending will be character strings.

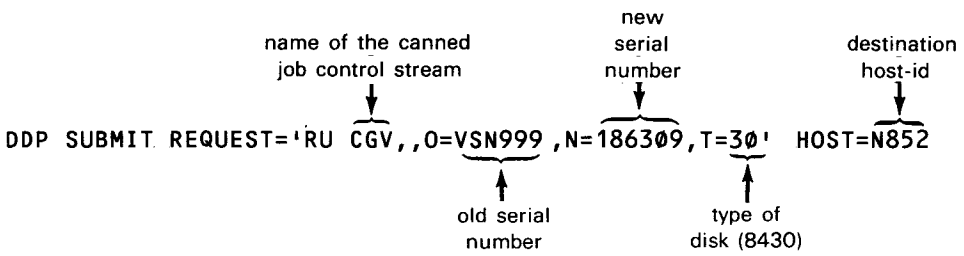
`host-id`

Is one to four alphanumeric characters naming the destination host. If you omit this parameter, DDP submits the request to your local host. For more information, see 7.1.2.

Example:

The purpose of the DDP SUBMIT REQUEST command is to save you time when you want to do routine tasks on the remote host. Let's say, for instance, that you're just starting out in your connection with a remote host, whose host-id is N852. You're going to be sending a large number of files to that host, so you need your own disk. The remote host has an 8430 disk that they've already prepped, using the volume serial number VSN999. Since this will be your disk, you want to put your own serial number on it, 186309.

One way to change the serial number would be to establish a file on your local host, put the job control stream to change the number into it, then enter the DDP SUBMIT FILE command. But that's three steps. A much easier way would be to use the DDP SUBMIT REQUEST command with the canned job control stream that Sperry Univac supplies to change volume numbers. All you'd have to do is enter:



The number is changed for you.

NOTES:

1. *You can get more information about using the RU CGV command to change a volume serial number by consulting the Series 90 system service programs (SSP) user guide, UP-8062 (current version) or the System 80 system service programs (SSP) user guide, UP-8841 (current version).*

2. *You cannot specify the following commands with the DDP SUBMIT REQUEST command: DISPLAY, DELETE, BREAKPOINT, FILE, IN, SU, TU, and PD. An error message is displayed if any of these commands are sent.*

8.5. INFORMATION COMMANDS: DDP STATUS AND DDP TALK

There are two DDP commands that send information back and forth between hosts without involving jobs or files. They are:

- DDP STATUS, which tells you about a remote host, file, or user, or about a command or job you've sent; and
- DDP TALK, which allows you to send a message to a remote user or operator.

8.5.1. Finding Out the Status of a DDP Command, Host, User, File or Job (DDP STATUS)

Function and Requirements:

You can find out the status of a command you entered, a host in your system, a user, a file, or a job, by using the DDP STATUS command. You might want to use this command, for instance, to see if a file your job needs is currently available or to see if a job you've submitted to a remote host via the DDP SUBMIT REQUEST command has executed successfully.

This command provides statistics only for those commands that you entered for that host. Do not use this command as the object of another DDP STATUS command. For systems with main storage capacity of 512 KB, DDP should be used in a dedicated system environment and commands should only be issued sequentially (a command must be processed before another can be issued).

Format:

```
DDP△STATUS△{COMMAND=work-order-number                                    }
           {HOST=host-id                                                 }
           {JOB=[{host-id      }::]jobname                               }
           {    [{local-host-id}  ]                                      }
           {file=[{host-id      }::]file-id[keyword parameter]           }
           {     [{local-host-id}  ]                                     }
           {USER=[{host-id      }::]                                     }
           {     [{local-host-id}  ]                                     }
```

Parameters:

`COMMAND=work-order-number`
Displays the status of a command you entered. For types of information provided, see 8.5.1.1.1.

`work-order-number`
Is the one to six alphanumeric work-order-number that is displayed to you when your command is accepted. Note and use this number when enquiring about a command you have issued. For more information, see 8.2.6.2.

`HOST=host-id`
Informs you whether users on a host are active or not. For the types of information provided, see 8.5.1.1.2.

`host-id`
Is one to four alphanumeric characters naming the host whose status you want. If you do not specify host-id, the local host is assumed. For more information, see 7.1.2.

`JOB=[{host-id      }::]jobname`
`     [{local-host-id}  ]`
Tells you the status of a job such as COMPLETED, BEING PROCESSED, or IN SCHEDULER.

`host-id`
Is one to four alphanumeric characters naming the host where the job was sent. DDP assumes the job is on the local host if the host-id is not specified. For more information, see 7.1.2.

`jobname`
Is one to eight alphanumeric characters naming the job that the DDP SUBMIT FILE command returned to you. For more information, see 8.2.6.3.

`local-host-id`
Is one to four alphanumeric characters naming the host you are using at your site. For more information, see 8.5.1.1.3.

`FILE=[{host-id | local-host-id}::]file-id[keyword parameter]`
Provides information on your data and library files. For the types of information provided, see 8.5.1.1.4.

`host-id`
Is one to four alphanumeric characters naming the host where the file resides. If you omit this name, DDP assumes the file is on your local host. For more information, see 7.1.2.

`file-id`
Is 1 to 74 alphanumeric characters identifying your file. This file-id can be a library file or data file. For more information see 8.2.1.

`keyword parameter`
Is the keyword parameter, if any, associated with your file.

`USER=[{host-id | local-host-id}::]user-id`
Lists a table of DDP functions for a particular user.

If you're working from a system console, you may request the status of any user in the system. If you're working from a terminal or workstation, though, you may request only your own status. (That is, you may enter this command only with your own user-id.)

`host-id`
Is one to four alphanumeric characters naming the host where the user is located. If omitted, DDP assumes the user is on your local host. For more information see 7.1.2.

`user-id`
Is the one to six alphanumeric characters identifying the user. This is the same user-id a user would use in the LOGON command.

8.5.1.1. DDP STATUS Command Information Summary

The DDP STATUS command provides the following information at interactive services shut-down time for the parameters listed.

8.5.1.1.1. COMMAND= parameter

This parameter displays:

- Originating (primary) host
- Destination (secondary) host
- Time the command was accepted
- Time the command was completed
- Status information such as:

  BEING PROCESSED

  IN SCHEDULER

  COMPLETED

Example: Response to DDP STATUS COMMAND:

```
W.O. # FUNCTION PRIM SECN STARTED COMPLETE STATUS
```

where:

`W.O. #`
Is the work order number of the command whose status you're requesting. For additional information about work order numbers, see 8.2.6.2.

`FUNCTION`
Is the name of the command whose status you're requesting, such as DDP COPY or DDP PURGE.

`PRIM`
Is the host-id of the primary host involved with a command. If the command involves an originating file, the primary host is the host that has that file. Otherwise, the primary host is the host originating this command.

`SECN`
Is this host-id of the secondary host involved with a command. If the command involves an originating file, the secondary host is the destination host of that file. Otherwise, the secondary host is the destination host for this command.

`STARTED`
Is the time a command was sent.

COMPLETE
Is the time a command was completed, if it was.

STATUS
Specifies status information.

Log printout sample:

```
DDP STATUS COMMAND=5
ØB DDPØØ2 CA1 ØØ2 STATUS  COMMAND ACCEPTED  WO=ØØØØ24 12:23:35
ØC W.C.#  FUNCTION PRIM SECN STARTED COMPLETE      STATUS
ØD ØØØØØ5 COPY     NOD4 NOD2 12:Ø9:14 12:Ø9:32  BEING PROCESSED
ØE DDPØ22 FTR Ø22 STATUS   COMMAND COMPLETED WO=ØØØØ24 12:23:42
BR CN
```

8.5.1.1.2. HOST= parameter

This parameter displays:

- Number of local DDP users
- Number of remote DDP users
- Number of buffers in use by DDP to service the users
- Total number of bytes of memory used by the buffers identified in the preceding item.

Example: Response to DDP STATUS HOST:

```
USERS   LOCAL   REMOTE   BUFFERS   BUF SIZE   W. O. COUNT
-----   -----   ------   -------   --------   -----------
USERID   HOST   STATUS
------   ----   ------
```

where:

USERS
Is the total number of users on the DDP network.

LOCAL
Is the number of local users on the DDP network.

REMOTE
Is the number of remote users on the DDP network.

BUFFERS
Is the number of buffers being used.

BUF SIZE
Is the size of the buffers used.

W. O. COUNT
Is the number of active work orders being processed for this user.

USERID
Lists the user-ids of all valid DDP users on the host.

HOST
Is the host-id of the host issuing the STATUS command.

STATUS
Is either ACTIVE or INACTIVE.

Log printout sample:

```
DDP STATUS HOST=NOD2
ØK DDPØØ2 CA1 Ø02 STATUS   COMMAND ACCEPTED  WO=ØØØØ27 12:25:51
ØL  USERS  LOCAL  REMOTE  BUFFERS  BUF SIZ  W.O. COUNT
ØM  ØØØØ5  ØØØØ2  ØØØØ3   ØØØ1Ø   ØØ48184    ØØØØØ3
ØP  USERID   HOST   STATUS
ØQ  $Y$CON   NOD2   ACTIVE
ØA  $Y$CON   NOD4   ACTIVE
ØB DDPØ22 FTR Ø22 STATUS   COMMAND COMPLETED WO=ØØØØ27 12:25:55
```

8.5.1.1.3. JOB= parameter

This parameter displays:

- Command work-order-number
- Jobname
- Job state: whether active, backlogged, rolled-out, terminated normally or abnormally
- Seconds of CPU time used by the job
- Amount of main storage used by the job
- Number of pages, cards of output generated

Example: Response to DDP STATUS JOB:

```
JOB=________PRI=__STEP=___PROG=________SIZE=________

CPU TIME=__:__:__:__  PAGES=______CARDS=______

CURRENT JOB CONDITION=__________ __________ __________
```

where:

JOB=
: Is the name of the job you're asking the status of.

PRI=
: Is the current priority of the job.

STEP=
: Is the current job step number.

PROG=
: Is the program currently being executed.

SIZE=
: Is the memory being used by the job.

CPU TIME=
: Is the amount of CPU time used by the job.

PAGES=
: Is the number of pages of spooled output produced by the job.

CARDS=
: Is the number of punched cards produced by the job.

CURRENT JOB CONDITION=
: Is the state of the job at this moment.

Log printout sample:

```
DDP STATUS JOB=NOD20003
ØB DDPØØ2 CA1 ØØ2 STATUS   COMMAND ACCEPTED  WO=ØØØØ22 12:2Ø:32
ØC JOB=NOD20003  PRI=Ø5  STEP=ØØ1  PROG=SMPLMUØØ  SIZE=ØØ3584Ø
ØD CPU TIME= ØØ:ØØ:12.965   PAGES= ØØØØØØ   CARDS= ØØØØØØ
ØE CURRENT JOB CONDITION = WAITING FOR I/O
ØF DDPØ22 FTR Ø22 STATUS   COMMAND COMPLETED WO=ØØØØ22 12:2Ø:44
BR CN
```

8.5.1.1.4. FILE= parameter

This parameter displays:

- Filename
- File type
- Number of extents
- Number of cylinders
- Number of tracks
- Block size
- Creation date
- Expiration date

The file-id parameter and security read and write keys must be expressable as part of file-id for each system.

The volume name must also be expressable as part of the file-id for files registered in the volume table of contents (VTOC).

If the file-id contains any of the following characters embedded, then it must be expressed as a quoted string by enclosing it in apostrophes:

ampersand	exclamation mark
apostrophe	number (#)
comma	semicolon
equals (=)	space

Each single occurrence of an apostrophe within the file-id must be replaced by two apostrophes, for example:

JOE'S FILE would be expressed as:

```
'JOE''S FILE'
```

Example: Response to DDP STATUS FILE:

```
FILE=____________________________  EXT=___
VSN=______CYL=____TRK=__TYPE=_____BKSZ=_____
RCSZ=_____CRE-DATE=__/__/__EXP-DATE=__/__/__
```

where:

FILE=
: Is the name of the file you're requesting status of.

EXT=
: Is the number of extents occupied by the file.

VSN=
: Is the VSN from the file name string, if specified.

CYL=
: Is the number of cylinders occupied by this file.

TRK=
: Is the number of tracks (beyond whole cylinders), occupied by this file.

BKSZ=
: Is the block size of the file.

RCSZ=
: Is the record size of the file.

CRE-DATE=
: Is the date the file was created.

EXP-DATE=
: Is the expiration date of the file.

NOTE:

BKSZ, RCSZ, CRE-DATE, and EXP-DATE parameters are only available for files that have been opened at least once for processing. A DDP CREATE command does not open the file, and requesting the status of a newly created file will not return all of the fields described previously.

Log printout sample:

```
DDP STATUS FILE=',$Y$SCLOD,RES'
ØG DDPØØ2 CA1 ØØ2 STATUS   COMMAND ACCEPTED  WO=ØØØØ25 12:24:32
ØJ FILE= $Y$SCLOD                                    EXT= ØØ2
ØK VSN= RES     CYL= ØØ15  TRK= ØØ  TYPE= SAT    BKSZ= ØØ256
ØL RCSZ= ØØ256  CRE-DATE= 82/Ø5/14  EXP-DATE= 99/12/31
DM DDPØØ2 FTR Ø22 STATUS   COMMAND COMPLETED WO=ØØØØ25 12:24:42
```

8.5.1.1.5. USER= parameter

This parameter displays:

- The DDP functions performed for a particular user
- Uncompleted and completed commands
- Status of any user in your system (if issued from a system console)
- Your own status only (if issued from a workstation or terminal)

Example: Response to DDP STATUS USER

```
USERID   BUFFERS   BUF SIZE   W. O. COUNT
------   -------   --------   -----------
W.O. #   FUNCTION   PRIM   SECN   STARTED   COMPLETE   STATUS
------   --------   ----   ----   -------   --------   ------
```

where:

USERID
: Is the user-id of the user whose status you're requesting.

BUFFERS
: Is the number of buffers the user is using.

BUF SIZE
: Is the total buffer space the user is using.

W. O. COUNT
: Is the number of active work orders being processed for this user.

W. O. #
: Is the work order number of the command whose status you're requesting. For additional information about work order numbers, see 8.2.6.2.

FUNCTION
Is the name of the command whose status you're requesting, such as DDP, COPY, or PAUSE.

PRIM
Is the host-id of the primary host involved with a command. If the command involves an originating file, the primary host is the host that has that file. Otherwise, the primary host is the host originating this command.

SECN
Is the host-id of the secondary host involved with a command. If the command involves an originating file, the secondary host is the destination host of that file. Otherwise, the secondary host is the destination host for this command.

STARTED
Is the time a command was sent.

COMPLETE
Is the time a command was completed, if it was.

STATUS
Specifies status information.

Log printout sample:

```
DDP STATUS USER=OPERATOR
ØQ DDPØØ2 CA1 ØØ2 STATUS   COMMAND ACCEPTED  WO=ØØØØ26 12:25:1Ø
ØA  USERID   BUFFERS   BUF SIZ   W.O. COUNT
ØB  $Y$CON    ØØØØ4    ØØ18352    ØØØØØØ
ØC W.C.#  FUNCTION PRIM SECN STARTED  COMPLETE     STATUS
ØD ØØØØØ1 PURGE    NOD2 NOD2 12:Ø5:28 12:Ø5:47   TERM NORMALLY
ØE ØØØØØ2 PURGE    NOD2 NOD2 12:Ø5:28 12:Ø5:44   TERM NORMALLY
ØF ØØØØØ3 CREATE   NOD2 NOD2 12:Ø6:16 12:Ø6:18   TERM NORMALLY
ØG ØØØØØ4 COPY     NOD4 NOD2 12:Ø7:41 12:Ø7:47  TERM ABNORMALLY
ØJ ØØØØØ5 COPY     NOD4 NOD2 12:Ø9:14 12:Ø9:32  BEING PROCESSED
ØK ØØØØØ9 COPY     NOD4 NOD2 12:13:36 12:13:56  BEING PROCESSED
ØL ØØØØ13 SUBMIT   NOD2 NOD4 12:15:34 12:16:Ø5   TERM NORMALLY
ØM ØØØØ15 TALKX    NOD4 NOD2 12:16:28 12:16:47   TERM NORMALLY
ØP ØØØØ17 TALKX    NOD4 NOD2 12:17:Ø4 12:17:Ø7   TERM NORMALLY
ØQ ØØØØ18 SPOOL    NOD2 NOD4 12:17:Ø5   :  :      IN PROGRESS
ØA ØØØØ21 SUBMIT   NOD2 NOD2 12:19:15 12:19:37   TERM NORMALLY
ØB ØØØØ22 STATUS   NOD2 NOD2 12:2Ø:34 12:2Ø:44   TERM NORMALLY
ØC ØØØØ23 STATUS   NOD2 NOD2 12:21:38 12:21:46   TERM NORMALLY
ØD ØØØØ24 STATUS   NOD2 NOD2 12:23:37 12:23:42   TERM NORMALLY
ØE ØØØØ25 STATUS   NOD2 NOD2 12:24:33 12:24:42   TERM NORMALLY
ØF ØØØØ26 PURGE    NOD2 NOD2 12:25:11   :  :      IN PROGRESS
ØG DDPØ22 FTR Ø22 STATUS   COMMAND COMPLETED WO=ØØØØ26 12:25:22
BR CN
```

8.5.2. Communicating with a Remote Operator or User (DDP TALK)

Function and Requirements:

The DDP TALK command lets you send a message to a remote operator or user. You might want to have the remote operator mount your disk pack, for instance. Or, you might need to ask a remote user if he has recently updated a file you need.

If the recipient of the DDP TALK message isn't logged on, DDP displays the message at the system console.

Format:

```
DDP△TALK△MESSAGE='string'△USER=[{host-id      }::]{OPERATOR}△[wait]
                                [{local-host-id}]  {user-id }
```

Parameters:

`string`
Is the character string message you want the operator or user to get.

`host-id`
Is one to four alphanumeric characters naming the host where the operator or user is. If you omit the host-id, DDP assumes the user is on your local host. For more information, see 7.1.2.

