IBM

DPPX/SP

General Information

8100 Information System

IBM DPPX/SP

General Information

Program Product

GC23-0600-2

Second Edition (October 1983)

This edition applies to the following IBM program products:

- DPPX/SP (5660-281)
- DPPX COBOL Compiler (5760-CB1)
- DPPX PL/I Compiler (5760-PL1)
- DPPX PL/I Library (5760-LM2)
- DPPX FORTRAN Compiler (5760-FO1)
- DPPX FORTRAN Library (5760-LM1)
- DPPX Assembler (5760-AS1)
- DPPX APL (5760-XR2)
- Cross System Product/Application Development for DPPX/SP (5660-284)
- Cross System Product/Application Execution for DPPX/SP (5660-285)
- DPPX/SP Interactive Map Definition (5660-282)
- Data Capture and Management System for DPPX (5760-XR6)
- DPPX Presentation Services for 3640 Terminals (5660-267)
- DPPX Parameter Table Generation Facility for IBM 3644 Automatic Data Unit (5760-ED1)
- DPPX Performance Tool (5760-XR5)
- Host Command Facility Version 1 (5735-XR1) and Version 2 (5668-985)
- Distributed Systems Executive Version 2 (5668-986)

The program products described in this manual, and all licensed material available for them, are provided by IBM under terms of the Agreement for IBM Licensed Programs. Your branch office can advise you on ordering procedures.

Information in this manual is subject to change. Subsequent revisions or technical newsletters will include such changes.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Copies of this and other IBM publications can be obtained through IBM branch offices.

A form for reader's comments has been provided at the back of this publication. Address any additional comments to IBM Corporation, Department 52Q, Neighborhood Road, Kingston, New York 12401. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

About This Book

This book describes the Distributed Processing Programming Executive System Product (DPPX/SP) and related program products that run on the IBM 8100 Information System.[1] Together, DPPX/SP and its associated program products make up a DPPX operating system.

This book's purpose is to help data processing managers and their technical staffs understand the capabilities of DPPX/SP.

This book consists of three parts:

- **Introduction**—a general explanation of DPPX/SP, its capabilities, examples of DPPX/SP in use, and descriptions of DPPX/SP operations and configurations.
- **DPPX/SP Features and Enhancements**—a more detailed description of DPPX/SP than is offered in the **Introduction** along with descriptions of important functional areas.
- **DPPX Program and Host-Related Products**—detailed summaries of the associated DPPX program products that work with DPPX/SP in the 8100 and the host computer.

Part 1 of this book provides a general introduction for the audience with limited knowledge of data processing.

Parts 2 and 3 assume that you understand data processing concepts, know what distributed processing is, and know what the IBM 8100 Information System (the hardware) is.

Some terms that are used regularly in this book and that you should understand are:

- **8100**—An 8130, 8140, or 8150 Processor, optionally an 8101 Storage and Input/Output Unit, and input/output devices.
- **System/370**—Any of the processors in the System/370 family, including the 308x and 303x series, with attached input/output devices.
- **Host computer** or **host**—The primary or controlling computer in a multiple computer operation. In a network with DPPX/SP, the host can be a 308x, 303x, 4300, or a System/370.
- **Subsystem and workstation attachments**—Subsystems that can be connected with DPPX/SP, including Series/1, 6580 Displaywriter, 5150 or 5160 IBM Personal Computer, Distributed Processing Control Executive (DPCX), 4700 Finance Communications System, 5280 Distributed Data System, and the 364x Plant Communications Devices.

Other terms and their meanings are listed in the **Glossary** of this book. You can read about distributed processing and the 8100 in *An Introduction to the IBM 8100 Information System,* GA27-2875.

---

[1] DPPX/SP consists of two releases, DPPX/SP 1 (Release 1) and DPPX/SP 2 (Release 2). Where this book uses DPPX/SP, it refers to both releases. DPPX/SP 2 enhancements are designated by referring specifically to DPPX/SP 2 or Release 2.

Contents

Figures

Part 1. Introduction

This part briefly describes the Distributed Processing Programming Executive System Product Release 1 (DPPX/SP 1) and Release 2 (DPPX/SP 2). Chapter 1 describes what they are and what they can do. Chapter 2 describes what program products work with them. Chapter 3 describes configuration possibilities. Read this part for a general overview of DPPX/SP capabilities and applications.

Part 1 also contains two examples of how people can use DPPX/SP.

"Chapter 4. PartsCo—A New DPPX/SP User" on page 35 shows how a business installs an 8100 system with DPPX/SP for the first time and solves a problem by converting to distributed data processing. (The 8100 system here includes DPPX/SP and other optional program products.) It shows the steps new users take to:

1. Install the DPPX/SP program product at a central site 8100 and connect it to an existing large computer
2. Start the system for the first time
3. Write programs on the system
4. Test the programs
5. Install DPPX/SP at distributed sites and connect it to the central system

Chapter 5 shows how a different kind of business uses DPPX/SP. The Milwaukee General example describes a system where distributed 8100s exchange data with a central host computer continually during the day.

The different DPPX/SP functions and other DPPX program products are mentioned by name as they come into play in these two examples. For full information on individual program products, see "Part 3. Additional DPPX and Host-Related Program Products" on page 155

The companies used in the examples in Chapters 4 and 5 are fictitious.

Note: This part assumes that you are not fully familiar with distributed data processing and the DPPX family of products. If you are acquainted with previous DPPX releases, you should review "DPPX/SP Releases 1 and 2 share advantages over DPPX Base" on page 6 and "DPPX/SP 2 has advantages over Release 1" on page 7 before reading Part 2.

Chapter 1. What Is DPPX/SP?

The Distributed Processing Programming Executive/System Product Releases 1 and 2 (DPPX/SP 1 and DPPX/SP 2) and their associated DPPX program products form an interactive operating system for running application programs on IBM 8100 Information System computers.

You may already be familiar with previous releases of the 8100 Distributed Processing Programming Executive (DPPX) operating system. DPPX/SP Release 1 (DPPX/SP 1) and Release 2 (DPPX/SP 2) combine several of the programs that were marketed separately under the DPPX Base operating system. Both DPPX/SP releases are installed, serviced, and customized as a single unit.