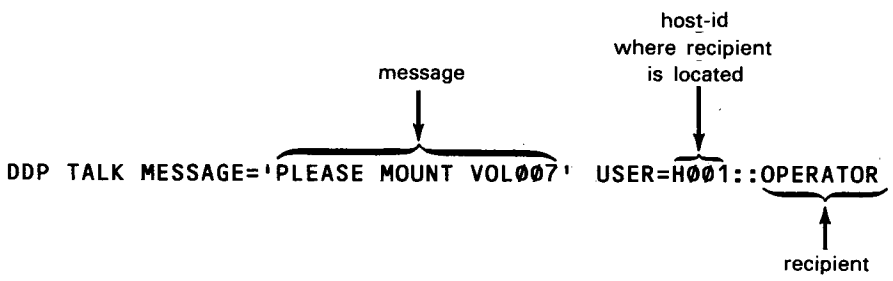
`OPERATOR`
Is the operator of the remote host.

`user-id`
Is one to six alphanumeric characters identifying the user to the system. This is the same id as used in the LOGON command.

`WAIT`
Informs the operator or user that you want a reply. You do not have to wait for this reply to go on with other commands. Your keyboard isn't locked during this period.

Example:

You want the operator on host H001 to mount your volume (VOL007). The command is:



8.6. HELP SCREENS FOR DDP COMMANDS

Help screens are available for all of the DDP commands. To obtain information about available DDP commands and further help:

1. Type: HELP DDP

   the following information will be displayed:

DDP Commands	Help Screen Requests
DDP CANCEL	HELP DDPCAN
DDP COPY	HELP DDPCOP
DDP CREATE	HELP DDPCRE
DDP PURGE	HELP DDPPUR
DDP STATUS	HELP DDPSTA
DDP SUBMIT FILE	HELP DDPFSB
DDP SUBMIT REQUEST	HELP DDPRSB
DDP TALK	HELP DDPTAL
Parameters for CREATE	HELP DDPPAR

2. Enter the appropriate help screen request format shown in the aforementioned step 1.

3. Enter HELP DDPPAR if you need help with the parameters for the DDP CREATE COMMAND.

NOTE:

If you attempt to enter DDP through the System Menu, you will be told to enter HELP DDP for information about DDP.

9. The DDP File Access Facility

9.1. INTRODUCTION

This section describes how to:

- access and process files residing on remote OS/3 systems; and
- write application programs that can initiate and communicate with one another to exchange data and control information.

Both of these capabilities are provided as part of the DDP file access facility.

9.2. REMOTE FILE PROCESSING

Your programs can now access and process remote disk data (not library) files by simply adding a HOST=host-id parameter on the // DVC job control statement associated with a remotely located disk file. For example, if the disk file declared in the following control stream:

```
// JOB MYJOB
     .
     .
     .
// DVC 50
// VOL D00028
// LBL INVFILE
// LFD INPUTA
// EXEC PROGA
```

was located at a remote site, all that would be required to indicate that the file is at a remote site is the addition of the host-id parameter; as shown in the following:

```
// JOB
    .
    .
    .
// DVC 50,HOST=REM2
// VOL D00028
// LBL INVFILE
// LFD INPUTA
// EXEC PROGA
/&
```

No program changes or additions are needed to process a remote file.

9.2.1. Cataloging Remote Files

Remotely accessed files can be cataloged as described in the file cataloging concepts and facilities manual, UP-8860 (current version). Further, they can be cataloged at multiple hosts. The host where the file resides would catalog it like any other file (excluding the HOST= parameter) while other hosts would include the HOST= parameter in its file description.

9.2.2. Sharing Remote Files

Remotely accessed files can be declared as sharable read files but cannot be shared for updating purposes (read/write files).

9.2.3. Tradeoffs Involved when Processing Remote Files

Figure 9–1 shows an application where the ability to access and process data at remote hosts has distinct advantages. As shown, the inventory display program is connected with three inventory files: one local and two remote. When information is needed about a particular product, the workstation user need only enter a display request.

The inventory display program running at site A opens the applicable files and displays the requested information. The workstation user is unconcerned about where the requested information is stored.

Such an application assumes that the inventory files maintained at the remote hosts are dynamic files that are constantly being updated, processed, and maintained by the remote hosts. Consequently, it would be inefficient to make copies of these files on host A each time host A wanted to access these files.

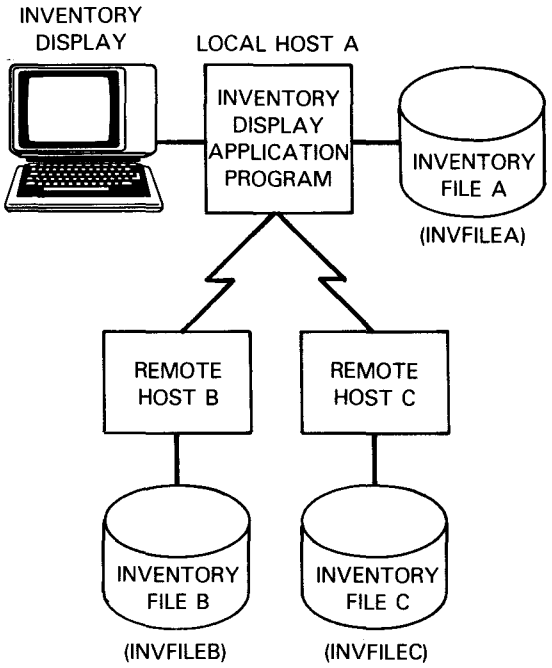


Figure 9–1. Remote File Processing Application

9.3. PROGRAM-TO-PROGRAM COMMUNICATIONS

You can now write assembly language programs that can initiate and carry on a *conversation* with other assembly language programs at remote hosts. The communicating programs must reside on different hosts in a DDP network.

9.3.1. Two Types of Programs: Primary and Surrogate

There are two basic communication programs: primary and surrogate. The program that initiates a conversation is called the *primary* program and the initiated program is called the *surrogate* program. However, primary and surrogate programs can be designed to reverse roles if the application they are being used in so dictates. A primary program can call up to 255 surrogate programs.

9.3.2. Two Types of Conversations: Simple and Complex

There are two types of conversations that may be conducted between communicating programs: simple and complex. By definition, a simple conversation is one that takes place between only two programs and the primary/surrogate relationship between these two programs never changes. The primary program remains the primary program throughout the life of the conversation. There is a simple exchange of information. A primary program activates a surrogate program and provides it with data to be processed. The surrogate program performs the required processing.

A complex conversation is one that allows for the communicating programs to reverse primary/surrogate status, or that allows for more than two programs to be involved in the conversation. The procedures and guidelines for preparing communicating programs follow.

9.4. PROGRAMMING CONSIDERATIONS FOR SIMPLE CONVERSATIONS

9.4.1. Job Control Requirements

Programs designed for use in a simple conversation application rely on job control to identify the location and name of the surrogate program. Simple conversation applications assume that the primary program will be initiated through the job control run process by a human operator and that the surrogate program will be initiated through the job control run process by the DDP facility. Consequently, both programs require an associated job control stream and both programs will occupy job slots in the system on which they are executing.

The job control streams associated with communicating programs must contain a device assignment set for the program being addressed. For the primary program, this device assignment set must consist of a // DVC statement and a // LFD statement. The // DVC statement must be in the form:

```
// DVC PROG,jobname,HOST=host-id
```

where:

`PROG`
: Associates the device assignment with the program-to-program component, rather than a conventional I/O device.

`jobname`
: Identifies the name of the job you wish to communicate with.

`HOST=host-id`
: Identifies the network name of the system on which the corresponding program is located. (This name is established when the communications network for your system is defined during system installation.)

The // LFD statement must be in the format:

```
// LFD filename
```

where:

`filename`
: Is the name specified for the FILENAME= parameter on the CDIB macroinstruction issued by the primary program.

The surrogate device assignment set must also issue a // DVC statement and // LFD statement. However, the // DVC statement must have the following format:

```
// DVC PROG
```

where:

PROG
: Associates the device assignment with the program-to-program software, rather than a conventional I/O device.

The // LFD statement must be in the format:

```
// LFD filename
```

where:

filename
: Is the name specified for the FILENAME= parameter on the CDIB macroinstruction issued by the surrogate program.

Example:

Control Stream To Run Primary Program at HOST AAAA

```
// JOB PRIMJOB
    .
    .
    .
// DVC PROG,SURJOB,HOST=BBBB
// LFD PRIMARY
// EXEC PROGA
/&
// FIN
```

Control Stream To Run Surrogate Program at HOST BBBB

```
// JOB SURJOB
    .
    .
    .
// DVC PROB
// LFD SURRGATE
// EXEC PROGB
/&
// FIN
```

9.4.2. Coding Requirements

To prepare BAL programs for use in simple conversation applications, you use a special set of consolidated data management macroinstructions to define, open and close, and transfer information between the communicating programs. These macroinstructions function the same way other data management macroinstructions do and, consequently, are designed to resemble them wherever possible. Two declarative and four imperative macroinstructions are used in this environment.

9.4.3. Declarative Macroinstructions

Declarative macroinstructions are used to supply information concerning the files required by the issuing program. These macroinstructions generate nonexecutable code such as constants and storage areas for variables. The following two macroinstructions are used: CDIB and RIB.

9.4.3.1. Defining a Common Data Interface Block (CDIB)

The CDIB macroinstruction identifies the logical file required by the issuing program and establishes a common data interface block for the file. The CDIB is the function passing mechanism for the file and it is referenced each time you issue an imperative macroinstruction.

Format:

LABEL	△OPERATION△	OPERAND
name	CDIB	FILENAME=filename

Parameters:

`name`
Specifies the name of the CDIB. This name cannot exceed seven characters.

`filename`
Specifies the filename that was assigned in the // LFD statement in the program execution job control stream for the issuing program.

9.4.3.2. Defining a Resource Information Block (RIB)

The RIB macroinstruction is used to describe the file characteristics required by the issuing program. The RIB is used in combination with the CDIB macroinstruction when you issue the DOPEN imperative macroinstruction to open the file.

Format:

LABEL	△OPERATION△	OPERAND
name	RIB	[HOSTID=host-id] [,PROGFD=symbol] [,RCSZ=n] [,WKFM={[IO] VAR VARI}]

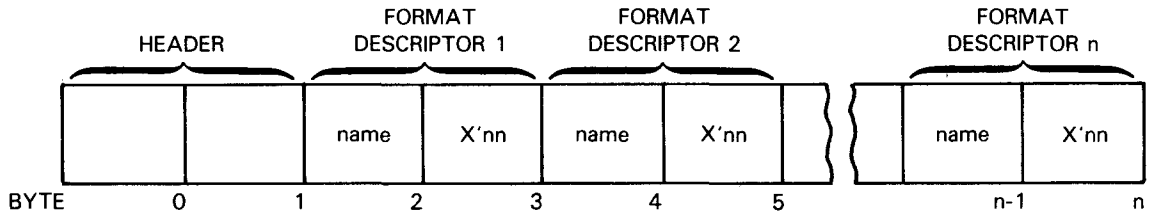
Parameters:

`HOSTID=host-id`

Specifies the 1- to 4-character alphanumeric name (first character must be alphabetic) of the host on which the destination program resides. This parameter is overridden by the HOST=host-id parameter of the // DVC PROG statement in a simple communications application environment.

`PROGFD=symbol`

Specifies the symbolic address of a data format descriptor list that you provide to indicate the type of data that is being transferred. This list has the following format:



Notice that the data format descriptor list consists of a 2-byte header followed by one or more contiguous 2-byte data format descriptor entries. The 2-byte header contains the number of list entries in binary. The first byte of the format descriptor entry contains a name that you define for a particular type of data. The second byte of the entry contains a hexadecimal value that specifies the particular type of data. The hexadecimal values and the type of data they specify are as follows:

Hexadecimal Value	Type of Data
X'80'	ASCII
X'81'	Compressed ASCII
X'82'	EBCDIC
X'83'	Compressed EBCDIC
X'84'	Katakana
X'85'	Compressed Katakana
X'86'	Transparent Octet
X'88'	Kanji
X'89'	Compressed Kanji

This list is used when you transfer data. It is used with the DMINP and DMOUT operative macroinstructions (9.4.4.5, 9.4.4.4). If the PROGFD parameter is omitted, the data type is assumed to be transparent octet.

RCSZ=
: Specifies the length, in bytes, of each record to be transferred. If RCSZ is omitted and WKFM=NO or WKFM=VAR is specified, the record length is assumed to be 256 bytes.

WKFM=
: Specifies the format of the work area in which input and output records are placed.

WKFM=NO
: Specifies that the work area format is the same as the record format. NO is the default.

WKFM=VAR
: Specifies that the work area format is variable. For output operations, VAR must specify the amount of data in the first two bytes of the work area. For input operations, the maximum amount of data (determined by the RCSZ keyword parameter) is placed in the first two bytes of the work area.

WKFM=VARI
: Specifies that the work area format is variable. For output operations, you must specify the amount of data in the first two bytes of the work area. For input operations, the first two bytes of the work area are automatically updated to reflect the amount of data.

9.4.4. Imperative Macroinstructions

The imperative macroinstructions are used to establish and close a communication path between user application programs, begin and end conversation between programs, transfer and receive data, and control the status of the programs. The instructions you use to cause these actions are: DOPEN, DCLOSE, DMOUT, and DMINP.

9.4.4.1. Register Conventions

When you use the imperative macroinstructions, the following registers must be used as indicated:

Register Ø
: The RIB address must be loaded in this register for the DOPEN macroinstruction. The workarea address must be loaded in this register for those macroinstructions that use workarea processing.

`Register 1`

The CDIB must always be loaded in this register.

If symbolic notation is used, the expansion of the imperative macroinstruction will load register 0 and/or register 1 with the appropriate address. All registers are returned unchanged when control is received back from an imperative macroinstruction.

9.4.4.2. Status Checking

Status checking is required after each imperative macroinstruction. This is necessary because control is always returned inline. A standard way of checking macroinstructions is described in the consolidated data management macroinstruction user guide, UP-8826 (current version).

9.4.4.3. Establishing a Communication Path (DOPEN)

You use this macroinstruction to establish a communications path between user application programs. This instruction must be issued first by both the primary and surrogate application programs before any transfer of information is attempted.

Format:

LABEL	△OPERATION△	OPERAND
name	DOPEN	{cdibname / (1) / 1}, {ribname / (Ø) / Ø}

Parameters:

`cdibname`
Is the symbolic name of the CDIB required by the program issuing the DOPEN.

`(1) or 1`
Indicates that you have preloaded register 1 with the address of the CDIB.

`ribname`
Is the symbolic name of the RIB required by the program issuing the DOPEN.

`(Ø) or Ø`
Indicates that you have preloaded register 0 with the address of the RIB.

9.4.4.4. Outputting Data (DMOUT)

The DMOUT macroinstruction is used to transfer (send) data from one application program to another.

Format:

LABEL	△OPERATION△	OPERAND
name	DMOUT	{cdibname / (1) / 1}, {workarea / (Ø) / Ø}

Parameters:

cdibname
Is the name of the CDIB required by the program issuing the DMOUT.

(1) or 1
Indicates that you have preloaded register 1 with the address of the CDIB.

workarea
Is the symbolic name of the work area that contains the record to be sent.

(Ø) or Ø
Indicates that you have preloaded register 0 with the address of the work area.

9.4.4.5. Receiving Data (DMINP)

The receiving macroinstruction has the following format:

Format:

LABEL	△OPERATION△	OPERAND
name	DMINP	{cdibname / (1) / 1}, {workarea / (Ø) / Ø}

Parameters:

cdibname
Is the name of the CDIB required by the program issuing the DMINP.

(1) or 1
Indicates that you have preloaded register 1 with the address of the CDIB.

workarea
Is the symbolic name of the work area that contains the record to be sent.

(Ø) or Ø
Indicates that you have preloaded register 0 with the address of the work area.

If the PROGFD is specified in the RIB, you select the data format from the data format descriptor list by moving the 1-byte name you defined for the particular format into the CI$FD field of the CDIB before you issue the DMOUT macroinstruction. If you do not move a name into the CDIB, the first data format in the list will be used. On a DMINP, the data name that comes across in the input buffer will be moved into the CI$FD field.

9.4.4.6. Closing a Communications Path (DCLOSE)

You use this macroinstruction to close a communications path between user application programs.

Format:

LABEL	△OPERATION△	OPERAND
name	DCLOSE	{cdibname / (1) / 1}

Parameters:

cdibname
Is the symbolic name of the CDIB declared in the program issuing the DCLOSE.

(1) or 1
Indicates that you have preloaded register 1 with the address of the CDIB.

9.4.5. Timing Considerations

We recommend that the job streams for program-to-program communications contain no other job steps than those required for the execution of these programs. The reason for this recommendation is that only 10 minutes is allotted for starting a job; otherwise, a time-out will result.

9.4.6. Designing a Simple Conversation Program

Simple conversation must contain the following three phases, which are also shown in the flow diagrams in Figure 9–2:

- Initialization Phase

  Wherein the primary program initializes the conversation, via a DOPEN, with one and only one surrogate program, which also must issue a DOPEN.

- Data Transfer Phase

  Wherein the primary program only transfers data to the surrogate program, after issuing DMOUTs and the surrogate receives data after issuing DMINPs.

- Termination Phase

  Wherein the primary program and surrogate program must each issue a DCLOSE to terminate.

The CDIBs and RIBs required by the DOPEN, DMOUT, DMINP, and DCLOSE macroinstructions are those of the program issuing the macroinstructions. Examples of simple conversation programs are given in 9.4.7 and a corresponding flow chart is shown in Figure 9–2.

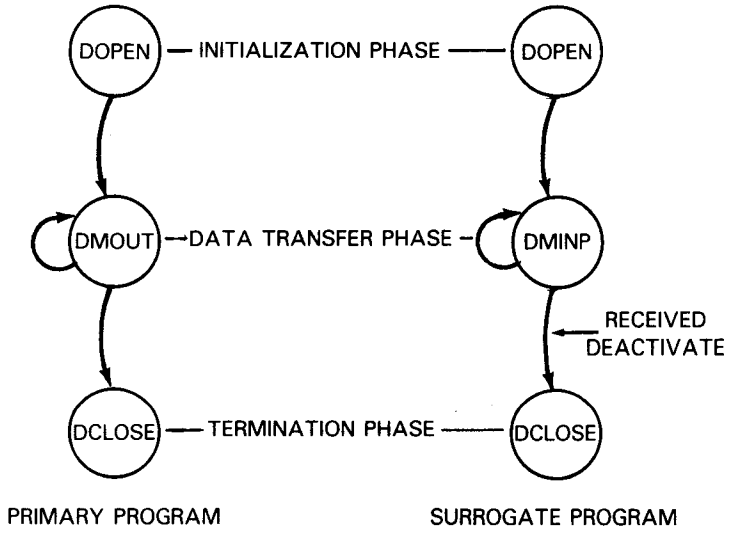


Figure 9–2. Simple Conversation Program Flow Diagrams

9.4.7. Examples of Simple Conversation Communications Programs

The following are examples of simple conversation programs.