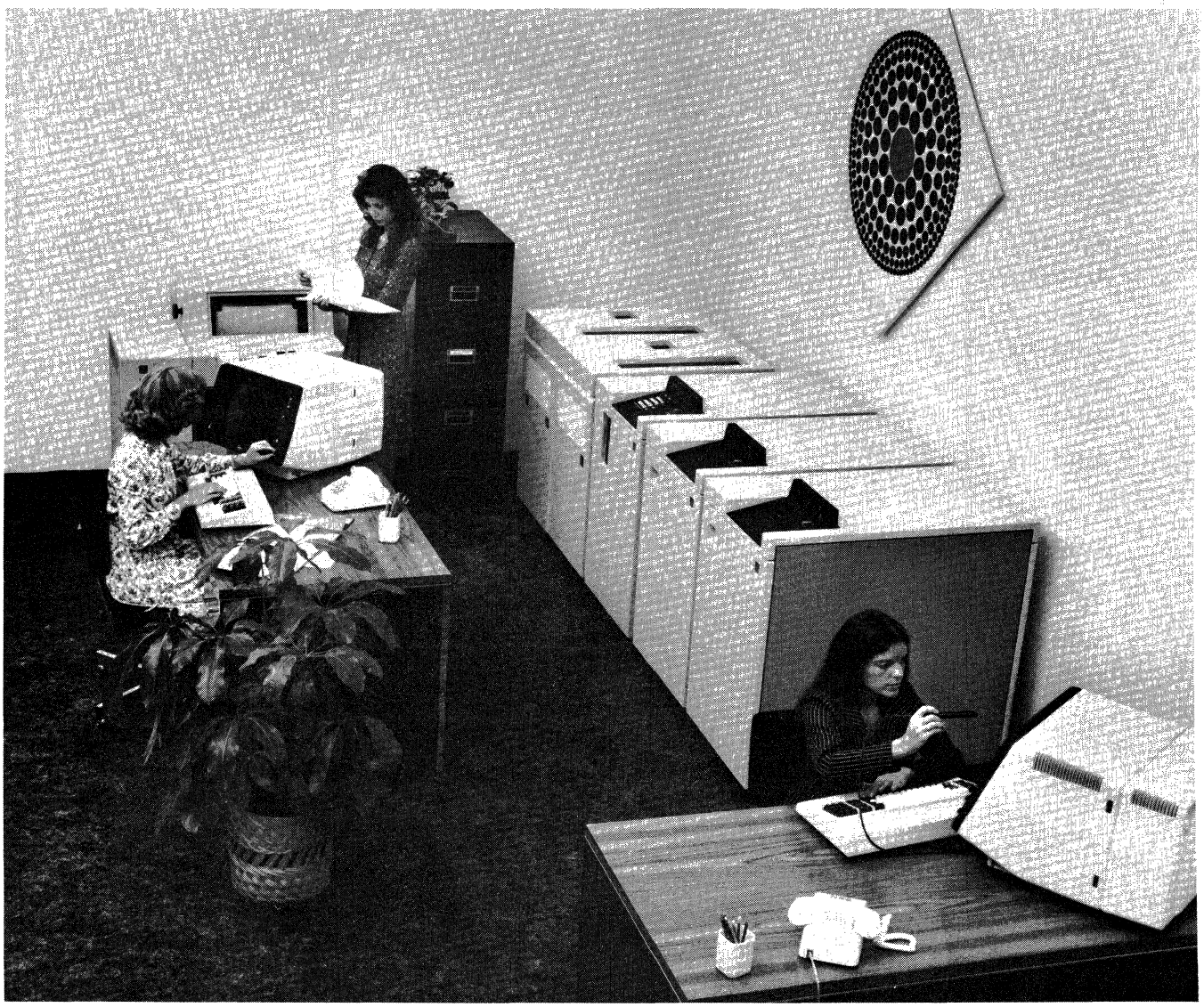
In addition, DPPX/SP 2 includes enhancements and new functions not available with Release 1. These modifications not only allow DPPX/SP 2 to take full advantage of the expanded hardware capabilities of the 8150 processor but will also enhance the productivity of 8130 and 8140 processors.

DPPX/SP and the 8100 are designed to distribute processing among computers in a cooperative network. By means of DPPX/SP, 8100 computers are connected to host computers (308x, 303x, 4300, or System/370), and all perform related tasks. An 8100 system can also be connected with another 8100. If communication with the host is lost, or in the absence of communication links, an 8100 can run as a stand-alone processor.

DPPX/SP (with optional program products) lets you write application programs to perform diverse functions. For example, you could track movement of products through a factory, check credit in a host-managed data base, solve engineering design problems, maintain inventories and price lists, and print shipping labels.

By storing information and programs in several locations, you ensure immediate access to vital data by various departments. Data is more likely to be kept up to date and less likely to be inaccessible because of a central computer's failure or inability to transfer data quickly to a remote site.

If you have application programs that run with DPPX Base, they will also run with DPPX/SP 1 and DPPX/SP 2. You can also develop customized application programs on DPPX/SP using associated program products such as DPPX COBOL and DPPX PL/I.



What can DPPX/SP do for me?

DPPX/SP has many features to help you run a distributed data processing system. Several of those features are described briefly in this chapter.

DPPX/SP combines several DPPX products as well as new functions to form a **single program product**. Because DPPX/SP is one product, it is easier to install, use, and service than the previous DPPX Base system and its optional products.

Through **distributed processing**, DPPX/SP lets you decrease the workload placed on a central computer. It also lets you increase productivity at remote locations by decreasing the time it takes to access information that those sites need. DPPX/SP also lets users manage an entire network from one central location.

DPPX/SP allows **access to data** in both 8100s and host computers. You can avoid needless and costly duplication of data.

DPPX/SP **supports many devices and subsystems**. You can change the configuration of your DPPX/SP system at any time. As new software and hardware products develop, you can expand your system smoothly. IBM's system network architecture (SNA) assures continued communications compatibility.

Users can **develop application programs** for their system by adding any of the programming languages supported by DPPX/SP. Several programming languages and tools are available. You can develop these application programs at both central and distributed sites.

DPPX/SP provides **online panels** to help the user obtain services. The panels prompt the user to enter information at the terminal. Inexperienced users can take advantage of special help panels and tutorials to become familiar with the on-line panels.

DPPX/SP also provides special enhancements which aid programmers in constructing panels of their own design. These **customized panels** will decrease the time spent in performing repetitive or complex tasks. Through online prompts provided by these panels, you minimize costly and time-consuming user error.