9.4.7.1. Example 1: Simple Conversation Primary Program and Surrogate Program Pair

Figure 9–3 and Figure 9–4 show the job control and coding, respectively for a very simple, complete primary program and surrogate program communicating pair.

The following is the required job control:

Primary Program	Surrogate Program
Job Control Stream	
// JOB PTP$PRIM	// JOB PT$SURR
.	
.	
.	
// DVC PROG,PTP$SURR,HOST=BBBB	// DVC PROG
.	.
.	.
.	.
// LFD filename field in the primary CDIB	// LFD filename field in the surrogate CDIB
.	.
.	.
.	.
// EXEC load module name	// EXEC load module name
.	.
.	.
.	.
/&	/&
// FIN	// FIN

Figure 9–3. Job Control for Example 1

The following is the coding for example 1:

Primary Program			Surrogate Program		
Line					
1	PRIMARY START	Ø	SURROGATE START		Ø
	.			.	
	.			.	
	.			.	
2	LA	R1,PRIMCDIB		LA	R1,SURRCDIB
	.			.	
	.			.	
	.			.	
3	LA	RØ,PRIMRIB		LA	RØ,SRGATRIB
4	DOPEN	(1),(Ø)		DOPEN	(1),(Ø)
	.			.	
	.			.	
	.			.	
5	LA	RØ,DATAMSG		LA	RØ,IOAREA
6	DMOUT	(1),(Ø)		DMINP	(1),(Ø)
	.			.	
	.			.	
	.			.	
7	DCLOSE	(1)		DCLOSE	(1)
	.			.	
	.			.	
	.			.	
8	EOJ			EOJ	
	PRIMCDIB DC	XL48'ØØ'	SURRCDIB	DC	XL48'ØØ'
	PRIMRIB RIB	RCSZ=8Ø	SURGATRIB	RIB	RCSZ=8Ø
	DATAMSG DC	CL8Ø'ØØ'	IOAREA	DC	XL8Ø'ØØ'

Figure 9–4. Simple Conversation Primary Program and Surrogate Program Pair for Example 1 (Part 1 of 2)

Line	Coding Description
1	Each program starts its own program.
2 and 3	The primary and surrogate programs each load their respective CDIBs and RIBs into registers R1 and R0, respectively.
4	Both the primary and surrogate programs open their own CDIBs and RIBs.
5	The primary program loads a data message into R0, and the surrogate program loads its I/O area into its R0.
6	The primary program issues a DMOUT to send the message and the surrogate program issues a DMINP to receive the message.
7	Both programs close their respective communication paths to each other.
8	Each program ends its own program.

Figure 9–4. Simple Conversation Primary Program and Surrogate Program Pair for Example 1 (Part 2 of 2)

9.4.7.2. Example 2: Simple Conversation Primary Program Repeatedly Transferring Data to a Surrogate Program

Figure 9–5 shows a primary program transferring 50 messages to a surrogate program (shown in Figure 9–6) during simple conversation.

Job Control Stream Required

```
// JOB PRIMJOB
    .
    .
    .
// DVC PROG,SURJOB,HOST=BBBB
// LFD PRIMARY
    .
    .
    .
// EXEC PROGA
/&
// FIN
```

Figure 9–5. Job Control and Coding for Primary Program for Example 2 (Part 1 of 2)

BAL Program for the Simple Conversation Primary Program PROGA

```
Line                                    Description
1       PROGA START  Ø
2            BALR    R15,Ø
3            USING   *,R15
4*                                      LOADING THE CDIB
5            LA      R1,PRIMARY             Load the address of the following CDIB into R1:
             .                                  PRIMARY CDIB FILENAME=PRIMARY
6*           .                          OPEN A CONNECTION TO THE SURROGATE HOST BBBB
7            LA      RØ,PRIMRIB             Load the address of the following RIB into R0:
                                                PRIMRIB RIB
8            DOPEN   (1),(Ø)                Open a session path to the surrogate program
9            .                              Check for a successful open
1Ø*          .                          EXECUTE DATA TRANSFER LOOP
11           .                                  Set up counter for Loop
12           LA      RØ,DATAAREA            Loop { R0= address of data area
13           DMOUT   (1),(Ø)                     Loop is executed, sending data
14           .                                   Check for a successful send
15           .                                   Loop through 40 times
16*          .                          CLOSE THE CONNECTION TO THE SURROGATE HOST
17           DCLOSE (1)                     Close the session path
18           .                              Check for a sucessful close
             .
             .
             EOJ
```

Figure 9–5. Job Control and Coding for Primary Program for Example 2 (Part 2 of 2)

9.4.7.3. Example 3: Simple Conversation Surrogate Program Repeatedly Receiving Data from a Primary Program

Figure 9–6 shows a surrogate program receiving 50 messages from the primary program (shown in Figure 9–5) during simple conversation.

Job Control Stream Required

```
// JOB SURJOB
     .
     .
     .
// DVC PROG
     .
     .
     .
// LFD SURRGATE
     .
     .
     .
// EXEC PROGB
     .
     .
     .
/&
// FIN
```

Figure 9–6. Job Control and Coding for Surrogate Program for Example 3 (Part 1 of 2)

BAL Program for the Simple Conversation Surrogate Program PROGB

```
Line                                    Description
1       PROGB START  Ø
2            BALR    R15,Ø
3            USING   *,R15
4*                                      LOADING THE CDIB
5            LA      R1,SURRGATE            Load the address of the following CDIB into R1:
             .                                   SURRGATE CDIB FILENAME=SURRGATE
             .
6*           .                          ATTACH TO THE PRIMARY PROGRAM
7            LA      RØ,SURRIB              Load the address of the following RIB into R0:
                                                SURRIB RIB
8            DOPEN   (1),(Ø)                Attach to the primary session
9            .                              Check for a successful attachment
             .
1Ø*          .                          PREPARE TO RECEIVE DATA FROM THE PRIMARY PROGRAM
11           LA      RØ,INPAREA         Loop { Load the address of the input area into R0
12           DMINP   (1),(Ø)                   Get data from primary program
13           .                                 Receive data transfer
14           .                                 Loop to get more input
15*          .                          CHECK WHICH OF THE EXCEPTION FLAGS IS SET IN THE CDIB
16*                                     TERMINATE THE ATTACHMENT TO THE PRIMARY PROGRAM
17           DCLOSE  (1)                    Detach from the primary program
18           .                              Check for a successful detach
             .
             .
             EOJ
```

Figure 9–6. Job Control and Coding for Surrogate Program for Example 3 (Part 2 of 2)

9.5. PROGRAMMING CONSIDERATIONS FOR COMPLEX CONVERSATIONS

A complex conversation enables both primary and surrogate programs to send and receive data and exchange status roles via a more controlled discipline. Additionally, a primary program can simultaneously communicate with several surrogate programs.

9.5.1. The Three Phases of Complex Conversation

A complex conversation also contains three phases:

- Initialization Phase

  Wherein the primary program initializes a conversation with one or more surrogate programs and receives a reply from the surrogate program.

- Data/control Transfer Phase

  Wherein the primary and surrogate programs exchange data and/or control.

- Termination Phase

  Wherein the primary program indicates to the surrogate that a conversation is to be terminated. The surrogate program responds to the primary and ends the conversation.

9.5.2. Operational Characteristics of Complex Conversation

The following operational characteristics apply:

1. A program written as a primary program cannot be started as a surrogate program. However, once the primary program has started, it can pass to a surrogate transfer phase.

2. A program written as a surrogate program cannot be started as a primary program.

3. A job started (and running) by a primary program, whether it is running as a primary or surrogate program, cannot be attached to by another job. That is, there is no way to start or connect to a job that is already running.

9.5.3. Macroinstructions Used in Complex Conversation

The same declarative and imperative macroinstructions used in simple conversation are required in complex conversation, except that the keyword USE=PROG is also required in the primary program and surrogate program RIB for complex conversation, as follows:

Format:

LABEL	△OPERATION△	OPERAND
ribname	RIB	USE=PROG
		other keyword parameters

The same imperative macroinstructions used in simple conversation are also used in complex conversation, along with the following two imperative macroinstructions: DMSEL and DMCTL.

9.5.3.1. Selecting the Surrogate (DMSEL)

The select imperative macroinstruction, DMSEL, is used to begin and end conversations between paired application programs. Only the primary program can issue a DMSEL, and, since the surrogate program must reply with a DMOUT, the primary must next issue a DMINP. The DMSEL format is as follows:

Format:

```
DMSEL {(1)      },PROG,{(Ø)             },{ACT  }[,DATA]
      {cdibname}       {surrogate-prog}  {DEACT}
```

Parameters:

`cdibname`
Is the name of the CDIB specified in the primary program. If (1) is specified, it is assumed that the address of the CDIB has been preloaded into register 1.

`PROG`
Identifies DMSEL as belonging to the program-to-progam facility.

`surrogate-prog`
Specifies the name of the surrogate with which a conversation is to be activated or deactivated. If (0) is specified, it is assumed that the address of the name of the surrogate program has been preloaded into register 0.

`ACT`
Indicates a conversation with the surrogate program named is to be activated (opened).

`DEACT`
Indicates a conversation with the surrogate program named is to be deactivated (closed).

`DATA`
Indicates that the next macroinstruction issued will contain data to be passed with the DMSEL.

9.5.3.2. Special Control (DMCTL)

The special control imperative macroinstruction can only be used:

- by the primary program to pass control to the surrogate program via the BEQueath parameter; or
- by the surrogate program to reply to a DMSEL with DEACT, issued by the primary program.

The format for DMCTL is as follows:

```
DMCTL{(1)      },PROG,{BEQ}
     {cdibname}       {END}
```

Parameters:

cdibname
: Is the name of the CDIB specified in the primary or surrogate program, depending on whether the BEQ or END parameters, respectively, are specified as described below.

PROG
: Identifies DMCTL as belonging to the program-to-program component.

BEQ
: Issued only by the primary program, indicating it is passing control to the surrogate program. Upon reception of the BEQueath, the surrogate program becomes the primary program and the primary program becomes the surrogate program. If a reply was required from the surrogate, it is sent before the BEQueath takes effect.

END
: Issued only by the surrogate program, and only as a reply to a DMSEL with DEACT, issued by the primary program, to indicate the conversation is to be terminated.

9.5.3.3. Closing a Conversation Path (DCLOSE)

The DCLOSE imperative macroinstruction is issued to close a communications path between user application programs, as for simple conversations (9.4.4.6), with the following exception. If the primary program has not issued a DMSEL with DEACT, or the surrogate program received the same when the DCLOSE was issued or received, the conversation will be aborted.

9.5.4. Capabilities of Complex Conversation

Compared to simple conversation, there are many more ways you can design primary and surrogate programs using the aforementioned imperative macroinstructions.

Various procedures have been designed using these macroinstructions that you can use to perform the three basic communications phases:

- conversation initiation phase;
- data and control transfer phase; and
- conversation termination phase.

Additionally, within these basic procedures, various smaller steps can be taken, or you can link the procedures to pass from primary data transfer state to surrogate transfer state and, then, go back to the primary transfer state.

Figure 9–13 is a master flowchart showing details of the procedures in all possible combinations that you might employ in writing a program set. Horizontally, the left section of the diagram shows the possible flow of a program starting as a primary program. The right section of the diagram shows a similar flow for a program starting as a surrogate program. Vertically, the diagram shows the program flow, for both programs, starting with the conversation initialization phase at the top, then the data and control phase and, finally, the conversation termination phase at the bottom of the diagram.

The required macroinstructions and parameters are shown in the large circles of the procedure links and the smaller circles define the state your program arrives at. For example, all small circles with a 1 within signify the program has arrived at the primary data/control transfer phase. The small circles containing 2 signify the program has arrived at the surrogate data/control transfer phase.

When a program starting as primary reaches the ② state, it continues from the surrogate state ② . Similarly, when a program starting as a surrogate reaches the ① state, it continues from the primary diagram state ① . See steps 6A to 7A, 6C to 7C, and 5D to 6D for examples 4, 5, and 6, respectively, and refer to Figure 9–13.

The small circles enclosing a 3 or 4 indicate the program can enter the primary conversation termination phase or surrogate conversation termination phase, respectively.

The notes signifying the conditions required before being able to continue in the flow, such as "received data", "pass data", "received BEQueath" and so forth, indicate the protocol conditions required to issue the macroinstructions or conditions that follow the use of the macroinstruction. For further information, refer to the previously described macroinstruction specifications.

9.5.5. Job Control Requirements

The job control for complex conversation is very simple, as shown for the following pair of communicating programs.

For the Primary Program	For the Surrogate Program
`// JOB jobname`	`// JOB jobname`
.	.
.	.
.	.
`// EXEC local module name`	`// EXEC local module name`
.	.
.	.
.	.
`/&`	`/&`
`// FIN`	`// FIN`

where:

`jobname`
For the primary program, jobname is the job name specified for the primary program.

For the surrogate program, jobname is the job name specified for the surrogate program.

`load module name`
For the primary program, load module name is the name of the load module specified for the primary program.

For the surrogate program, load module name is the name of the load module specified for the surrogate program.

9.5.6. Examples of Complex Conversation Programs

The following are examples of complex conversation programs. All of the programs listed have been referenced on the complex conversation procedures flow diagrams chart (Figure 9–13), according to the coding line number listed at the left side of each program that follows. Thus, you can relate the examples to the flow diagrams and learn how and why the examples were coded as they are and what other procedures could be used, if your program required it.

9.5.6.1. Example 4: Complex Conversation Primary Program and Surrogate Program Pair With BEQueath

The primary program (on the left side of Figure 9–8) issues a DOPEN, selects the surrogate program via DMSEL ACTivate to exchange status via DMCTL BEQueath. After receiving acceptance from the surrogate program, the program becomes the new surrogate program and receives data from the new primary program.

The surrogate program (on the right side of Figure 9–8) issues a DOPEN, replies to a message from the primary program, accepts the BEQueath, and, as the new primary program, sends data to the new surrogate program. The program then issues a DMSEL DEACTivate and, after receiving a reply, closes the conversation.

The required job control is shown in Figure 9–7.

Primary Program	Surrogate Program
Job Control Streams	
// JOB PTP$PRIM	// JOB PTP$SRGT
.	.
.	.
.	.
// EXEC load module name	// EXEC load module name
.	.
.	.
.	.
/&	/&
// FIN	// FIN

Figure 9–7. Job Control for Example 4

The following is the coding for example 4:

Primary Program			Surrogate Program		
	PRIMARY START	Ø	SURROGATE	START	Ø
	.			.	
	.			.	
	.			.	
	LA	R1,PRIMCDIB		LA	R1,SURRCDIB
	.			.	
	.			.	
	.			.	
	LA	RØ,PRIMRIB		LA	RØ,SRGATRIB
Line			Line		
1A	DOPEN	(1),(Ø)	1B	DOPEN	(1),(Ø)
	.			.	
	.			.	
	.			.	
	LA	RØ,=CL8'PTP$SRGT'			
2A	DMSEL	(1),PROG,(Ø),ACT,DATA			
	LA	RØ,ACTMSG		LA	RØ,IOAREA
3A	DMOUT	(1),(Ø),UNLOCK	2B	DMINP	(1),(Ø)
	.			.	
	.			.	
	.			.	
	LA	RØ,IOAREA		LA	RØ,RPLYMSG
4A	DMINP	(1),(Ø)	3B	DMOUT	(1),(Ø)
	.			.	
	.			.	
	.			.	
5A	DMCTL	(1),PROG,BEQ			
	LA	RØ,BEQMSG		LA	RØ,IOAREA
6A	DMOUT	(1),(Ø)	4B	DMINP	(1),(Ø)
	.			.	
	.			.	
	.			.	
	LA	RØ,IOAREA		LA	RØ,=CL8'PTP$PRIM'
			5B	DMSEL	(1),PROG,(Ø),DEACT
7A	DMINP	(1),(Ø)		LA	RØ,DEACTMSG

Figure 9–8. Complex Conversation Primary Program and Surrogate Program Pair for Example 4 (Part 1 of 2)

Primary Program

```
8A          DMCTL   (1),PROG,END
            .
            .
            .
            LA      RØ,RPLMSG
9A          DMOUT   (1),(Ø)
            .
            .
            .
1ØA         DCLOSE  (1)
            .
            .
            .
            EOJ
  PRIMCDIB  DC      XL48'ØØ'
  PRIMRIB   RIB     USE=PROG,
                    HOSTID=BBBB
                    RCSZ=8Ø
  ACTMSG    DC      CL8Ø'ACTIVATE'
  BEQMSG    DC      CL8Ø'BEQUEATH'
  RPLYMSG   DC      CL8Ø'REPLY'
  IOAREA    DC      XL8Ø'ØØ'
```

Surrogate Program

```
6B          DMOUT   (1),(Ø),UNLOCK
            .
            .
            .
            LA      RØ,IOAREA
7B          DMINP   (1),(Ø)
            .
            .
            .
8B          DCLOSE  (1)
            .
            .
            .
            EOJ
SURRCDIB    DC      XL48'ØØ'
SRGATRIB    RIB     USE=PROG,
                    RCSZ=8Ø
RPLYMSG     DC      CL8Ø'REPLY'
DEACTMST    DC      CL8Ø'DEACTIVATE'
IOAREA      DC      XL8Ø'ØØ'
```

Figure 9–8. Complex Conversation Primary Program and Surrogate Program Pair for Example 4 (Part 2 of 2)

9.5.6.2. Example 5: Complex Conversation Primary Program Issuing a BEQueath and Repeatedly Receiving Data Transfers

Figure 9–9 shows the primary program issuing a DOPEN and DMSEL to select the surrogate program (shown in Figure 9–10) and passes control to it via a BEQueath. As the new surrogate program, it repeatedly receives data transfers from the new primary program, replying whenever necessary.