DPPX/SP provides **security and data integrity** through several features. An authorized user can control which users have access to what data, and whether they can view, change, or print it. DPPX/SP also protects the integrity of data bases. It allows recovery from errors as well as re-creation of a data base in case of damage.

If you are migrating from DPPX, you will find that DPPX/SP incorporates the functions of DPPX Base with many of the related program products and several new functions. A migration guide is provided to ease your transition to DPPX/SP.

DPPX/SP Releases 1 and 2 share advantages over DPPX Base

DPPX/SP combines several DPPX program products into a unit that can be ordered, installed, and serviced as one product. DPPX/SP by itself automatically supplies functions that you get with DPPX Base only if you ordered several program products.

Both Release 1 and Release 2 include functions not previously offered as part of the DPPX Base and optional program products. These include:

- **Improved response time and improved system performance:** This is accomplished by keeping the frequently used information in a special real storage buffer called a *DASD Cache buffer*. The DASD Cache buffer is continually updated so that it always contains the most recent information. The user specifies the size of the DASD Cache buffer when tuning the system.

- **Improved communications hardware testing:** DPPX/SP lets users test 386x modems used in conjunction with the 8100. Results of the test are displayed at the user's terminal and copied to an error file. Under certain error conditions, DPPX/SP will automatically test modems and report the results. Tests can be initiated by a network operator at the host.

- **Revised logon procedures:** Users do not have to log off and on again between sessions. This lets users switch from one application to another smoothly.

- **Reduced initial program load time:** Users can speed up an initial program load (IPL) of the system by taking advantage of the DPPX/SP *save/restore IPL* service. A user performs a save IPL. Then, when the system must be IPLed again, the user performs a restore IPL. The system is restored to the condition it was in when the save IPL was issued.

- **Increased ease of installation and operation:** DPPX/SP comes with online panels to help the operator install, customize, and run the system. The panels are easier and faster to use than commands. Tutorials and help panels are provided for inexperienced users.

  The many DPPX products combined for DPPX/SP include:

  - DPPX Base
  - Data Base and Transaction Management System (DTMS)
  - Data-Stream Compatibility
  - Remote Job Entry (RJE)
  - Problem Determination Application
  - Programmed Operator Facility
  - The Format Management (FM) component of Distributed Presentation Services (DPS)
  - Sort/Merge
  - Interactive Productivity Facility
  - Distributed Host Command Facility (DHCF)
  - Host Data Transfer (HDT)
  - Host Transaction Facility (HTF)

DPPX/SP 2 has advantages over Release 1

In addition, DPPX/SP 2 includes features not offered in Release 1.

DPPX/SP 2 supports several new 8150 hardware features:

- Key/lock approach to storage: improves address space accessibility by allowing a systems administrator to define "user private" address blocks in addition to the Common Address Space Section (CASS) shared by concurrent users. Under previous DPPX addressing systems, CASS was duplicated in each user's address space. Key/lock addressing allows larger CASS sizes and more user address space.

- Higher availability through dual processors and dual input/output devices; Both processors of the 8150 Model B can be connected to input devices (such as terminals) and output devices (such as printers). Depending upon system configuration, all input/output function can be maintained even with one of the two processors failing.

DPPX/SP 2 supports these functions in 8130, 8140, and 8150 processors:

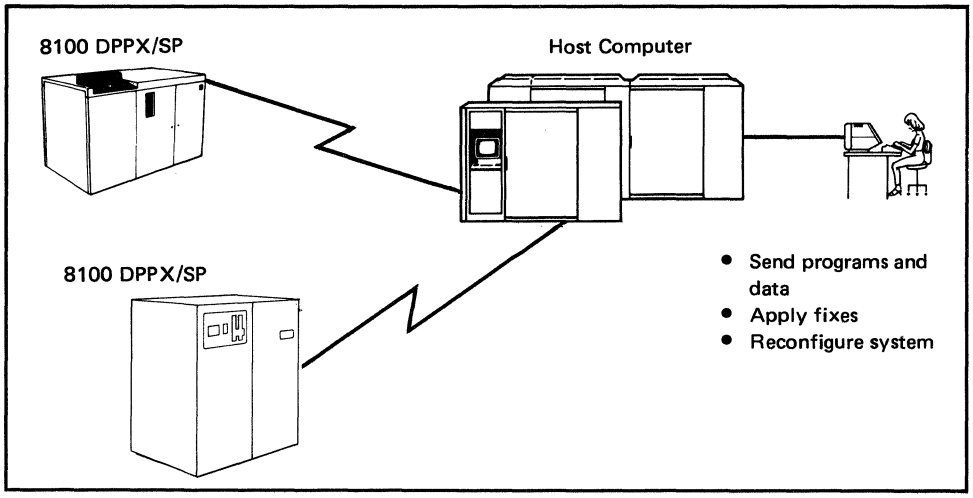
- Improved recovery of data bases: revised data is automatically written to a backup data base. This "shadow" function assures the availability of up-to-date, easily recoverable files if the primary files become inaccessible.

- Enhanced disk migration capabilities: DPPX/SP 2 provides for simplified transition to a disk drive of greater storage capacity. The transition is accomplished without having to transfer data to an external medium and then back to disk. This simplified process guards against data loss and system "down time."

- Multiple batch environments: DPPX/SP2 provides up to 16 environments in batch mode. This support, combined with the faster 8150 processor, allows faster application program excution.

- Improved management of files: DPPX/SP 2 automatically reformats "indexed" files so that they demand less storage space. This automatic reorganization represents a significant time savings over previous methods, which required deactivation of files for reorganization. You can still access, use, and update data while it is being reorganized.

DPPX/SP lets you centrally control and manage a network



DPPX/SP allows you to distribute data and processing requirements throughout a network while a site operator maintains central control. Centering technical skills at the host site minimizes the amount of data processing skills required at the distributed sites. You can decide the amount of central control you want with your 8100 system.