Job Control Stream Required

```
// JOB PRIMJOB
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

BAL Program for the Complex Conversation Primary Program PROGC

```
Line                                          Description
     PROGC    START  Ø
              BALR   R15,Ø
              USING  *,R15
                                              LOADING THE CDIB
                                                  Load address of the following CDIB into R1:
                                                      PRIMCDIB   CDIB   FILENAME=PRIMARY
                                              OPEN CONNECTION TO SURROGATE
              LA     RO,PRIMRIB                   Load address of the following RIB into RO:
              .                                       PRIMRIB    RIB    USE=PROG,HOSTID=ISLC
              .                                                         WKFM=VARI
              .

  1C          DOPEN  (1),(Ø)                      Open session path
              .                                   Check for successful open
              .                               SELECT THE SURROGATE PROGRAM
              .
              LA     RO,=CL6'SURJOB'              Load surrogate program name into RO
  2C          DMSEL  (1),PROG,Ø,ACT,DATA          Select surrogate program
              .                                   Check for successful open
              .
              .
              LA     RO,ACTMSG                    Load address of ACTivate DATA
```

Figure 9–9. Job Control and Coding for Example 5 (Part 1 of 3)

BAL Program for the Complex Conversation Primary Program PROGC

```
Line                                          Description
 3C      DMOUT  (1),(Ø),UNLOCK          Issue ACTIVATE
         .                              Check for successful ACTivate
         .                              ISSUE INPUT REQUEST TO GET REPLY FROM SURROGATE
         .
         LA     RO,RPLYAREA             Load address of reply are into RO
         .
         .
         .
 4C      DMINP  (1),(Ø)                 Wait for surrogate reply
         .                              Check whether reply was received
         .                              ISSUE BEQUEATH TO SURROGATE
         .
 5C      DMCTL  (1),PROG,BEQ            Indicate BEQueath
         .
         .
         .
         LA     RO,BEQMSG               Load address of BEQueath data
 6C      DMOUT  (1),(Ø)                 Send data message – No reply required
         .                              PREPARE TO RECEIVE DATA FROM THE PRIMARY
         .
         .
         LA     RO,INPTAREA             Load address of input area
         .
         .
         .
 7C      DMINP  (1),(Ø)                 Get data from primary
         .                              Check for successful read
         .                              IF REPLY IS REQUIRED, SEND REPLY TO PRIMARY
         .
         LA     RO,INPTAREA             Load address of reply area
 8C      DMOUT  (1),(Ø)                 Reply to primary
         .                              Check for successful reply
         .                              Loop to get more input (DMINP performed)
         .                              CHECK WHICH EXCEPTION FLAGS ARE SET IN CDIB
```

Figure 9–9. Job Control and Coding for Example 5 (Part 2 of 3)

BAL Program for the Complex Conversation Primary Program PROGC (cont)

Line			Description
9C	DMCTL	(1),PROG,END	Indicate REPLY END after receiving DEACTivate
	.		
	.		
	.		
	LA	RO,INPTAREA	Load address of reply
	.		
	.		
	.		
10C	DMOUT	(1),(0)	Reply to primary
	.		Check for successful reply
	.		CLOSE CONNECTION TO PRIMARY
	.		
11C	DCLOSE	(1)	Close session path
	.		Check for successful close
	.		
	.		
	EOJ		

Figure 9–9. Job Control and Coding for Example 5 (Part 3 of 3)

9.5.6.3. Example 6: Complex Conversation Surrogate Program Receiving a BEQueath and Repeatedly Transferring Data

Figure 9–10 shows the surrogate program issuing a DOPEN and receiving an ACTivate message from the primary program shown in Figure 9–9. After replying, it receives a BEQueath from the primary program and, as the new primary, sends 500 messages to the new surrogate, checking proper receipt after each 50 messages. It issues a DEACTivate message to the new surrogate program and, after receiving a reply, END, it terminates the conversation.

Job Control Required

```
// JOB SURJOB
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

BAL Program for the Complex Conversation Surrogate Program PROGD

```
Line
 PROGD      START  0                  Description
            BALR   R15,0
            USING  *,R15
                                      LOADING THE CDIB
            LA     R1,SURCDIB             Load address of the following CDIB into R1:
            .                                 SURCDIB  CDIB  FILENAME=SURRGATE
            .                         ATTACH TO THE PRIMARY
            .
            LA     R0,SURRIB              Load address of the following RIB into R0:
                                              SURRIB   RIB USE=PROG,WKFM=VARI

 1D         DOPEN  (1),(Ø)                 Attach to primary session
            .                              Check for successful attachment
            .                         GET ACTIVATE MESSAGE FROM THE PRIMARY
            .
            LA     R0,INPTAREA            Load input area into R0
            .
            .
            .
 2D         DMINP  (1),(Ø)                 Get input from primary
            .                              Check for successful input
            .                         REPLY TO PRIMARY
            .
            LA     R0,INPTAREA            Load input area into R0
```

Figure 9–10. Job Control and Coding for Example 6 (Part 1 of 3)

BAL Program for the Complex Conversation Surrogate Program PROGD (cont)

```
Line                                                    Description
 3D          DMOUT   (1),(Ø)                            Reply to primary
             .                                          Check for successful reply
             .                                          RECEIVE BEQUEATH FROM PRIMARY
             .
             LA      RO,INPTAREA                        Load address of input area
 4D          DMINP   (1),(Ø)                            Get message from primary
             .                                          Check whether BEQueath was specified
             .                                          ENTER DATA TRANSFER LOOP
             .                                          Loop2 is executed 49 times and
                                                        a reply is expected on the 50th message.
   SENDLOOP  LA      R9,1Ø(Ø,Ø)                         Set Loop1 counter (R9)
   LOOP1     LA      R8,49(Ø,Ø)                         Set Loop2 counter (R8)
   LOOP2     MVC     DATAAREA(84),                      Load address of data area
             .       DATAMSG
             .
             .
 5D          DMOUT   (1),(Ø)                    Loop2   Send data w/o request for reply
             .                                          Check for successful send
             .
             .
             BCT     R1,LOOP2                           Loop through 49 times
             .                                  Loop1   SET UP MESSAGES FOR WHICH REPLY IS REQUIRED
             .
             .
             LA      RO,DATAAREA                        Load address of data area
 6D          DMOUT   (1),(Ø),UNLOCK                     Send message and wait for reply
             .                                          Check for successful send
             .                                          GET REPLY FROM SURROGATE AND CHECK VS LAST
             .                                          MESSAGE SENT
             LA      RO,RPLYAREA                        Load the reply area
             .
             .
             .
```

Figure 9–10. Job Control and Coding for Example 6 (Part 2 of 3)

BAL Program for the Complex Conversation Surrogate Program PROGD (cont)

Line			Description
7D	DMINP	(1),(Ø)	Get the reply
	.		Check for successful reply
	.		Loop 1 (cont)
	.		
	BCT	R9,LOOP1	If check positive, loop through 10 times
			TERMINATE CONVERSATION WITH SURROGATE
	LA	RO,=CL7'PRIMJOB'	Load surrogate program name
8D	DMSEL	(1),PROG,(Ø),	Issue DEACTivate
	.	DEACT,DATA	
	.		Check for successful DEACTivate
	.		
	LA	RO,DATAMSG	Load address of DEACTivate message
9D	DMOUT	(1),(Ø),UNLOCK	Issue DEACTivate message
	.		Check for successful DEACTivate
	.		CHECK WHETHER REPLY TO DEACTIVATE MATCHES
	.		MESSAGE SENT TO SURROGATE
	LA	RO,RPLYAREA	Load address of reply area
1ØD	DMINP	(1),(Ø)	Get the reply
	.		Check whether reply was received
	.		CHECK THAT REPLY END WAS RECEIVED
			TERMINATE ATTACHMENT TO SURROGATE
11D	DCLOSE	(1)	Detach from the surrogate
	.		Check for successful detachment
	.		
	.		
	EOJ		

Figure 9-10. Job Control and Coding for Example 6 (Part 3 of 3)

9.5.6.4. Example 7: Primary Program Simultaneously Transferring Data During Complex Conversation with Three Surrogate Programs on Three Different Hosts

Figure 9–11 shows a primary program simultaneously transferring 50 messages to three surrogate programs on three different, remote hosts.

Job Control Stream Required

```
// JOB PTP$PRIM
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

BAL Program for the Complex Conversation Primary Program PROGE

```
Line                                          Description
     PROGE    START  Ø
              .
              .
              .
              BALR   R15,Ø
              USING  *,R15,R12,R11,R1Ø
              .                                BUILD THREE CDIBs FOR PROGRAM-TO-PROGRAM
              .                                OPERATION
              .
     LOOPSTRT EQU    *
              LA     R1,CDIB1                  Load address of CDIB1 into R1
              USING  CD$DCIB,R1
              BLDCDIB FILENAME=PRIME1,MSGSUP=NO
              LA     R1,CDIB2                  Load address of CDIB2 into R1
              BLDCDIB FILENAME=PRIME2,MSGSUP=NO
              LA     R1,CDIB3                  Load address of CDIB3 into R1
              BLDCDIB FILENAME=PRIME3,MSGSUP=NO
```

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 1 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

Line			Description
	.		OPEN A CONNECTION TO EACH SURROGATE
	.		
	.		
	LA	RO,RIB1	Load address of the following RIB1 into RO:
			RIB1 RIB USE=PROG,HOSTID=NOD4
			WKFM=VARI
	LA	R1,CDIB1	Load address of the following CDIB1 into R1:
			CDIB1 CDIB
	DOPEN	(1),(Ø)	Open connection
	.		Check for successful open
	.		
	.		
	LA	RO,RIB2	Load address of the following RIB2 into RO:
			RIB2 RIB
	LA	R1,CDIB2	Load address of the following CDIB2 into R1:
			CDIB2 CDIB
	DOPEN	(1),(Ø)	Open connection
	.		Check for successful open
	.		
	.		
	LA	RO,RIB3	Load address of the following RIB3 into RO:
			RIB3 RIB
	LA	R1,CDIB3	Load address of the following CDIB3 into R1:
			CDIB3 CDIB
	DOPEN	(1),(Ø)	Open connection
	.		Check for successful open
	.		SELECT EACH SUROGATE PROGRAM
	.		
	LA	RO,=CL8'PTP$SRG1'	Load surrogate program name into RO
	LA	R1,CDIB1	Load address of CDIB1 into R1
	DMSEL	(1),PROG,(Ø),	Select surrogate program
	.	ACT,DATA	
	.		Check for successful select
	.		
	LA	RO,=CL8'PTP$SRG2'	Load surrogate program name into RO
	LA	R1,CDIB2	Load address of CDIB2 into R1
	DMSEL	(1),PROG,(Ø),	Select surrogate program
	.	ACT,DATA	
	.		
	.		

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 2 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

Line			Description
	LA	R0,CL8'PTP$SRG3'	Load surrogate program name into R0
	LA	R1,CDIB3	Load address of CDIB3 into R1
	DMSEL	(1),PROG,(Ø),	Select surrogate program
	.	ACT,DATA	
	.		Check for successful select
	.		SET UP DATA FOR ACTIVATE COMMAND
	LA	R0,ACTVMSG1	Load activate data message into R0
	LA	R1,CDIB1	Load address of CDIB1 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue ACTivate
	.		Check for successful ACTivate
	.		
	.		
	LA	R0,ACTVMSG2	Load activate data message into R0
	LA	R1,CDIB2	Load address of CDIB2 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue ACTivate
	.		Check for successful ACTivate
	.		
	.		
	LA	R0,ACTVMSG3	Load address of CDIB3 into R0
	LA	R1,CDIB3	Load address of CDIB3 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue ACTivate
	.		Check for successful ACTivate
	.		ISSUE INPUT REQUEST TO GET REPLY FROM
	.		SURROGATE
	LA	R0,RPYAREA1	Load address of reply area into R0
	.		
	.		
	.		
	LA	R1,CDIB1	Load address of CDIB1 into R1
	DMINP	(1),(Ø)	Wait for surrogate reply
	.		Check for successful reply
	.		
	.		
	LA	R0,RPYAREA2	Load address of reply area into R0
	.		
	.		
	.		
	LA	R1,CDIB2	Load address of CDIB2 into R1

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 3 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

```
Line                                                    Description
          DMINP   (1),(Ø)                               Wait for surrogate reply
          .                                             Check for successful reply
          .
          .
          LA      R0,RPYAREA3                           Load address of reply area into R0
          .
          .
          .
          LA      R1,CDIB3                              Load address of CDIB3 into R1
          DMINP   (1),(Ø)                               Wait for surrogate reply
          .                                             Check for successful reply
          .
          .
                                                        ENTER DATA TRANSFER LOOP
                                                        Loop2 is executed 49 times and a reply
                                                        is expected on the 50th message. The
                                                        reply must match the 49th message.
          LA      R9,1Ø(Ø,Ø)                            Set LOOP1 counter (R9)
LOOP1     LA      R8,49(Ø,Ø)                            Set LOOP2 counter (R8)
LOOP2     AP      MSGCOUNT(4),                          Update message counter
          .       =PL4'1Ø'
          .
          .
          LA      R0,DATAREA1                           Load address of data area into R0
          LA      R1,CDIB1                              Load address of CDIB1 into R1
          DMOUT   (1),(Ø)                               Send data - w/o request for reply
          .                                             Check for successful send
          .                                  Loop1
          .                                  Loop2
          LA      R0,DATAREA2                           Load address of data area into R0
          LA      R1,CDIB2                              Load address of CDIB2 into R1
          DMOUT   (1),(Ø)                               Send data - w/o request for reply
          .                                             Check for successful send
          .
          .
          LA      R0,DATAREA3                           Load address of data area into R0
          LA      R1,CDIB3                              Load address of CDIB3 into R1
```

Figure 9-11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 4 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

```
Line                                          Description
                DMOUT   (1),(Ø)               Send data - w/o request for reply
                .                             Check for successful send
                .                  Loop2
                .                  (cont)
                BCT     R8,LOOP2              Loop through 49 times
                .
                .
                .
                                              SET UP MESSAGE FOR WHICH A REPLY IS EXPECTED
                LA      RO,DATAREA1           Load address of data area into RO
                LA      R1,CDIB1              Load address of CDIB1 into R1
                DMOUT   (1),(Ø),UNLOCK        Send data - request reply
                .                             Check for successful send
                .
                .
                LA      RO,DATAREA2           Load address of data area into RO
                LA      R1,CDIB2              Load of CDIB2 into R1
                DMOUT   (1),(Ø),UNLOCK        Send data - request reply
                .                  Loop 1     Check for successful send
                .                  (cont)
                .
                LA      RO,DATAREA3           Load address of data area into RO
                LA      R1,CDIB3              Load address of CDIB3 into R1
                DMOUT   (1),(Ø),UNLOCK        Send data - request reply
                .                             Check for successful send
                .
                .
                                              GET REPLY FROM SURROGATE CHECK IT AGAINST
                                              LAST OUTPUT MESSAGE SENT
                LA      RO,RPYAREA1           Load address of reply area into RO
                .
                .
                .
                LA      R1,CDIB1              Load address of CDIB1 into R1
                DMINP   (1),(Ø)               Get the reply
                .                             Check for successful reply
                .
                .
```

Figure 9-11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 5 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

Line				Description
	LA	R0,RPYAREA2		Load address of reply area into R0
	.			
	.			
	.			
	LA	R1,CDIB2		Load address of CDIB2 into R1
	DMINP	(1),(Ø)		Get the reply
	.			Check for successful reply
	.			
	.		Loop 1 (cont)	
	LA	R0,RPYAREA3		Load address of reply area into R0
	.			
	.			
	.			
	LA	R1,CDIB3		Load address of CDIB3 into R1
	DMINP	(1),(Ø)		Get the reply
	.			Check for successful reply
	.			
	.			
	BCT	R9,LOOP1		Loop through 10 times
				TERMINATE THE CONVERSATION WITH THE SURROGATE PROGRAM
	LA	R0,=CL8'PTP$SRG1'		Load surrogate program name with a R0
	LA	R1,CDIB1		Load address of CDIB1 into R1
	DMSEL	(1),PROG,(Ø),		Select surrogate program
	.	DEACT,DATA		Check for successful select
	.			
	.			
	LA	R0,=CL8'PTP$SRG2'		Load surrogate program name into R0
	LA	R1,CDIB2		Load address of CDIB2 into R1
	DMSEL	(1),PROG,(Ø),		Select surrogate program
	.	DEACT,DATA		Check for successful select
	.			
	.			
	LA	R0,=CL8'PTP$SRG3'		Load surrogate program name into R0
	LA	R1,CDIB3		Load address of CDIB3 into R1
	DMSEL	(1),PROG,(Ø),		Select surrogate program
	.	DEACT,DATA		Check for successful select
	.			
	.			

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 6 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

Line			Description
			SET UP DATA FOR DEACTIVATE MESSAGE
	LA	RO,DACTMSG1	LOAD address of DEATCTivate message into RO
	LA	R1,CDIB1	Load address of CDIB1 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue DEACTivate
	.		Check for successful DEACTivate
	.		
	.		
	LA	RO,DACTMSG2	Load address of DEACTivate message into RO
	LA	R1,CDIB2	Load address of CDIB2 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue DEACTivate
	.		Check for successful DEACTivate
	.		
	.		
	LA	RO,DACTMSG3	Load address of DEACTivate message into RO
	LA	R1,CDIB3	Load address of CDIB3 into R1
	DMOUT	(1),(Ø),UNLOCK	Issue DEACTivate
	.		Check for successful DEACTivate
	.		
	.		
			CHECK IF REPLY TO DEACTIVATE MATCHES MESSAGE SENT TO THE SURROGATE
	LA	RO,RPYAREA1	Load address of reply area into RO
	.		
	.		
	.		
	LA	R1,CDIB1	Load address of CDIB1 into R1
	DMINP	(1),(Ø)	Wait for the reply
	.		Check whether REPLY (END) was received
	.		
	.		
	LA	RO,RPYAREA2	Load address of reply area into RO
	.		
	.		
	.		
	LA	R1,CDIB2	Load address of CDIB2 into R1
	DMINP	(1),(Ø)	Wait for the reply
	.		Check whether REPLY (END) was received
	.		
	.		

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 7 of 8)

BAL Program for the Complex Conversation Primary Program PROGE (cont)