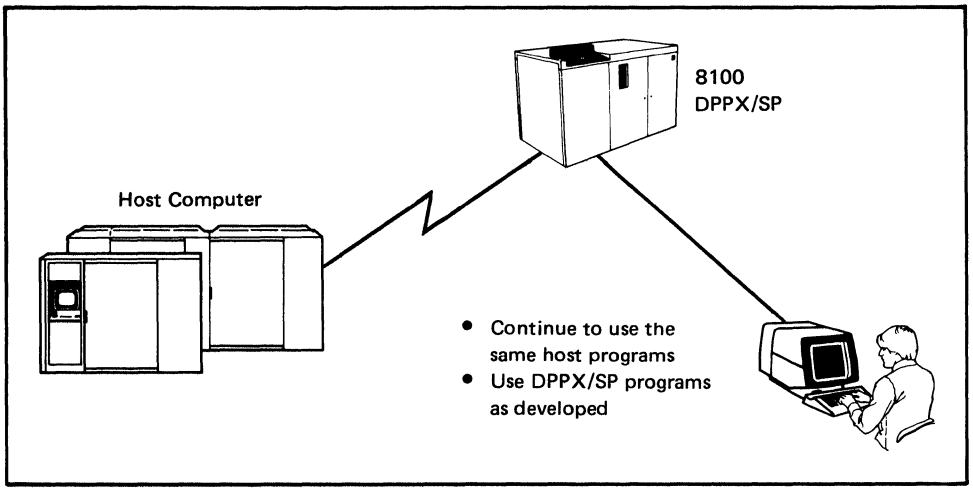
DPPX/SP lets you attach a terminal to a host computer and use it as though it were directly connected to an 8100.

A person at this terminal can issue DPPX/SP commands or use the DPPX/SP online panels to:

- Start and stop work in distributed 8100s
- Configure a distributed 8100 system
- Develop and test applications and hardware
- Access error logs
- Run network and device diagnostics
- Apply program fixes

A person at the central site can also track problems, coordinate changes, and maintain libraries of programs and data to be distributed throughout a network of 8100s.

DPPX/SP lets you ease into distributed data processing



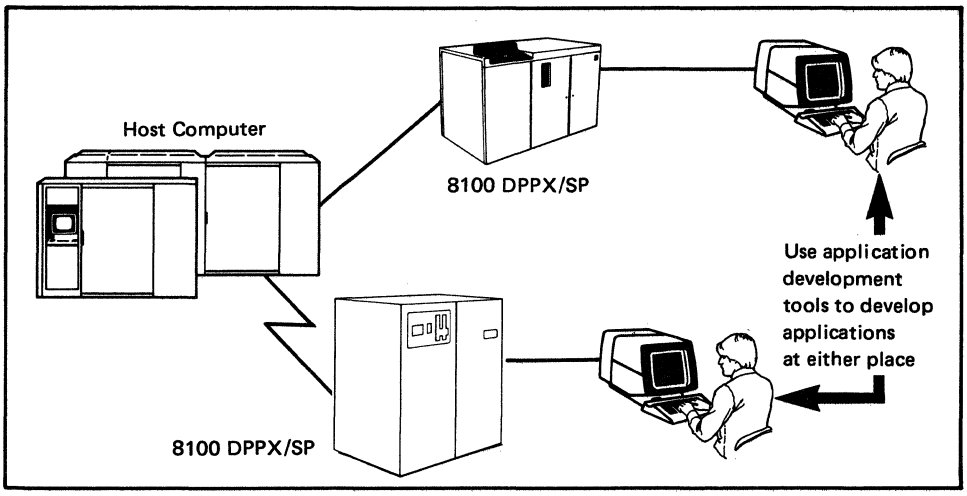
DPPX/SP can ease your transition to distributed data processing by letting you continue to use existing host application programs while you gain experience with new DPPX/SP-based applications. When used in this way, the 8100 functions more like a communication controller than a distributed processor. The 8100 controls the flow of data but does not modify it.

As you develop application programs with DPPX/SP and its related optional program products, you can begin to use the 8100 as a distributed processor. You can run your programs at the same time you are using the existing host application programs.

If you have retained application programs that run with DPPX Base, they will run with the DPPX/SP program product as well.

By using the 8100's terminals and data links in this way, you gain experience with DPPX/SP and the 8100 while keeping existing host application programs in production.

DPPX/SP lets you develop, transmit, and store information where it is used



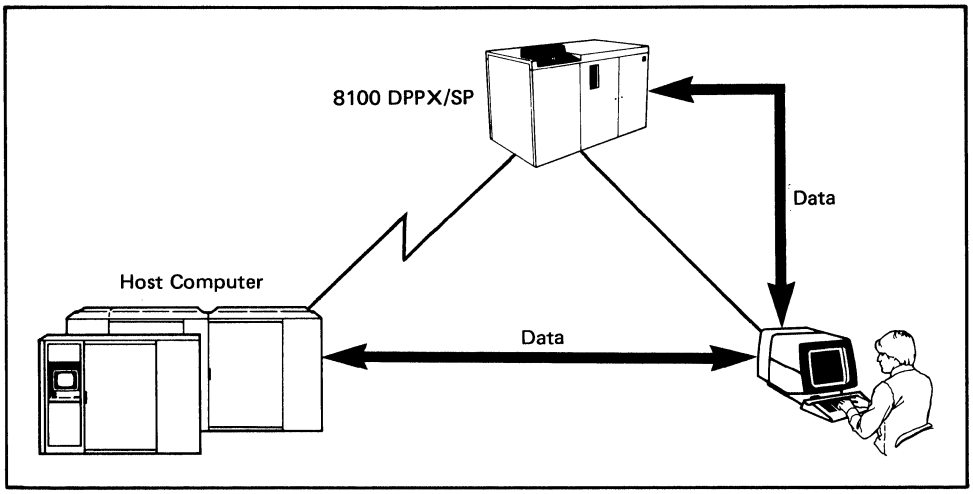
DPPX/SP allows application programs to be developed at a central site, at distributed sites, or at both. You can use the skilled people you already have in your central data processing department to develop DPPX/SP application programs. Once the programs are tested, they can be sent to distributed 8100 locations by data link, diskette, or tape. This keeps down both implementation and operation costs. You also avoid the expense of separate distributed sites developing the same programs.

Using DPPX/SP, you can put the programs and data used by one branch office, department, plant, or other remote location on an 8100 at that location. Programs and data that you want to access from several remote nodes can be conveniently stored in a central host computer, such as a System/370. The remote locations can be connected to the central location by data links. Then the 8100s and host computer can process data independently of each other or share data and programs.

Compared with centralized processing, distributed processing:

- Reduces the load on the central computer.
- Reduces the volume of communication between the central and the remote locations.
- Decreases communication costs.
- Puts computing power in the remote locations, letting them function independently of one another.
- Provides protection against system failures. If one part of the system is down, other parts can still function.

DPPX/SP lets your programs access 8100 and host data bases

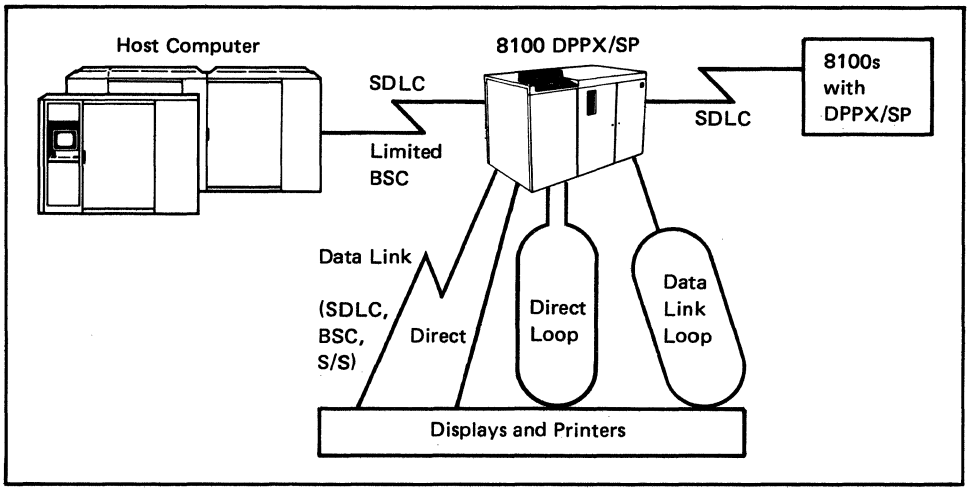


DPPX/SP manages data bases in the 8100 and handles the transfer of data bases to and from a host computer. This allows you have DPPX/SP transaction programs to run in the 8100 and to access a wide range of data bases in various systems. You can decide how much processing the 8100s do on their own and how often they access the host. For example, you might set up your 8100 system so that the programs rely primarily on the local data base and access the host only in predefined situations.

DPPX/SP can handle these types of data base applications:

- Online inquiry—a terminal user requests and immediately receives information from an 8100 or host data base.
- Online inquiry with update (order entry)—a terminal user can alter or store information, such as an order entry, in an 8100 or host data base.
- Online data entry—a terminal user types data with minimum interruption from the computer. The data is used to create batch files at the 8100 location for subsequent processing or access by others.
- Remote batch processing—a remote location sends data to the host computer for batch processing, such as periodic updating of central files. Output can be sent back to the remote location.

DPPX/SP allows many different configurations



The three basic hardware configurations supported by DPPX/SP are:

- Host-connected 8100s, where a processor is connected to a 308x, 303x, 4300, or System/370 host.
- Interconnected 8100s, where a processor can communicate with another processor using a data link.
- Stand-alone 8100s, where a processor does not communicate with another processor.

Application programs running with DPPX/SP can communicate with:

- A host computer
- An adjacent (peer or downstream) 8100 system with DPPX/SP
- Many types of displays, printers, systems, and downstream controllers. For a complete list, see "DPPX/SP-Supported Devices by Device Type" on page 104 in Chapter 8.
- Subsystems and workstations, including:
  - Series/1
  - 6580 Displaywriter
  - 5150 or 5160 Personal Computer
  - An 8100 Information System operating under the control of the Distributed Processing Control Executive (DPCX)
  - 4700 Finance Communications System
  - 5280 Distributed Data System

Devices and subsystems can be attached by loop, directly, or by data link. (Not all devices and subsystems can be attached in all ways.)

DPPX/SP supports *systems network architecture (SNA)*. SNA is IBM's communication architecture for interconnecting IBM products into networks. This assures you that DPPX/SP application programs can continue to use existing and new software and hardware products. They can continue to share a common corporate data communication network.

DPPX/SP has many application development program products

Language/Tool	Program Product
COBOL	DPPX COBOL Compiler (5760-CB1)
FORTRAN	DPPX FORTRAN Compiler (5760-FO1) DPPX FORTRAN Library (5760-LM1)
PL/I	DPPX PL/I Compiler (5760-PL1) DPPX PL/I Library (5760-LM2)
Assembler	DPPX Assembler (5760-AS1)
APL	DPPX APL (5760-XR2)
Cross System Product set for DPPX/SP	Cross System Product/Application Development for DPPX/SP (5660-284) Cross System Product/Application Execution for DPPX/SP (5660-285)
Full-screen data formatting	DPPX/SP IMD (5660-282)

DPPX/SP supports several program products for developing application programs. Among their services are:

- High-level languages
- Menu-driven tools
- Full-screen data formatting
- Common interface to devices

DPPX/SP supports the following products to help develop application programs:

- COBOL
- PL/I
- FORTRAN
- Assembler language
- APL
- Cross System Product set for DPPX/SP
- Interactive Map Definition (DPPX/SP IMD)

COBOL and PL/I provide high-level facilities for developing data base and transaction programs. FORTRAN is available for general problem solving, and Assembler language is available for system programming. PL/I can be used for general problem solving and systems programming applications. APL is an interactive language for application prototyping and mathematical statistics.

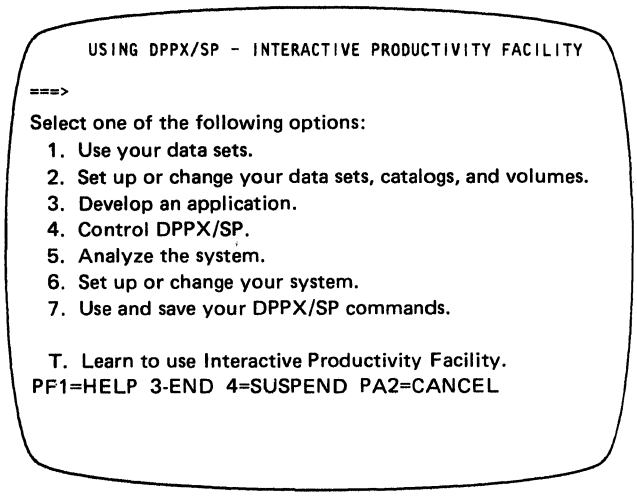
You do not have to use a programming language to develop application programs. An interactive program product, Cross System Product/Application Development for DPPX/SP, guides users through program development by requesting responses to prompts. Both new and experienced users can use this program product to develop application programs quickly with a minimal learning period. (To run application programs developed with Cross System Product/Application Development for DPPX/SP, you also need Cross System Product/Application Execution for DPPX/SP).