```
Line                                          Description
          LA      R0,RPYAREA3                 Load address of reply area into R0
          .
          .
          .
          LA      R1,CDIB3                    Load address of CDIB3 into R1
          DMINP   (1),(0)                     Wait for the reply
          .                                   Check whether REPLY (END) was received
          .
          .
                                              CLOSE CONNECTION TO SURROGATE PROGRAM
          LA      R1,CDIB1                    Load address of CDIB1 into R1
          DCLOSE  (1)                         Close the connection
                                              Check for successful close
          LA      R1,CDIB2                    Load address of CDIB2 into R1
          DCLOSE  (1)                         Close the connection
          .                                   Check for successful close
          .
          .
          LA      R1,CDIB3                    Load address of CDIB3 into R1
          DCLOSE  (1)                         Close the connection
          .
          .
          .
          BCT     R7,LOOPSTRT                 Loop for execution count
          EOJ                                 End of job
          CDIB1   DC      XL48'00'
          CDIB2   DC      XL48'00'
          CDIB3   DC      XL48'00'
          RIB1    RIB     USE=PROG,HOSTID=NOD4,WKFM=VARI
          RIB2    RIB     USE=PROG,HOSTID=NOD4,WKFM=VARI
          RIB3    RIB     USE=PROG,HOSTID=NOD4,WKFM=VARI
```

Figure 9–11. Job Control and Coding for Primary Program and Three Surrogate Programs for Example 7 (Part 8 of 8)

9.5.6.5. Example 8: Typical Surrogate Program Used During Complex Conversation with the Primary Program of Example 7

Figure 9–12 shows a typical surrogate program used along with two other surrogate programs simultaneously conversing with the primary program shown in Figure 9–11.

Job Control Stream Required

```
// JOB PTP$SRGn                    (where n is 1, 2, or 3)
      .
      .
      .
// EXEC load module name
      .
      .
      .
/&
// FIN
```

BAL Program for the Complex Conversation Typical Surrogate Program PROGF

```
Line                                        Description
     PROGF    START  Ø
              .
              .
              .
              BALR   R15,Ø
              USING  *,R15
                                        LOADING THE CDIB
              LA     R1,SURCDIB             Load address of the following CDIB into R0:
              .                                 SURCDIB   CDIB   FILENAME=PTP$SRGn
              .
              .
              LA     R0,SURRIB              Load address of the following CDIB into R0:
                                                SURRIB   RIB   USE=PROG,WKFM=VARI
                                        ATTACH TO THE PRIMARY
              DOPEN  (1),(Ø)                Attach to the primary session
                                            Check for successful attachment
                                        GET ACTIVATE MESSAGE FROM PRIMARY
              LA     R0,INPTAREA            Load address of input area into R0
              .
              .
              .
```

Figure 9–12. Job Control and Coding for a Typical Surrogate Program for Example 8 (Part 1 of 3)

BAL Program for the Complex Conversation Typical Surrogate Program PROGF (cont)

```
Line                                                    Description
           DMINP   (1),(Ø)                              Get input from primary
           .                                            Check for successful input
           .
           .
                                                        REPLY TO PRIMARY
           LA      RO,INPTAREA                          Load input area into RO
           DMOUT   (1),(Ø)                              Reply to primary
           .                                            Check for successful reply
           .
           .
                                                        PREPARE TO RECEIVE DATA FROM PRIMARY
RECVLOOP   LA      RO,INPTAREA                          Load address of input area into RO
           .                                  Recvloop
           .                                  Recvloop
           .                                  Recvloop
           DMINP   (1),(Ø)                    Recvloop  Get data from primary
           .                                  Recvloop  Check for successful read
           .                                  Recvloop
           .                                  Recvloop
                                              Recvloop  IF A REPLY IS REQUIRED, SEND REPLY TO PRIMARY
           BE      RECVLOOP                   Recvloop  Continue loop if no flags are set
           .                                  Recvloop
           .                                  Recvloop
           .                                  Recvloop
           LA      RO,INPTAREA                          Load address of reply into RO
           DMOUT   (1),(Ø)                              Reply to primary
           .                                            Check for successful reply
           .                                  Recvloop
           .                                  Recvloop
           BE      RECVLOOP                   Recvloop  Loop to get more input
           .                                  Recvloop
           .                                  Recvloop
           .                                  Recvloop
```

Figure 9–12. Job Control and Coding for a Typical Surrogate Program for Example 8 (Part 2 of 3)

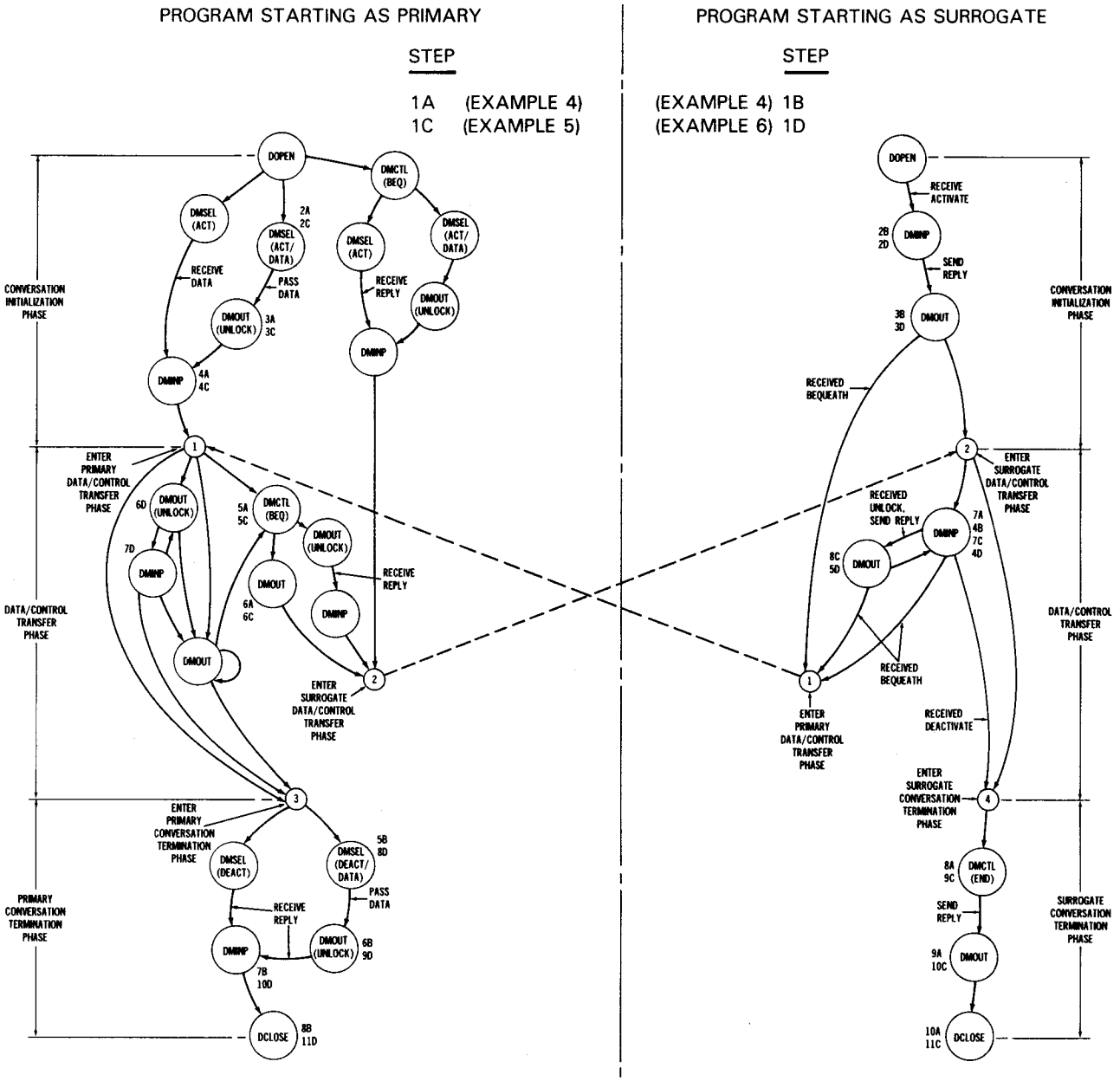
BAL Program for the Complex Conversation Typical Surrogate Program PROGF (cont)

```
Line                                          Description
                                              CHECK WHICH EXCEPTION FLAGS ARE SET IN CDIB
                                                   Check whether DEACTivate was received
                                              REPLY TO THE DEACTIVATE FROM THE PRIMARY
            DMCTL  (1),PROG,END
            .
            .
            .
            LA     R0,INPTAREA                    Load address of reply into R0
            .
            .
            .
            DMOUT  (1),(Ø)                        Reply to primary
            .                                     Check for successful reply
            .
            .
                                              TERMINATE ATTACHMENT TO PRIMARY
            DCLOSE (1)                            Detach from the primary
            .
            .
            .
            EOJ
SURCDIB     CDIB   FILENAME=PTP$SRGn (where n is 1, 2, or 3)
SURRIB      RIB    USE=PROG,WKFM=VARI
INPTAREA    DC     XL'input-area-size'
```

Figure 9–12. Job Control and Coding for a Typical Surrogate Program for Example 8 (Part 3 of 3)

9.5.6.6. Complex Conversation Procedures Flow Diagrams

Figure 9–13 shows all of the procedures available for DDP complex conversation and the possible combinations you might use in designing a communicating program pair. For further description, see subsection 9.5.4.



LEGEND:

Large circle ◯ indicates the imperative macroinstruction shown can be issued.

When the program starting as primary reaches the ② state, it continues from the surrogate diagram state ② .

When the program starting as surrogate reaches the ① state, it continues from the primary diagram state ① .

Figure 9–13. Complex Conversation Procedures Flow Diagrams

PART 3. APPENDIXES

Appendix A. Sample DDP File Copy/ Job Submission Session

The following example shows a typical file copy and job submission from a local to a remote host. We have divided the example into parts for easy explanation.

White print on a black background indicates entries the user makes.

- Part 1: Logging on the system and mounting the disk volume

```
1.  LOGONΔJSMITH
2.  LOGON ACCEPTED date time
3.  DDP TALKΔMESSAGE='PLEASE MOUNT VOLØØ7'ΔUSER=HØØ1::OPERATORΔWAIT
4.  DDPØØ2 CAI ØØ2 TALK COMMAND ACCEPTED WO=ØØØØØ1 1Ø:15:49
5.  DDPØ22 TRM Ø22 TALK COMMAND COMPLETED WO=ØØØØØ1 1Ø:15:59
6.  HØØ1:$Y$CON VOLUME MOUNTED. PROCEED.
```

Explanation:

1. This command attaches our job to the system. JSMITH is our user-id.
2. The LOGON command is accepted at the indicated date and time.
3. We tell the remote operator to mount our disk volume.
4. The DDP TALK command is accepted.
5. The DDP TALK command is completed.
6. The remote operator responds to the DDP TALK command telling us he has mounted our volume.

- Part 2: Allocating file space on the remote host and copying the file

```
1.  DDP CREATE△FILE=H001::',REM/PERS(AXS9/AXSA7),VOL007'△BLOCK_SIZE=182&
2.  RECORD_SIZE=91
3.  DDP002 CAI 002 CREATE COMMAND ACCEPTED WO=000002 10:20:37
4.  DDP022 mmm 022 CREATE COMMAND COMPLETED WO=000002 10:25:51
5.  DDP COPY△FROM=',PERSNNEL(JMS8/)'△TO=H001::',REM/PERS'#&
6.  '(/AXSA7)'''△MODE=WAIT
7.  DDP002 CAI 002 COPY COMMAND ACCEPTED WO=000003 10:37:21
8.  DDP022 mmm 022 COPY COMMAND COMPLETED WO=000003 10:42:39
```

Explanation:

1. We allocate space on our remote host H001 for our file, which we name and REM/PERS. We give it read and write passwords (AXS9 and AXSA7). In
2. the next command, we're going to be copying one of our local files to this file. So we have to establish the same block and record sizes as we have in our local file (182 and 91 characters, respectively). This command uses the defaults for DEVICE—CLASS, FILE—TYPE, INITIAL—SIZE, INCREMENT—SIZE, RECORD—FORM, and REGISTER.

3. The DDP CREATE command is accepted.

4. The DDP CREATE command is completed. Our file is now established and cataloged on the remote host.

5. We copy our local file, PERSNNEL, whose read password is JMS8, on our and remote host H001 in our remote file REM/PERS. If a direct connection
6. isn't possible immediately, we want the system to hold the command until it is. This command uses defaults for the POSITION and TRANSLATE parameters.

   This line also shows the method for breaking a line using character string concatenation.

7. The DDP COPY command is accepted.

8. The DDP COPY command is completed.

- Part 3: Sending the job to the remote host