Another interactive program product, DPPX/SP Interactive Map Definition (DPPX/SP IMD), lets programmers specify how application programs format data on display terminals and printers.

DPPX/SP has a common interface to most devices so that programmers do not have to be concerned with physical device characteristics and communication protocols. This lets programmers concentrate on the applications themselves rather than the devices the applications will use.

These program products are discussed in Chapter 2 and are detailed in Part 3.

DPPX/SP has easy-to-use panels to invoke most of its services



DPPX/SP provides online panels through the *Interactive Productivity Facility (IPF).* You can use these panels to request most DPPX/SP services. They help users install, customize, operate, and diagnose problems on their 8100 distributed data processing system.

Although panel dialogs do not replace any DPPX/SP commands, they let users issue these commands more easily. Users do not have to remember all the details of the commands, and mistakes are less likely.

Panels are provided to help you do the following tasks:

- Set up or change your system
- Set up, use, and change data sets, catalogs, and volumes
- Develop application programs
- Control DPPX/SP
- Analyze the system
- Use DPPX/SP commands

DPPX/SP also provides help panels and tutorials to help you learn how to use the panels.

Using panels is advantageous because:

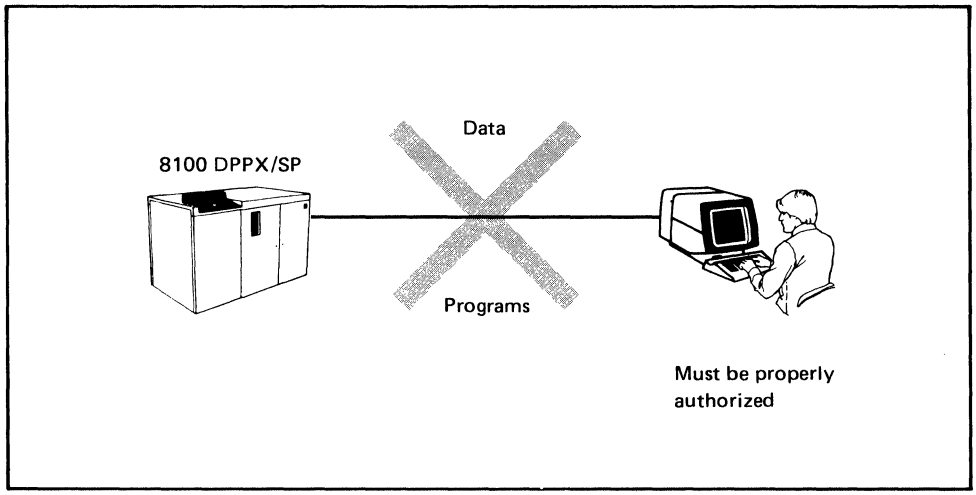
- Panels are organized by task. This means that all the information needed to complete one task is obtained from a series of related panels.

- There is less chance of error. The panels state explicitly what options are allowed.

- There is less to remember. You do not have to remember the syntax of most DPPX/SP commands. You need only supply the information required in a panel to request a service.

- Panels include many defaults. By using the defaults that DPPX/SP provides, you type less information.
- Panel responses are saved from session to session. This eliminates the need to type in the same information over and over again.
- Panels provide easy access to system facilities for all authorized users, whatever their experience level.

DPPX/SP provides data security and integrity

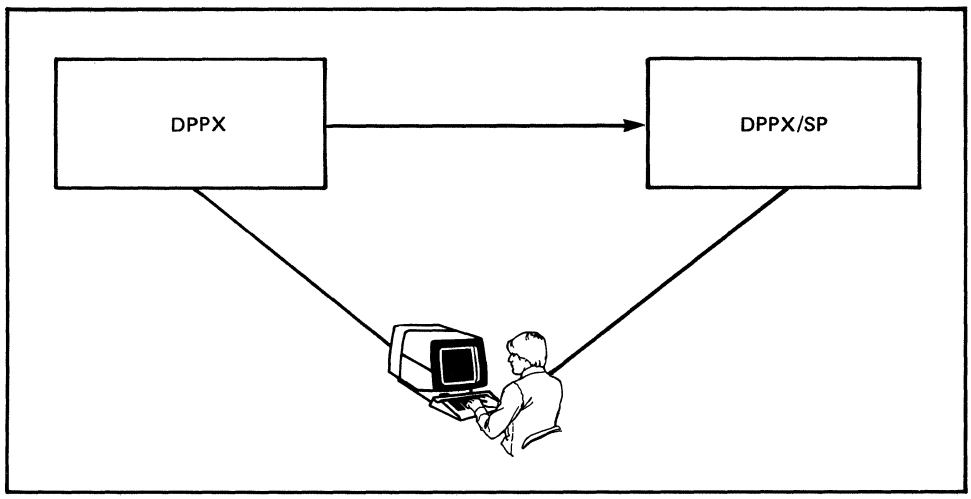


The most important aspect of data security is controlling who may access programs and data. DPPX/SP makes it easy for one person, such as a system administrator, to specify who is allowed to look at, print, and modify programs and data.

There are other data security and integrity features:

- DPPX/SP lets an administrator control the user's access of individual system printers and printer queues. The system administrator places these resources in groups and decides which users can access which groups.
- One failing program in an 8100 does not affect other programs running in the 8100. And one failing 8100 in a network does not affect other 8100s in the network.
- A data base can be automatically reset in the event of system or program failure. DPPX/SP lets a user re-create a data base if the data base becomes unusable. With the "shadow file" enhancement to DPPX/SP 2, a duplicate data base is automatically created with each save.
- The more that data is handled and dispersed, the more likely it is to get out of date. But distributed data processing keeps data close to where it is entered into the system. Data is handled less frequently, is nearer those responsible for it, and is therefore more likely to be accurate and timely.