```
1.  DDP STATUS△JOB=UPDATE
2.  DDPØØ2 CA Ø02 STATUS COMMAND ACCEPTED WO=ØØØØØ4 1Ø:49:Ø1
3.  WAITING FOR I/O
4.  DDPØ22 mmm Ø22 STATUS COMMAND COMPLETED WO=ØØØØØ4 1Ø:52:16
5.  DDP SUBMIT△FILE='PAYØ6,PAYJOBS(JMS17/),,S'△HOST=HØØ1
6.  DDPØØ2 CA Ø02 SUBMIT COMMAND ACCEPTED WO=ØØØØØ5 11:Ø1:45
7.  DDPØ44 mmm mid HØØ1ØØ35 JOB SUBMITTED FOR WO=ØØØØØ5 11:Ø1:52
```

Explanation:

1. We want to find out what has happened to a job we submitted in a previous session.
2. The DDP STATUS command is accepted.
3. The job is being executed and is currently waiting for I/O.
4. The DDP STATUS command is completed.
5. We submit the job in module PAY06, located in library file PAYJOBS, to the remote host H001.
6. The DDP SUBMIT command is accepted.
7. Our job name is H0010035 (renamed by DDP on remote system).
8. We have successfully exited from DDP and can now use our terminal for some other task.

Appendix B. Command Function Summary

Table B–1. Command Function Summary (Part 1 of 4)

To do the following:	Use this command: With these parameters:	Required	See also:
Cancel a job	DDP△CANCEL△JOB=		8.4.2
on a specific host	{host-id / local-host-id}::	No	8.4.2
with a job name of	jobname	Yes	8.4.2
where you want the job's output to be discarded or delivered to you	△OUTPUT={DISCARD / DELIVER}	No	8.4.2
Cancel a command on a specific host	△COMMAND={host-id / local-host-id}		
with work-order-number of	work-order-number		
Copy a file	DDP△COPY△FROM=		8.3.2
on a specific host	{source-host-id / local-host-id}::	No	8.3.2
with a file-id of	source-file-id	Yes	8.3.2
to a host	△TO=	Yes	8.3.2
with a host-id of	{destination-host-id / local-host-id}::	No	8.3.2
to a file with an id of	destination-file-id	Yes	8.3.2
where the source file has an element type	△ELEMENT_TYPE={SYMBOLIC / RELOCATABLE / ABSOLUTE / MACRO / PROC / COMPILED_JOB / SCREEN_FORMAT}	No	8.3.2

Table B-1. Command Function Summary (Part 2 of 4)

To do the following:	Use this command: With these parameters:	Required	See also:
Copy a file	DDP△COPY△FROM=		
with an index that's being changed from the original	△KEY[_ {n / 1}]=(size,location [△{DUPLICATES / NO_DUPLICATES}] [△{CHANGE / NO_CHANGE}])	No No No	8.3.1
with a specification of how the copy is to be performed	△MODE={DIRECT / WAIT / INDIRECT}	No	8.4.2
that should overwrite current file contents (SOF) or add to current file contents (EOF)	△POSITION={EOF / SOF}	No	8.4.2
that needs translating to the host's code	△TRANSLATE={ASCII / EBCDIC / NONE}	No	8.4.2
Create a file	DDP△CREATE△FILE=	Yes	8.3.1
on a specific host	{host-id / local-host-id}::	No	8.3.1
with a file-id of	file-id	Yes	8.3.1
with a specific block size	△BLOCK_SIZE={#-characters / 256}	No	8.3.1
with a specific tape density	△DENSITY={200 / 556 / 800 / 1600 / 6250 / host's-option}	No	8.3.1
on a specific device	△DEVICE_CLASS={DISK / TAPE / DISKETTE}	No	8.3.1
with a specific file type	△FILE_TYPE={SEQUENTIAL / INDEXED / LIBRARY / UNDEFINED}	No	8.3.1
with a size for extention	△INCREMENT_SIZE={#-blocks / 3 cyl}	No	8.3.1
with an initial size	△INITIAL_SIZE={#-blocks / 3 cyl}	No	8.3.1

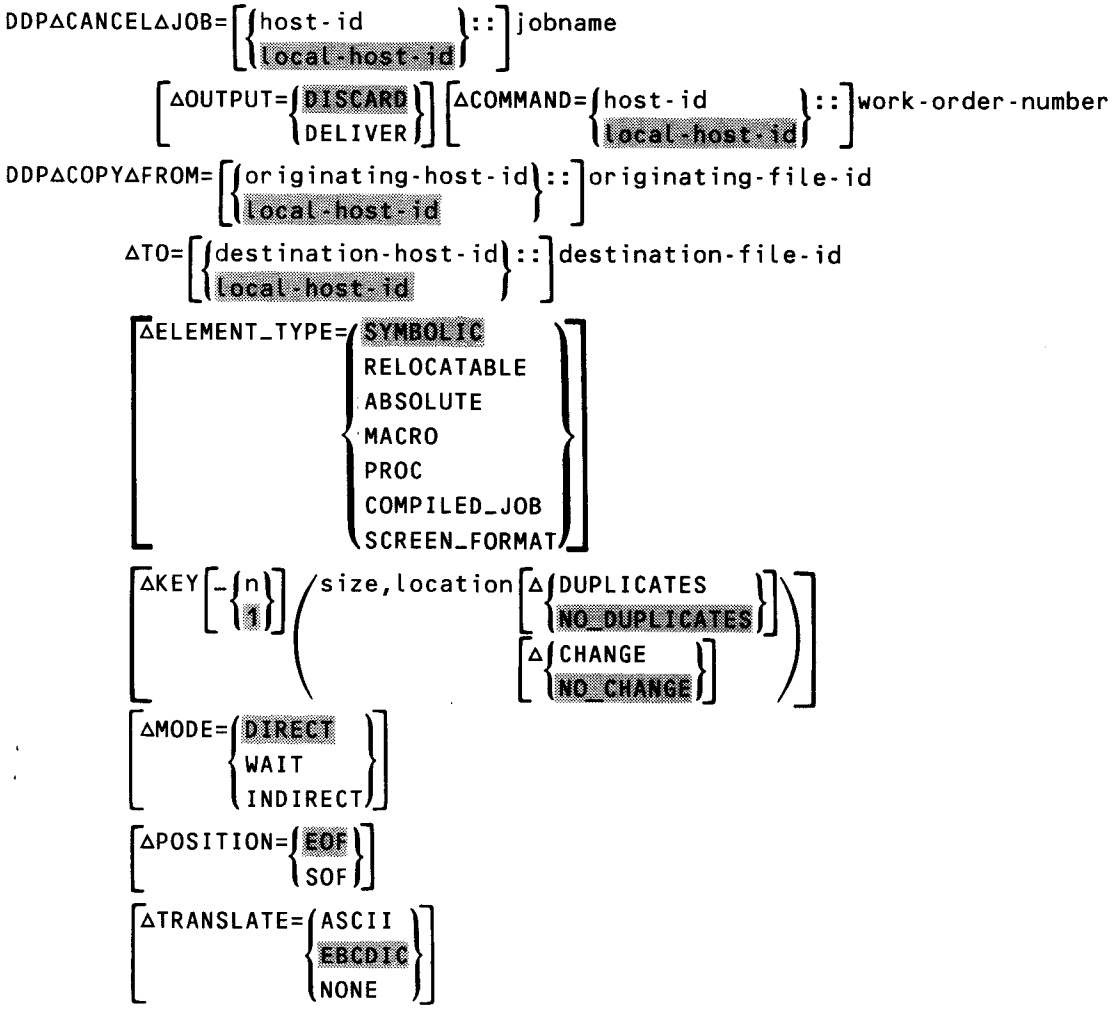
Table B–1. Command Function Summary (Part 3 of 4)

To do the following:	Use this command: With these parameters:	Required	See also:
Create a file	DDP△CREATE△FILE=		
with a new index	△KEY [_{n / 1}] = (size,location [△{DUPLICATES / NO_DUPLICATES}] [△{CHANGE / NO_CHANGE}])	No No No	8.3.1
with a specific tape parity system	△PARITY={ODD / EVEN}	No	8.3.1
with a specific record form	△RECORD_FORM={FIXED / VARIABLE / UNDEFINED}	No	8.3.1
with a specific record size	△RECORD_SIZE={#-characters / 256}	No	8.3.1
that's either cataloged or not	△REGISTER={VTOC / CATALOG}	No	8.3.1
Purge a file	DDP△PURGE△FILE=		
on a specific host	{host-id / local-host-id}::	No	8.3.3
with a file-id of	file-id	Yes	8.3.3
Find out the status	DDP△STATUS△COMMAND=		8.5.1
of a command you entered	work-order-number	No	8.5.1
of a file you control	△FILE=[{host-id / local-host-id}::] file-id [keyword parameter]	No	8.5.1
of a host	△HOST=host-id	No	8.5.1
of a job you entered	△JOB=[{host-id / local-host-id}::] jobname	No	8.5.1
of another user	△USER=[{host-id / local-host-id}::] user-id	No	8.5.1
Submit a job for execution	DDP△SUBMIT△FILE=		8.4.1
from a specific host	{source-host-id / local-host-id}::	No	8.4.1
where the job is located in a specific file	file-id	Yes	8.4.1

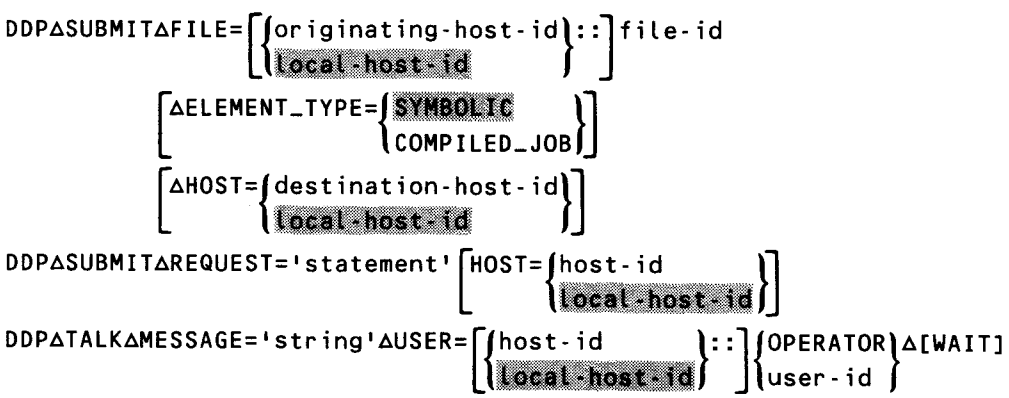
Table B–1. Command Function Summary (Part 4 of 4)

To do the following:	Use this command: With these parameters:	Required	See also:
Submit a job for execution	DDP△SUBMIT△FILE=		
where the job in the file has an element type not specified in the file-id	△ELEMENT_TYPE={SYMBOLIC / COMPILED_JOB}	No	8.4.1
to a specific host	△HOST={destination-host-id / local-host-id}	No	8.4.1
Send a statement	DDP△SUBMIT△REQUEST=		8.4.3
where you make a specific request	statement	Yes	8.4.3
to a specific host	△HOST={host-id / local-host-id}	No	8.4.3
Send a message	DDP△TALK△MESSAGE=		8.5.2
where the message is	'string'	Yes	8.5.2
to a user or operator	△USER=	Yes	8.5.2
located on a specific host	{host-id / local-host-id}::	No	8.5.2
who is either the operator or a user with an id	{OPERATOR / user-id}	Yes	8.5.2
where you want a reply	△WAIT	No	8.5.2

Appendix C. Command Format Summary



```
DDP△CREATE△FILE=[{host-id       }::]file-id
                 [{local-host-id}]
          [△BLOCK_SIZE={number-of-characters/1-9 digits}]
                       {256                           }
          [△DENSITY={200                    }]
                    {556                    }
                    {800                    }
                    {1600                   }
                    {6250                   }
                    {host's-SYSGEN-option   }
          [△DEVICE_CLASS={DISK     }]
                         {TAPE     }
                         {DISKETTE }
          [△FILE_TYPE={SEQUENTIAL }]
                      {INDEXED    }
                      {LIBRARY    }
                      {UNDEFINED  }
          [△INCREMENT_SIZE={number-of-blocks/1-9 digits}]
                           {3 cylinders                }
          [△INITIAL_SIZE={initial-number-blocks/1-9 digits}]
                         {3 cylinders                     }
          [△KEY[_{n}]=(size,location [△{DUPLICATES   }]
                     {1}]            {NO_DUPLICATES}]
                                     [△{CHANGE   }]
                                      {NO_CHANGE}])]
          [△PARITY={ODD  }]
                   {EVEN }
          [△RECORD_FORM={FIXED    }]
                        {VARIABLE }
                        {UNDEFINED}
          [△RECORD_SIZE={maximum-number-of-characters/1-5 digits}]
                        {256                                    }
          [△REGISTER={VTOC   }]
                     {CATALOG}
DDP△PURGE△FILE=[{host-id       }::]file-id
                [{local-host-id}]
DDP△STATUS△{COMMAND=work-order-number
           {FILE=[{host-id       }::]file-id
                 [{local-host-id}]
           {HOST=host-id
           {JOB=[{host-id       }::]jobname
                [{local-host-id}]
           {USER=[{host-id       }::]user-id
                 [{local-host-id}]
```


DDP△SUBMIT△FILE=[originating-host-id::]file-id
local-host-id
[△ELEMENT_TYPE=SYMBOLIC
COMPILED_JOB]
[△HOST=destination-host-id
local-host-id]
DDP△SUBMIT△REQUEST='statement'[HOST=host-id
local-host-id]
DDP△TALK△MESSAGE='string'△USER=[host-id::]OPERATOR△[WAIT]
local-host-id
user-id

Appendix D. Operator's Distributed Data Processing Reference

D. OPERATOR'S DISTRIBUTED DATA PROCESSING REFERENCE (SEPARABLE)

D.1. OVERVIEW

You distribute your data processing when you perform different computer tasks on different independent computers but, at the same time, link those computers together so that they can use each other's resources. There is a series of SPERRY UNIVAC products that make it easy to send jobs, files, transactions, requests for data and communicate via your application program from one computer to another. These products make distributed data processing (DDP) easy to use.

D.2. DISTRIBUTING FILES AND JOBS: THE DDP TRANSFER FACILITY

The first product in the SPERRY UNIVAC DDP line is the DDP transfer facility. It lets you:

- send a message to someone at the remote computer;
- create files on a remote computer;
- copy files from one computer to another;
- purge files on a remote computer;
- send jobs to a remote computer and later cancel them if you need to;
- send a system request to the remote computer; and
- find out the status of a command you entered, a file you control, a host, a user, or a job you submitted.

D.2.1. What You Need to Do for DDP Transfer Facility Users on Your Computer

If you have a user on your system who wants to use the DDP transfer facility, you have to make sure that the following are all true:

- Spooling is configured.
- ICAM is loaded, and the ICAM network includes the remote computer your DDP user wants to work with.
- Interactive services is configured.

The DDP software takes care of everything else.

D.2.2. How DDP Transfer Facility Users Get Their Output

When your DDP users have jobs performed at a remote computer, the remote computer sends the output back to your spool file. From there it's printed or punched, the same as any other output in the spool file. You should distribute it according to your current procedures.

D.2.3. Requests to You from Remote DDP Users

The DDP command language contains a DDP TALK command that lets remote DDP users send messages directly to you. They may want you, for instance, to mount a disk that they're going to be using or to remove that disk once they're done. You can respond to them using the DDP TALK command also. See D.2.4 for information on how to enter the DDP software, and D.2.6 for the format of the DDP TALK command.

D.2.4. Entering and Leaving the DDP Transfer Software from the Console

If your terminal, workstation, or system console is connected to a DDP network (see 1.3, 3.5, and 8.2), you enter DDP software whenever you issue a DDP software command and exit whenever your DDP command terminates.

D.2.5. Special Terms in DDP Command Parameters

D.2.5.1. Host-id

Each host in your DDP network has a host identification (host-id). This is the same identification as the node-id you specify in the REMOTE parameter of the LOCAP macroinstruction in your ICAM network definition.

The host-id is one to four alphanumeric characters long. The first character must be alphabetic.

D.2.5.2. File-id

The DDP transfer facility uses a specific form of file identification (file-id). It has a maximum of 74 characters and is always expressed as a character string (in apostrophes).

Format:

```
'[module-name],filename[([read-passwrd]/[write-passwrd])],[vol],[element]'
```

where:

`module-name`
: Is used only when you're referencing just one module from a file. It is the one to eight alphanumeric characters identifying the module. It must begin with an alphabetic character.

`filename`
: Is always required. It is the same as the name on the // LBL job control statement for the program. For disk files, it is 1 to 44 alphanumeric characters long. For tape files and logical files (spool files), it is 1 to 17 alphanumeric characters long.

`read-passwrd`
: Is a password (one to six alphanumeric characters) enabling you to read the file. If omitted, the system assumes there is no read password on the DDP command. If the file does have a read password that you omit, you won't be able to read the file. Use only with cataloged files.

`write-passwrd`
: Is a password (one to six alphanumeric characters) enabling you to write to the file. If omitted, the system assumes there is no write password on the DDP command. If the file does have a write password that you omit, you won't be able to write to the file. Use only with cataloged files.

`vol`
: Is the volume serial number (one to six alphanumeric characters) of the volume on which your file resides. Specify this when the file is not cataloged.

`element`
: Specifies the type of code contained in the program module you're sending. In OS/3, we'd normally call this "type" or "module-type". But since DDP is designed eventually to connect you with non-OS/3 hosts, we're using the word "element".

  The element types are the same ones you're familiar with for all OS/3 library files. SAT files may have the following types:

  - S (source code)
  - M (macro)
  - P (procedure)
  - L (load code)
  - O (object code)

MIRAM files may have the following types:

- F or FC (screen format modules)
- J (saved job control stream)
- MENU (menu modules)
- HELP (help screen modules)

The default is S (source).

For MIRAM files, you may create your own element type and identify it with one to four characters. (For further information on MIRAM files, see consolidated data management concepts and facilities, UP-8825 (current version).

Table D–1 gives some examples of complete file-ids.

Table D–1. Examples of File-ids

Type of File	Example
A cataloged file with no read and write passwords	',SITE/3'
A cataloged file with read and write passwords	',SITE/3(XX2674/BENNIS)'
A module from a cataloged file with read and write passwords	'PROG14,JOBFIL19(XX4982/RILEY)'#& ',,S'
An uncataloged file with a write password but no read password	',SITE/A(SMPRO),VSN149'
An uncataloged file with a read password but no write password	',SITE/A&B(READ/),296410'

D.2.5.3. User-id

The user-id is one to six alphanumeric characters identifying you as a user to the host.

D.2.5.4. Job Name

When you enter a DDP SUBMIT FILE command, DDP returns a job name to you with which you can cancel the job if you later need to. DDP renames each job with a unique name with the format xxxxyyyy

where:

xxxx
: Is the four-character host-id of the host that initiated the command.

yyyy
: Is the four-digit unique number.

If you submit a file containing many jobs, DDP returns each job name to you in a separate message.

The format for the job name message is:

```
DDP004 EJS 044 xxxxyyyy JOB SUBMITTED FOR WO=work-order-number time
```

See D.2.5.5 for an explanation of work order number.

D.2.5.5. Work Order Number

Once DDP accepts your command, it returns a work order number to you using the format:

```
DDP002 CAI 002 cccccccccc COMMAND ACCEPTED WO=nnnnnn time
```

where:

cccccccccc
: Is the name of the command you've just entered.

nnnnnn
: Is the work order number of the command.

For instance, let's say you've just entered the DDP COPY command. DDP responds:

```
DDP002 CAI 002 COPY COMMAND ACCEPTED WO=A48107 13:46:02
```

Write down the work order number, since error messages use it for identification. If, for example, you enter a command that cannot be completed at the remote host, DDP informs you of the problem by sending you a message such as:

```
DDP001 ELH 023 COPY COMMAND ABORTED WO=A48107 13:48:39
```

The WO=A48107 is the work order number that DDP returned to you at the time you entered your copy command. You may have entered several copy commands, but each has a different work order number, so you can tell them apart.

D.2.6. Communicating with DDP Users (DDP TALK)

Function:

The DDP TALK command lets you send a message to a remote DDP user or to the operator of a remote system.

Format:

```
DDPΔTALKΔMESSAGE='string'ΔUSER=[{host-id      }::][{OPERATOR}Δ[WAIT]
                               [{local-host-id}]  [{user-id }]
```

Positional Parameters:

`string`
Indicates a character string message you want the operator or user to get. Enclose it in apostrophes and double all enclosed apostrophes.

`host-id`
Is one to four alphanumeric characters naming the host where the operator or user is. See D.2.5.1 for additional details.

`OPERATOR`
Specifies the operator of the remote host.

`user-id`
Is one to six alphanumeric characters identifying the user to the system.

`WAIT`
Informs the operator or user that you want a reply. You do not, however, have to wait for this reply to go on with other commands. Your keyboard isn't locked during this period.

If you receive a message with a WAIT indicated, you should reply as soon as possible.

Example:

Operator keyin:

```
DDP TALK MESSAGE='VOLØØ7 MOUNTED' USER=HØØ1::CJONES
```

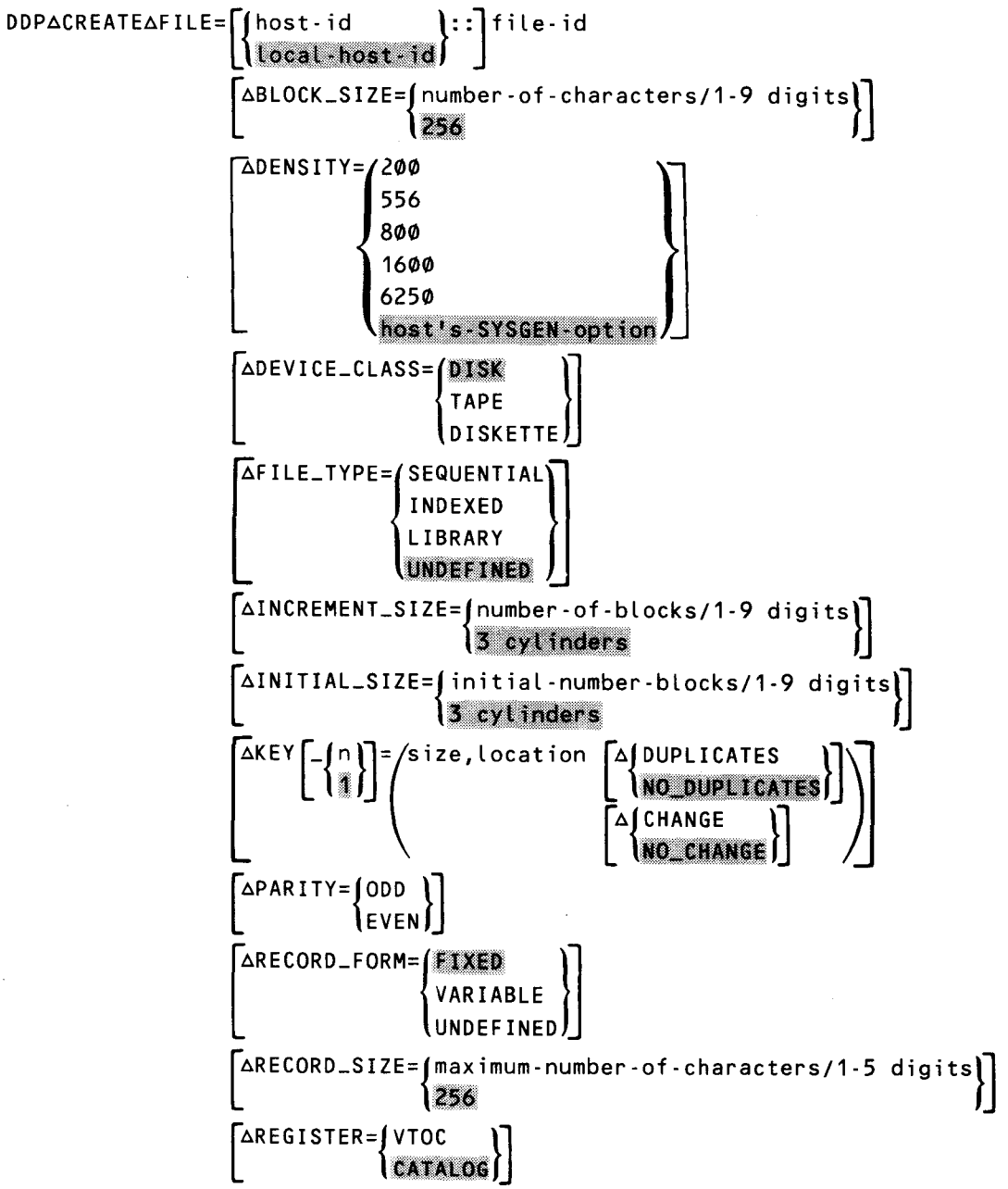
User CJONES on host H001 has previously sent you a message asking to have VOL007 mounted. You reply that you have done so.

D.2.7. Creating a File on a Remote Host (DDP CREATE)

Function:

Use the DDP CREATE command to allocate space for a file on a remote host.

Format:



Positional Parameters:

`host-id`
Is one to four alphanumeric characters that name the host you create the file on. See D.2.5.1 for additional details.

`file-id`
Is 1 to 74 alphanumeric characters identifying your file. See D.2.5.2 for additional details.

Keyword Parameters:

`BLOCK_SIZE=`{`number-of-characters/1-9 digits` | **`256`**}
Is the number of characters in a block.

`DENSITY=`{**`200`** | `556` | **`800`** | `1600` | `6250` | **`host's-SYSGEN-option`**}
Is the tape density in bytes per inch (bpi).

`DEVICE_CLASS=`{**`DISK`** | `TAPE` | `DISKETTE`}
Is the device class that will contain your file.

`FILE_TYPE=`{`SEQUENTIAL` | `INDEXED` | `LIBRARY` | **`UNDEFINED`**}
Is the type of file being created.

`INCREMENT_SIZE=`{`number-of-blocks/1-9 digits` | **`3 cylinders`**}
Is the number of blocks of storage to be added to the file whenever you extend its size.

`INITIAL_SIZE=`{`initial-number-of-blocks/1-9 digits` | **`3 cylinders`**}
Is the number of blocks initially allocated to the file.

```
KEY[-{n }]=(size,location [ {DUPLICATES   }] )
     [{1 }]                [ {NO_DUPLICATES}]
                           [ {CHANGE       }]
                           [ {NO_CHANGE    }]
```

Creates an index for the file on the receiving host. Specify one KEY—n parameter for each index key (to a maximum of five) you create.

```
PARITY={ODD }
       {EVEN}
```

Is the parity system for tape files.

```
RECORD_FORM={FIXED     }
            {VARIABLE  }
            {UNDEFINED }
```

Specifies whether record length is fixed or variable.

```
RECORD_SIZE={maximum-number-of-character/1-5 digits}
            {256                                   }
```

Is the maximum number of characters in a record.

```
REGISTER={VTOC   }
         {CATALOG}
```

Either catalogs your file or registers it in the volume table of contents.

Example:

Operator keyin:

```
DDP CREATE FILE=H00l::',REM/PERS(AXS9/AXSA7),VOL007' BLOCK_SIZE=182&
RECORD_SIZE=91
```

Creates a cataloged file on host H001, disk volume VOL007, that has read and write passwords, a block size of 182, and a record size of 91. Records are fixed. The file is smaller than three cylinders. FILE TYPE is UNDEFINED (the default).

D.2.8. Copying Files or Modules (DDP COPY)

Function:

The DDP COPY command duplicates either an entire file or a module from a file on another system.

Format:

```
DDP△COPY△FROM=[{originating-host-id}::]originating-file-id
                {local-host-id     }
        △TO=[{destination-host-id}::]destination-file-id
             {local-host-id     }
[△ELEMENT_TYPE={SYMBOLIC    }]
               {RELOCATABLE }
               {ABSOLUTE    }
               {MACRO       }
               {PROC        }
               {COMPILED_JOB}
               {SCREEN_FORMAT}
[△KEY[-{n}]=(size,location [△{DUPLICATES   }])]
       {1}                 [△{NO_DUPLICATES}]
                           [△{CHANGE    }]
                           [△{NO_CHANGE }]
[△MODE={DIRECT  }]
       {WAIT    }
       {INDIRECT}
[△POSITION={EOF}]
           {SOF}
[△TRANSLATE={ASCII }]
            {EBCDIC}
            {NONE  }
```

Positional Parameters:

`originating-host-id`
Is one to four alphanumeric characters naming the originating host. See D.2.5.1 for additional details.

`originating-file-id`
Is 1 to 74 alphanumeric characters identifying the input file. See D.2.5.2 for additional details.

`destination-host-id`
Is one to four alphanumeric characters naming the host to receive the information. See D.2.5.1 for additional details.

`destination-file-id`
Is 1 to 74 alphanumeric characters identifying your file. See D.2.5.2 for additional details.

Keyword Parameters:

```
ELEMENT_TYPE=(SYMBOLIC
              RELOCATABLE
              ABSOLUTE
              MACRO
              PROC
              COMPILED_JOB
              SCREEN_FORMAT)
```

Is the element type of the module you are copying.

```
KEY[-{n}]=(size,location [{DUPLICATES   }] [{CHANGE   }])
   [ {1} ]               [{NO_DUPLICATES}]  [{NO_CHANGE}]
```

Changes an index for the file on the receiving host. Specify one KEY—n parameter for each index key in the record (to a maximum of five), even if you change only one.

```
MODE=(DIRECT
      WAIT
      INDIRECT)
```

Specifies whether the device or medium needed to copy the file at the destination host must be immediately available (DIRECT), whether you want to wait until it is available (WAIT), or whether you want to copy the file into a temporary file on the destination host if the device or medium needed to receive the copy is not available (INDIRECT).

```
POSITION=(EOF
          SOF)
```

Specifies overwriting (SOF) or extending (EOF) of the destination file.

```
TRANSLATE=(ASCII
           EBCDIC
           NONE)
```

Indicates the character code you want the file translated to as it arrives at the destination host.

Example:

Operator keyin:

```
DDP△COPY△FROM=',PERSNNEL(JMS8/)'△TO=HØØ1::',REM/PERS(/ASXA7)'△MODE=WAIT
```

Copies file PERSNNEL into the blank cataloged file REM/PERS on host H001, which uses EBCDIC. If a direct connection isn't possible, you want the system to hold the command until it is.

D.2.9. Purging Files (DDP PURGE)

Function:

The DDP PURGE command physically removes a file or element and all references to it from a host.

Format:

```
DDP△PURGE△FILE=[{host-id      }::]file-id
                {local-host-id}
```

Positional Parameters:

`host-id`
Is one to four alphanumeric characters naming the host you are purging the file from. See D.2.5.1 for additional details.

`file-id`
Is 1 to 74 alphanumeric characters identifying the input file. See D.2.5.2 for additional details.

Example:

Operator keyin:

```
DDP PURGE FILE=HØØ1::',REM/PERS(AXS9/AXSA7)'
```

Purges cataloged file REM/PERS on host H001.

D.2.10. Sending Jobs to a Remote Host (DDP SUBMIT FILE)

Function:

The DDP SUBMIT FILE command sends a file of jobs or a module to a remote host for execution.

Format:

```
DDP▵SUBMIT▵FILE=[{originating-host-id}::]file-id
                [{local-host-id     }::]
                [▵ELEMENT_TYPE={SYMBOLIC    }]
                               {COMPILED_JOB}
                [▵HOST={destination-host-id}]
                       {local-host-id     }
```

Positional Parameters:

`originating-host-id`
Is one to four alphanumeric characters naming the job control stream file location. See D.2.5.1 for additional details.

`file-id`
Is 1 to 74 characters identifying your file. See D.2.5.2 for additional details.

Keyword Parameters:

`ELEMENT_TYPE={SYMBOLIC    }`
`             {COMPILED_JOB}`
Is the element (module) type of the submitted file. SYMBOLIC indicates source code. COMPILED—JOB indicates compiled job streams with expanded jprocs. You may omit this parameter if the element type is SYMBOLIC (the default) or if the element type is specified in the file name. Normally, you won't be using COMPILED—JOB.

`HOST={destination-host-id}`
`     {local-host-id     }`
Is one to four alphanumeric characters naming the host you submit the file to. If omitted, the command assumes the file is to be submitted to your local system. For more information about the host-id, see D.2.5.1.

Example:

Operator keyin:

```
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/),,S' HOST=H001
```

Submit the source code job in module PAY06 of file PAYJOBS to remote host H001.

Alternate operator keyins to accomplish same task:

```
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/)' ELEMENT_TYPE=SYMBOLIC HOST=H001
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/)' HOST=H001
```

System Response:

```
DDP044 JNR 044 H0020038 JOB SUBMITTED FOR WO=000012 13:47:56
```

Write down the job name (H0020038) and its work order number (000012). You need them to cancel the job or to inquire as to its status.

D.2.11. Cancelling a Job or Command (DDP CANCEL)

Function:

The DDP CANCEL command terminates an executing or backlogged job or a command executing on a host.

Format:

```
DDP△CANCEL△JOB=[{host-id       }::]jobname
               [{local-host-id }]
               [△OUTPUT={DISCARD}]
               [        {DELIVER}]
               [△COMMAND={host-id       }::]work-order-number
                         {local-host-id }]
```

Positional Parameters:

`host-id`
Is one to four alphanumeric characters naming the host that the job is executing on. See D.2.5.1 for additional details.

`jobname`
Is the one to eight alphanumeric characters naming your job. See D.2.5.4 for additional details.

Keyword Parameters:

OUTPUT={DISCARD / DELIVER}

Specifies whether you want the spooled output from the job erased from the spooled output files (DISCARD) or sent to the originating host (DELIVER).

COMMAND={host-id / local-host-id}

Specifies the command that you entered but now wish to cancel.

work-order-number

Is the one to six alphanumeric work-order-number that was displayed to you when command was accepted.

Example:

Operator keyin:

```
DDP CANCEL JOB=H001::H0020038 OUTPUT=DELIVER
```

Cancels job H0020038 on host H001. Any spooled output is returned to the originating host.

D.2.12. Sending a Request to a Remote Host (DDP SUBMIT REQUEST)

Function:

The DDP SUBMIT REQUEST command lets you send an instruction for the remote host to perform some task.

Format:

DDP△SUBMIT△REQUEST='statement' [HOST={host-id / local-host-id}]

Positional Parameters:

statement

Is the system command or message being submitted. Normally it is a character string enclosed in apostrophes.

Keyword Parameters:

HOST=host-id

Is one to four alphanumeric characters naming the destination host. See D.2.5.1 for additional details.

Example:

Operator keyin:

```
DDP SUBMIT REQUEST='RU CGV,,O=VSN999,N=186309,T=30' HOST=N852
```

Submits a canned job control stream to host N852 to change the volume serial number on a disk.

D.2.13. Finding Out the Status of a DDP Command, Host, User, File, or Job (DDP STATUS)

Function:

You can find out the status of a command you entered, a host in your system, a user, a file, or a job, by using the DDP STATUS command.

Format:

```
DDP△STATUS△{ COMMAND=work-order-number
           { FILE=[{host-id      }::]file-id[keyword parameter]
           {      {local-host-id}
           { HOST=host-id
           { JOB=[{host-id      }::]jobname
           {     {local-host-id}
           { USER=[{host-id      }::]user-id
           {      {local-host-id}
```

Keyword Parameters:

`COMMAND=work-order-number`

Tells you the status of a command you entered. The response tells you the originating (primary) host, the destination (secondary) host, the time the command was sent, the time it was completed (if it was completed), and status information such as BEING PROCESSED, IN SCHEDULER, or COMPLETED. See D.2.5.5 for additional details about the work order number.

`FILE=[{host-id      }::]file-id`
`      {local-host-id}`

Tells you the status of a library file or module on a host. See D.2.5.1 for additional details about the host-id.

`HOST=host-id`

Tells you how busy a host is by showing you the number of users (both local and remote), the number of buffers DDP is using and their size, and the number of DDP work orders currently being processed. See D.2.5.1 for additional details about the host-id.

JOB=[{host-id / local-host-id}::]jobname

Tells you job status, such as COMPLETED, BEING PROCESSED, or IN SCHEDULER. See D.2.5.1 for additional details about the host-id. See D.2.5.4 for additional details about the jobname. For restrictions, see 8.5.1.

USER=[{host-id / local-host-id}::]user-id

Lists a table of the most recent DDP functions for a particular user. At a minimum, all commands that aren't completed are listed. In addition, some completed commands are listed. See D.2.5.1 for additional details about the host-id. See D.2.5.3 for additional details about the user-id.

As the system console operator, you may request the status of any user in the system.

Example:

Operator keyin:

```
DDP STATUS HOST=H00001
```

Inquires about how busy host H001 is.

Appendix E. Command Requirements

Certain DDP commands and parameters require the use of other commands or parameters. This table describes those requirements.

Table E–1. Command Requirements (Part 1 of 2)

If you specify:	You must also specify:
DDP△CANCEL JOB=	a job name for a job that you submitted through the SUBMIT FILE command with the same user-id
COMMAND=	a command that you submitted with the same user-id that you wish to cancel
DDP△COPY FROM=	originating-file-id destination-file-id TO=
file-id that includes a nonsource module name	file-id must also include element type or you must specify the ELEMENT TYPE= parameter
MODE=INDIRECT	file must be cataloged
For additional restrictions on this command, see Tables 8–3 and 8–4.	
DDP△CREATE FILE=	file-id
BLOCK_SIZE=	either INITIAL_SIZE or INCREMENT_SIZE or both
DEVICE_CLASS=DISKETTE	BLOCK_SIZE=256 or less RECORD_SIZE=256 or less
INCREMENT_SIZE=	BLOCK_SIZE=
INITIAL_SIZE=	BLOCK_SIZE=
KEY[_n]=	either FILE_TYPE=INDEXED or FILE_TYPE= UNDEFINED (if you're going to copy an indexed file into this file)
PARITY=	DEVICE_CLASS=TAPE
RECORD_FORM=UNDEFINED	DEVICE_CLASS=TAPE
REGISTER=VTOC	file-id with a volume serial number included for all subsequent commands referring to this file

Table E–1. Command Requirements (Part 2 of 2)

If you specify:	You must also specify:
DDP△PURGE FILE=	file-id
DDP△SUBMIT FILE=	file-id
file-id that includes module name	file-id must also include element type or you must specify the ELEMENT_TYPE= parameter
DDP△SUBMIT REQUEST=	statement being submitted
DDP△TALK MESSAGE=	string being sent USER=

Appendix F. Entering DDP Commands in a Batch Stream

F.1. OVERVIEW

You usually use DDP as an interactive product. But you can submit DDP commands as a remote batch job. The advantage of doing so is that you can perform remote tasks using these relatively simple commands during periods when you can't be at the terminal, such as overnight. The disadvantage is that you don't get your messages and output as promptly as you would if you were working interactively. Instead, all messages and output are printed at your local host for you to pick up.

F.2. CARD FORMAT

The following table shows the format for the cards required to enter your DDP task as a batch job.

Table F–1. Entering DDP Commands in a Batch Stream (Part 1 of 2)

Card Required	Card Format	Explanations
Yes	`// DATA FILEID=name`	`name` The name through which you reference this enter stream. It is one to eight alphanumeric characters long.

Yes	`LOGON user-id,BU={YES|NO}`	`user-id`
		The one to six alphanumeric characters identifying you as a user to the host.
		
		`BU={YES|NO}`
		BU=YES prints the day's bulletin on your output. The default is NO. Although you can't change your DDP commands as a result of knowing the bulletin, knowing it may explain output errors.

Table F–1. Entering DDP Commands in a Batch Stream (Part 2 of 2)

Card Required	Card Format	Explanations
Yes	DDP command cards	Type these on the cards exactly as you would type them in at your terminal. Because this is an enter stream, your DDP commands are executed in the order in which you list them. Each command is completed before the next is started. (This is different from terminal operation, where you can start your next DDP command before the previous one is complete.) Therefore, it's all right to list commands that depend on previous commands. For instance, you can first create a file and then copy to it. DDP won't attempt to copy before it creates the file. NOTE: Operations that proceed in a local-to-remote direction are done serially. However, operations in a remote-to-local, or remote-to-remote direction, are merely routed to the remote host for processing. As soon as routing is complete, the next command is executed. Care should be taken when performing these operations from an enter stream, since concurrent command execution may result.
Yes	LOGOFF	
Yes	// FIN	

F.3. PROCEDURES

Follow these steps to enter your DDP commands as a remote batch job:

1. Place cards in the card reader.

2. At the system console, enter: IN

   A message appears on the system console indicating that the file you specified on your FILEID parameter in the // DATA statement has been created.

3. At the system console, enter:

```
ENTER△Q=RDR,HOLD={YES},FIL=name
                 {NO }
```

where:

HOLD={YES / NO}
: Indicates whether the enter stream is to be retained by the host (HOLD=YES) or deleted once the stream has been processed (HOLD=NO). The default is HOLD=NO.

name
: Is the reference name of the enter stream as indicated in your FILEID parameter in the // DATA statement.

Enter streams can also be created through the OS/3 editor and saved as source modules in a user library. They can be executed by typing the following command:

```
ENTER module-name,type,filename,vsn
```

For parameter specifications of the ENTER command, see the OS/3 general editor user guide/programmer reference, UP-8828 (current version).

F.4. SAMPLE ENTER STREAM

Here are the cards you'd use to enter the sample session from Appendix A in a remote batch enter stream. See Appendix A for explanations of any commands you don't understand.