DPPX/SP provides a migration aid to help you migrate from DPPX Base



DPPX/SP provides a special migration feature to help DPPX users migrate from their current 8100 DPPX Base system to DPPX/SP. The migration aid helps you:

- Determine the amount of extra disk space needed for DPPX/SP.
- Obtain additional disk space in system catalogs if necessary.
- Prepare remote DPPX sites to receive DPPX/SP from the central site.

You can migrate from DPPX/SP 1 to SP 2 through a DPPX/SP 2 Revision Package.

DPPX/SP combines many features into one operating system

In conclusion, DPPX/SP forms a comprehensive operating system for the IBM 8100 Information System.

DPPX/SP:

- Manages system use in the 8100 by
  - Letting many programs and terminals use the system at the same time.
  - Supporting interactive and batch applications.
  - Letting programs and terminals communicate with:
    - Subsystems and workstations, including DPCX, IBM 5150 and 5160 personal computers, 6580 Displaywriter, Series/1, and others.
    - Other 8100s
    - A host computer, such as the 308x, 303x, 4300, or System/370.
    - Other programs and terminals
  - Supporting many optional application development program products.
  - Allowing configuration flexibility.
- Manages data bases and processes transactions through:
  - Data base recovery from detected errors caused by program, system, or machine malfunction.
  - Concurrent processing of user-written transaction programs.
  - Transaction programs' ability to reuse system resources serially.
  - System-supplied services to link successive transaction programs.
  - Detection and management of situations that would result in unresolved contention for resources (deadlock).
  - Aids for program testing.
  - Utilities to re-create a damaged data base and to reorganize a data base for more efficient access.
- Requires minimal data processing skill for production users. DPPX/SP provides a panel interface to many services, allowing users to request services by responding to questions instead of entering commands. DPPX/SP first displays a series of menus and then a set of data entry panels, prompting the user to select the desired function. Panels let users perform such tasks as handling files and managing the DPPX/SP system.

- Lets users switch smoothly between applications. DPPX/SP lets terminal users move quickly from one application to another. Users do not have to end one session and start another to use different applications. These applications can be anywhere in the network. They do not have to be in the local 8100.

- Sorts, merges, and copies data. Users can select records based on a comparison (less than, equal to, greater than, and so on) with a constant when invoking these features. DPPX/SP can sort on multiple control fields positioned anywhere in a record and can merge several previously sorted data sets. DPPX/SP does not change records. It only rearranges their order. The output record is always identical to the input record.

- Lets 8100-attached devices communicate with the host. DPPX/SP lets certain devices attached to an 8100 system communicate with application programs in a host computer as though they were directly attached by data link to the host. These devices can be keyboard displays, printers, attached subsystems and workstations, controllers, or processors. DPPX/SP host attachment services aid in transition to distributed data processing. 8100 systems can be used to access existing host applications while new distributed applications are developed and tested.

- Lets jobs be sent to the host. DPPX/SP lets users at an 8100 submit user-defined jobs to a host operating system's job queue to await execution. (A host Remote Job Entry subsystem, such as JES2, handles the jobs at the host.)

  DPPX/SP also controls job output sent back to the 8100 location after execution. The host operating system's job management routine runs each job and returns the output to DPPX/SP. Another advantage of DPPX/SP is that its Remote Job Entry component can run unattended.

- Allows centralized network management. DPPX/SP monitors activity in the 8100 processor and alerts the system operator of a Synchronous Data Link Control (SDLC) connected host if certain errors occur. It can alert a host operator of:

  - Unrecoverable device and adapter errors

  - Certain failures of an IBM program product or a user-written application program

  - An ALERT command issued by an application program or terminal user

- Performs many system operator functions. DPPX/SP requires less skill and effort to operate a remote 8100. It can intercept messages and return codes targeted for the 8100 system operator and apply predefined actions or responses to them. The responses can be user-specified or IBM-specified. (IBM supplies default responses that users can alter.)

- Provides a key/lock address structure that improves storage space accessibility in the 8150. This offers system administrators greater control over user access and eliminates redundant common address space sections. (Release 2 only.)

- Runs COBOL application programs. COBOL is a high-level programming language suited to developing business application programs. It is especially efficient for coding programs that manipulate files of formatted data. DPPX/SP contains subroutines that provide services at execution time to run a DPPX COBOL program. An optional program product, the **DPPX COBOL Compiler**, is also available for developing COBOL applications.

- Reformats indexed files without data base deactivation, automatically increasing available storage. (Release 2 only.)

- Provides "Shadow file" support. The Shadow file feature allows the user to maintain an exact duplicate of data bases on a second, backup disk. The Shadow feature aids in immediate file recovery from primary disk failure. (Release 2 only.)

- Presents data according to user-developed maps. DPPX/SP helps users format data on input/output devices. The **DPPX/SP Interactive Map Definition (DPPX/SP IMD)** program product is needed for users to develop maps specifying how data is to be formatted. DPPX/SP uses these maps to format application program data.

- Multiple batch environments. Up to 16 batch mode environments are supported (Release 2 only.)

Chapter 2. Introducing the DPPX Program Products

DPPX/SP and optional program products provide the functions described in Chapter 1. Figure 1 summarizes them, and the following pages describe them briefly. See "Part 3. Additional DPPX and Host-Related Program Products" on page 155 for detailed descriptions of these products.

Many of the program products that work with DPPX Base work with DPPX/SP also.

Program Product	Description
System Management	
DPPX/SP	• Manages system use • Manages data bases and transactions • Makes access to most DPPX/SP system services easier by providing a full-screen panel interface • Lets users switch from one application to another anywhere in the network without logging off and on • Sorts, merges, and copies data • Lets users use host applications from 8100 terminals and downstream controllers and processors • Lets users submit host batch jobs from 8100s and print batch jobs at the 8100 • Alerts a host operator of 8100 errors • Reduces the activities required of a human operator • Allows execution of COBOL applications • Formats data according to the maps created by the DPPX/SP IMD program product • Multiple batch mode environments (Release 2 only) • Improves address space accessibility in the 8150 by using key/lock addressing (Release 2 only) • Allows dual I/O buses for 8150 (Release 2 only) • Reorganizes and reformats indexed data sets without temporary loss of system capabilities (Release 2 only)