```
// DATA FILEID=ARCHER
LOGON JSMITH,BU=YES
DDP TALK MESSAGE='PLEASE MOUNT VOL007' HOST=H001::OPERATOR
DDP CREATE FILE=H001::',REM/PERS(AXS9/AXSA7),VOL007' BLOCK_SIZE=182 RECORD_SIZE=91
DDP COPY FROM=',PERSNNEL(JMS8/)' TO=H001::',REM/PERS(/AXSA7)' MODE=WAIT
DDP SUBMIT FILE='PAY06,PAYJOBS(JMS17/),,S' HOST=H001
LOGOFF
// FIN
```

Appendix G. Logging Chart

All DDP local and remote activity is automatically logged in the Interactive Services log file during shutdown processing. The log printout is normally sent to the central site printer.

The following chart can be used to list your system assigned job number and any other pertinent data from your displayed log for accounting purposes.

DDP LOG SHEET

TERMINAL: _____

User-id	Date	Command	Time Accepted	Work Order Number	Jobname	Remote Host ID	Time Completed	Time Aborted

Appendix H. Generating Your ICAM Network with DDP

DDP requires ICAM's demand mode interface (DMI) and distributed communications architecture (DCA). Your host-ids appear in the REMOTE parameter of the LOCAP macroinstruction. No other adjustments are required when you define your ICAM network.

Here is a sample ICAM network definition that you could use with DDP. It's a very simple one, with just two hosts and one user connected to one of the hosts on a terminal. For further information on using ICAM and DMI, see current versions of integrated communications access method (ICAM) concepts and facilities, UP-8194, and the integrated communications access method (ICAM) network definition and operations user guide, UP-8947.

```
COMMCT
ACT1     CCA    TYPE=(GBL,,A),DCA=YES,GAWAKE=YES,CCAID=ACT1
         BUFFERS 24,256,2,ARP=36,UDUCT=(6,24,2),LINKPAK=(45,80,2)
LNE1     LINE   DEVICE=(UNISCOPE),TYPE=(9600,UNAT,SWCH,SYNC),ID=4
ATRM     TERM   FEATURES=(U400,1920),ADDR=(21,51),AUX1=(COP,21),          X
                HIGH=MAIN,LOW=MAIN,MEDIUM=MAIN,INPUT=(YES),TCTUPD=YES
CUP1     LOCAP  TYPE=(DMI),MT=YES,IAS=(YES,OFF)
VLN1     VLINE  DEVICE=(ABM,PRIMARY),TYPE=(9600),ID=7
         LPORT  LINE=VLN1,REMOTE=(B),PORT=1,EU1=CUP1,EU2=CUP2,
                USERTP=(DMI),NUMSS=1
BTRM     TERM   FEATURES=(U400,1920),ADDR=(21,51),AUX1=(COP,21),          X
                HIGH=MAIN,LOW=MAIN,MEDIUM=MAIN,INPUT=(YES),              X
                TCTUPD=YES,REMOTE=(B)
CUP2     LOCAP  TYPE=(DMI),MT=YES,IAS=(YES,OFF),REMOTE=(B)
         ENDCCA
         MCP
                MCPVOL=REL071
                MCPNAME=C1
         CACH=(04,9600,SWITCHED,SYNC)
         CACH=(07,9600,FULL,ILA)
END
```

← This is your local host, whose id is A

← This is your local terminal

← Here's where you specify DMI

← This is your remote host, whose id is B

Appendix I. DDP Program-To-Program Macroinstructions Summary

The following (Table I–1) is a summary of the program-to-program macroinstructions used with the OS/3 DDP file access facility.

Table I–1. DDP Program-To-Program Macroinstructions Summary (Part 1 of 3)

To do the following	Issue this imperative	Reference
Establish a communications path between Primary IPC and Destination IPC	DOPEN {(1) / cdibname} , {(Ø) / ribname} where: cdibname Address of user's CDIB (1) CDIB address is preloaded into register 1. ribname Address of user's RIB (Ø) RIB address is preloaded into register 0.	9.4.4.3
Begin and end a conversation between paired programs	DMSEL {(1) / cdibname} ,PROG, {(Ø) / surrogate-prog} , {ACT / DEACT} [,DATA] where: cdibname Address of user's CDIB (1) CDIB address is preloaded into register 1.	9.5.3.1

Table I–1. DDP Program-To-Program Macroinstructions Summary (Part 2 of 3)

To do the following	Issue this imperative	Reference
Begin and end a conversation between paired programs (cont)	**PROG** Identifies the imperative belongs to the DDP file access facility. **surrogate-prog** Name of the program with which conversation is to be initiated (used with ACT option only). **(Ø)** Assumes address of name of program is preloaded into register 0. **ACT** Indicates Primary program conversation with Surrogate program is to be activated. **DEACT** Indicates Primary program conversation with Surrogate program is to be deactivated (closed). **DATA** Indicates next imperative contains data to be passed with ACT or DEACT IPC function.	
Transfer data	{DMOUT / DMINP},{(1) / cdibname},{(Ø) / workarea}[,UNLOCK] where: **DMOUT** Used to send data **DMINP** Used to receive data **cdibname** Address of user's CDIB **(1)** CDIB address is preloaded into register 1. **workarea** Address of data buffer that can be fixed or variable according to the WKFM parameter description in the RIB.	9.4.4.4 9.4.4.5

Table I–1. DDP Program-To-Program Macroinstructions Summary (Part 3 of 3)

To do the following	Issue this imperative	Reference
Transfer data (cont)	**(Ø)** Buffer address is preloaded into register 0. **UNLOCK** Indicates reply required from Surrogate program (only issuable by Primary program)	
Pass conversation control from Primary program to Surrogate program	**DMCTL** {(1) / cdibname}, **PROG**, {**BEQ** / **END**} where: **cdibname** Address of user's CDIB **(1)** CDIB address is preloaded into register 1. **PROG** Identifies the imperative belongs to the DDP file access facility. **BEQ** Indicates conversation control is to be passed to the Surrogate program. **END** Indicates an end to the conversation (only a DMOUT imperative can follow; indicates a reply is expected by the Primary program; usable only in TWA mode by the Surrogate program).	9.5.3.2
Close the conversation	**DCLOSE** {(1) / cdibname} where: **cdibname** Address of user's CDIB **(1)** CDIB address is preloaded into register 1.	9.4.4.6

Glossary

A

absolute code
Load code.

application program
An assembly language program, written by the user to "converse" or communicate with one or more communicating application programs on OS/3 remote hosts in a DDP environment, furnished with the OS/3 DDP file access facility. See Section 9.

B

BEQ
A parameter of the DMCTL imperative, which indicates the primary program wishes to bequeath primary status to the surrogate program. The primary program becomes the surrogate program after the solicited response has been received from the surrogate.

C

catalog
An online directory of information containing file names with their passwords that enables easy access to the files and also restricts files to certain users.

CDM
See consolidated data management.

centralized data processing (computer) system
A method of data processing in which a large processing unit accommodates the varied needs of many users. The centralized computer may have many nonintelligent terminals and peripherals.

command
An action performed on a computer.

command analyzer
The part of the DDP software that analyzes your commands and forwards them for processing.

compiled job
A series of expanded job control statements in a system library that can be executed immediately.

complex conversation
A 2-way alternating communications discipline whereby primary and surrogate programs may send and receive data. The primary and surrogate programs may reverse statuses. A primary program can also communicate with more than one surrogate program.

consolidated data management
The name of the System 80 data management system. Also available as an option on Series 90.

conversation
An exchange of information between two or more communicating programs conducted via CDM program-to-program imperative macroinstructions. Conversations may be simple or complex.

D

data base
A collection of data fundamental to an organization.

data handling
The ability to access data on a remote host using programs on the local host.

DDP
See distributed data processing.

DDP command analyzer
See command analyzer.

DDP file access facility
A SPERRY UNIVAC DDP program facility composed of a;

- remote file processing component – whereby you can access and process files residing on remote OS/3 systems; and
- program-to-program communications component – whereby you can write application programs that can initiate data, jobs, and control information on a remote host or hosts.

DDP system
See distributed data processing system.

DDP transfer facility
A SPERRY UNIVAC DDP program facility, which allows workstations and terminal operators to copy files and libraries from one system in a network to another system in a network, and submit jobs to other systems in the network. The files may reside on the system to which the operator is connected (local or home system), or they may reside on some other system in the network (remote system).

DEACT
Parameter of the DMCTL imperative issued by the primary program to provide data, to and solicit a response from, the surrogate program in conjunction with a request to terminate the conversation. The surrogate program may accept ((REPLY (END)) or reject (REPLY) the terminating request when sending the response message.

decentralized data processing (computer) system
A method whereby several independent computers exist within an organization but are not electronically connected.

defined record management
The facility of IMS to access a defined file.

demand mode interface
The ICAM system interface that supports both distributed data processing and all OS/3 interactive services.

destination file
In a DDP system, the file into which another file is being copied.

destination-host-id (with the DDP file access facility)
Is one to four alphanumeric characters naming the host to receive the data. If omitted, DDP assumes the destination file is on your local host. See 7.1.2, 8.3.2 and 8.4.1.

destination (with the DDP file access facility)
Is an extension of the user-id node used in the DDP file access facility, with the format:

destination::=[host–id:]user-id

Host-id is a one to four alphanumeric character identification of the computer on which you wish your job to be executed. The host-id is optional and defaults to the local host, that is the host computer on which the job is executing. The generic host-id, $HOST, specifies that the host-id is the originator or master.

display messages
Information about a job being executed, displayed on a console or terminal screen.

distributed communications architecture (DCA)
The SPERRY UNIVAC architecture that draws together all aspects of the communication products by defining a set of logical concepts and a set of rules (protocols and interfaces) and guidelines to be used in applying the concepts in the design of hardware, software, and network products.

distributed data processing system
A configuration of independent computers joined in a peer relationship.

DMI
See demand mode interface.

E

element
In OS/3, a discrete part of a module in a program library file. In DDP commands, however, an element is a module.

element type
The kind of code that the elements are written in, such as symbolic (source code) or absolute (load code). See also element.

encoding
The converting of data through use of a code so that it can be reconverted to its original form.

F

file-id
The complete name by which a file is identified. In DDP, the file-id must include the file name, and may also include the module name, read and write passwords, volume, and element type, in this format:

```
'[module-name],filename[([read-passwrd]/write-passwrd])],[vol],[element]'
```

file name
From 1 to 44 alphanumeric characters identifying your file when it resides on disk, and 1 to 17 alphanumeric characters for tape files and logical files (spool files).

H

help screens
A display of how information and additional help can be obtained for the DDP commands.

heterogeneous DDP system
A DDP system in which the various hosts use different operating systems.

hierarchical relationship
An arrangement of hosts so that there is a different rank among them.

homogeneous DDP system
A DDP system in which the various hosts all use the same operating system.

host
An independent computer attached to a DDP system.

host-id
The identifying name of your host. In DDP, it is the same as the node-id you specify in the REMOTE parameter of your LOCAP macroinstruction in your ICAM network definition. If host-id is omitted, the local host is assumed. See 7.1.2, 10.4.1 and the individual commands and imperatives where host-id is used for particular specifications. Also, see the variations: destination, local and originating host-ids.

I

ICAM
Integrated communications access method. A generalized software package for OS/3 communications that gives you multiple levels of interface to remote devices.

IMS DDP transaction processor facility (IMS DDP)
A SPERRY UNIVAC IMS DDP transaction processor facility, which enables workstations and terminal operators to process IMS transaction at a remote OS/3 computer. Multithreaded IMS systems are required at both the local and remote sites. IMS can route a transaction to a remote system by:

- transaction directory routing;
- transaction program routing; or
- transaction operator routing

indexed file
A file accessed according to one or more index keys (for example, IRAM or MIRAM files).

information management system (IMS)
A software product facilitating the development and installation of online, transaction-oriented data base management applications under OS/3.

interactive services
A collection of commands and system programs that enables you to prepare, run, and control jobs and to perform system housekeeping routines easily and quickly from a workstation.

J

jobname
Is one to eight alphanumeric characters naming the job that the DDP SUBMIT FILE command returned to you. DDP assumes the job is one the local host. See 8.2.6.3.

jproc
A series of job control statements stored in a library that can be executed by a simple job control statement. (Same as "proc" or "procedure".)

L

library file
A file consisting of a collection of program modules.

local host
A one to four alphanumeric identifying the computer you are using at your site. see the command using the local host-id for particular specifications at 8.3.1, 8.3.3 and 8.5.1.

LOCAP (LOCal APplication) file
The file that is necessary to permit program-to-program transfer, whether local or remote, and supplies the name and type of program that other applications may address for access.

M

macro or macroinstruction
A source statement in an assembler program that is converted into many assembler source statements to do a particular function (for example, OPEN or CLOSE).

master-slave relationship
A communications connection in which one computer completely controls the data sending and receiving functions.

message
Consists of some arbitrary amount of information whose beginning and end are defined or implied, and is transmitted from one program to another program in a DDP network.

module
A part of a library file that is accessible by name as a unit.

N

network
The total collection of physical entities joined in a data communications system (nodes, network processors, terminals, etc).

nine-thousand remote (NTR)
A software utility program that controls data transmission to allow the SPERRY UNIVAC 90/30 System to be used as a terminal to the SPERRY UNIVAC 1100 System.

nonencoded characters
Standard codes (such as EBCDIC or ASCII) are made up of bit patterns. There are more possible bit patterns than are actually used in these codes. Some users alter their computers to recognize patterns that are not part of the standard code. These unique bit patterns are called nonencoded characters.

NTR
See nine-thousand remote.

O

originating file
In DDP, the file from which a copy is being made.

originating host-id
Is one to four alphanumeric characters naming the originating host. Used with the DDP SUBMIT FILE command, it names the job control stream file location. If HOST-ID is omitted, the system assumes the file is on your local host. See 8.3.2 and 8.4.1.

P

password
One to six alphanumeric characters you must specify to read a file (read-passwrd) or write to a file (write-passwrd). Passwords are opitonal parts of the file-id. They are used only with cataloged files.

primary host
The originating host in a DDP system.

primary program
The initiating part of a pair of application programs in the program-to-program component of the DDP file access facility. See surrogate program for the other part of the communicating pair.

procedure/proc
See jproc.

program-to-program communication
A component of the DDP file access facility whereby you can write a BAL program that can initiate and carry on a "conversation" with another BAL surrogate program at a remote OS/3 host. The communicating programs must reside on different hosts in the DDP network.

protocol
A set of rules defining the structure, content, sequencing procedures, and error detection and recovery techniques for the transmission of data. Also used to establish, maintain, and control communications between two corresponding levels in a level hierarchy. Normally implies the sending and receiving of unique command and response messages or message headers.

R

read password
See password.

relative file
A file you may access either record-by-record or by the relative number of the logical record within the file.

relocatable code
Object code.

remote file processing
A component of the DDP file access facility whereby you can access and process disk files residing on remote OS/3 systems in your DDP network.

remote host
The computer geographically separated from you to which you are electronically connected.

ring structure
The joining of hosts in a circle so that each is connected to two others.

S

screen format
A form displayed on a video terminal used to input data to a program or to output data from a program.

secondary host
The destination host in a DDP system.

sequential file
A file you access record-by-record, according to the order of the records in the file.

simple conversation
A 1-way-only communications discipline whereby the primary program is only allowed to send and the surrogate program is only allowed to receive. Only one primary program and one surrogate program is involved and that relationship remains for the duration of the conversation.

spooling
The process by which the central processor reads and writes records to or from a high-speed storage device rather than a slower device.

stand-alone processing
The ability of a computer to process data with no connection to any other computer.

star structure
The joining of hosts so all hosts are connected to one central host.

surrogate program
A surrogate user application program that is one part of a pair of programs used with the DDP file access facility. See primary program for the other part of the program pair. However, the two programs can exchange statuses via the BEQueath parameter of the DMCTL imperative.

symbolic code
Source code.

system
See distributed data processing system.

T

Telcon
A SPERRY UNIVAC communications system designed to implement network communications in a wide variety of applications. The Telcon system is based on a unified family of hardware and software components that implement the distributed communications architecture. The principal hardware component is the distributed communications processor.

terminal
A point in a network where data can either enter or leave. Normally operator oriented in that it provides for human interpretable input/output media.

transaction directory routing
A routing method of the IMS DDP transaction processor facility whereby the terminal operator routes a transaction via a transaction code that identifies a transaction at a particular remote system. See transaction operator writing and transaction program routing.

transaction operator routing
A routing method of the IMS DDP processor transaction facility whereby the terminal operator routes a transaction via a transaction code with a special character that routes the transaction to a particular remote system. See transaction directory routing and transaction program routing.

transaction processing
The use of an information management system to request information or to change records, and to receive a response. Each transaction involves one input request from a terminal followed by one output response from a host. DDP transaction processing involves the use of information management systems across two or more hosts.

transaction program routing
A routing method of the IMS DDP transaction processor facility whereby the terminal operator initiates a transaction at the local IMS system. The COBOL or basic assembly language action program sends a message that initiates a transaction at the remote system. See transaction directory routing and transaction operator routing.

tree structure
The joining of hosts so that each is connected to at least one other host but may in addition be connected to others.

two-way alternating communication
A DDP file access facility communications discipline between pairs that provides a request/response exchange to coordinate data transfer. A sense of primary and surrogate program is maintained. See complex conversation.

U

undefined file
Any type of file except a library file containing source (symbolic) code.

UNIQUE (uniform inquiry update element)
An easy-to-use inquiry language that lets you display data and update your files by entering commands from the terminal. A set of IMS-supplied action programs processes these UNIQUE commands.

user-id
Is the one to six alphanumeric characters identifying you as a user to the host.

V

volume table of contents (VTOC)
A directory written on your disk volume that lists the addresses and other information about the files on that volume.

W

work order number
A reference number assigned to each command entered into the DDP software.

work order request
Any command you enter into the DDP software. Some messages use this term when they're referring to the command you entered immediately preceding this message.

workstation
Any terminal, consisting of a CRT display and typewriter-like keyboard, that you use to access the interactive services.

write password
See password.

Index

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

_____

(Document Title)

_____ _____ _____

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

Cut along line.

From:

_____

(Name of User)

_____

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

CUT

FOLD