Figure 1 (Part 1 of 3). DPPX/SP and Associated Program Products

Program Product	Description
Application Programming Languages and Tools	
Cross System Product Application Development and Application Execution for DPPX/SP	Cross System Product—for developing programs interactively and executing them
DPPX APL	APL—for writing programs to do application prototyping and mathematical statistics
DPPX Assembler	Assembler—for writing system programs
DPPX COBOL Compiler	COBOL—for writing programs that access data base
DPPX FORTRAN Compiler, DPPX FORTRAN Library	FORTRAN—for writing general-purpose problem-solving programs
DPPX PL/I Compiler, DPPX PL/I	PL/I—for writing programs Library that access data bases, general purpose and system programs
Application Support	
Data Capture and Management System (DCMS) for DPPX	Handles online bulk data entry
DPPX/SP Interactive Map Definition (DPPX/SP IMD)	Creates maps during program development
DPPX Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit (GEN 3644)	Helps customize 3644 terminal operation
DPPX Performance Tool	Supplies information useful in tuning the system
DPPX Presentation Services for 3640 Terminals (PS3640)	Helps format output data and edit input data for most 3640 terminals
Network Management (Program Products) (These program products run in the host computer)	
Distributed Systems Executive (DSX)	Lets a user distribute programs, maps, and data to 8100s from a host
Host Command Facility (HCF)	Lets a user operate, debug, and manage 8100s from a host
Network Problem Determination Application (NPDA)	Handles problem-related data retrieved from network components*

Figure 1 (Part 2 of 3). DPPX/SP and Associated Program Products

Program Product	Description
Network Management (DPPX/SP Components) (These are part of DPPX/SP)	
Distributed Host Command Facility (DHCF)	Helps create a link between the host and the 8100
Host Data Transfer (HDT)	Allows data transfers between the host and an 8100
Host Transaction Facility (HTF)	Lets 8100 transaction programs communicate with the host
Problem Determination Application	Alerts an operator at the host of 8100 hardware and software errors

*NPDA is listed in this chart and is discussed briefly in the rest of this manual where appropriate. It does not have a separate chapter in Part 3, however, because it is not an 8100 program product.

Figure 1 (Part 3 of 3). DPPX/SP and Associated Program Products

Application Programming Languages and Tools

Cross System Product Set for DPPX/SP

The Cross System Product Set for DPPX/SP consists of two program products[1]:

- **Cross System Product/Application Development for DPPX/SP**, which lets you develop application programs interactively through a "fill in the blanks" approach.
- **Cross System Product/Application Execution for DPPX/SP**, which lets you run applications developed with Application Development or non–Cross System Product programs (such as COBOL) that are called from Cross System Product–defined programs.

Cross System Product/Application Development for DPPX/SP guides users online in writing application programs by requesting responses to prompts at display terminals.

Cross System Product/Application Execution for DPPX/SP executes Cross System Product–generated application programs.

Using Cross System Product/Application Development for DPPX/SP is a convenient and simple way to interactively define data structures, maps (screen formats), and processing statements; test these definitions; and run them in production. Cross System Product/Application Development for DPPX/SP has some important advantages over a programming language:

- It requires less programming experience and skill.

[1] The Cross System Product Set for DPPX/SP is the successor to the Development Management System (DMS) program product.

- It takes the initiative by prompting the user for information that it can turn into a program. High-level language programming (in COBOL, for instance), requires the programmer to determine what statements to code and in what order, a time-consuming and error-prone process.

- It does not require the user to know technical system interfaces.

See "Chapter 15. Cross System Product Set for DPPX/SP" on page 161 for more detailed information on these two program products.

DPPX COBOL Compiler

COBOL is a high-level programming language suited to developing business application programs. It is especially efficient for coding programs that manipulate files of formatted data.[2]

The DPPX COBOL Compiler program product translates source statements into machine language. This is an optional program product that runs with DPPX/SP and aids in application development.

The major features of the DPPX COBOL language are:

- Access to many of the input/output services of DPPX/SP through COBOL input/output language.
- Access to all DPPX/SP data bases through COBOL input/output language.
- Access to DPPX/SP. for formatting data on input/output devices. The DPPX/SP IMD program product eases definition of these formats.
- Aids for structured programming.
- An option to generate reentrant code, permitting many users to share a single application program.
- Sharing of the subroutines of a single DPPX COBOL subroutine library by multiple application programs running concurrently.

DPPX FORTRAN

FORTRAN is a mathematically oriented, high-level programming language best suited to manipulating numerical data. It can help solve research and analytical problems in science, engineering, and business and is simple enough to be used by the nonprofessional programmer.[3]

DPPX FORTRAN consists of two program products:

- The DPPX FORTRAN *Compiler*, which translates source statements into machine language
- The DPPX FORTRAN *Library*, which contains intrinsic functions, service subroutines, and I/O routines to support program execution

DPPX FORTRAN is useful for coding general problem-solving programs.

---

[2] DPPX COBOL is a level-1 implementation of 1974 American National Standard COBOL (ANSI X3.23-1974). It also has most of the features in the December 1975 Federal Information Processing Standard (FIPS), PUB 21-1, low-intermediate level.

[3] DPPX FORTRAN is a complete implementation of American National Standard Basic FORTRAN (ANSI X3.10-1966), has most of the features of American National Standard FORTRAN (ANSI X3.9-1966), and has many features not contained in either standard.

DPPX APL

APL is a programming language for interactive problem solving. Users can express problems in APL in a very compact format.

People at different skill levels can use DPPX APL. Programmers can use it to develop DPPX/SP applications. People who are not data processing professionals can use it to write programs for their own use.

The DPPX APL language is compatible with VS APL (Release 4.0), with a few exceptions. DPPX APL has extensions for system interfaces, such as issuing DPPX/SP commands and accessing DPPX/SP services.

Users who are familiar with VS APL will learn DPPX APL quickly.

DPPX Assembler

The DPPX Assembler program product is a language processor that translates source code in Assembler language into machine language that is executable by an 8100 Information System processor.

Assembler language consists of mnemonics that substitute for machine language. Assembler language requires more training for use than high-level languages. It allows close control of program data and execution, and thus it lends itself to such tasks as coding system programs.

The DPPX Assembler includes a macro processor that expands source code macro instructions into functions performed by predefined instruction sequences known as macro definitions. Macro definitions for frequently used system services are part of DPPX/SP. These definitions perform supervisory and common programming functions, such as writing to data sets. DPPX Assembler users can use the IBM-defined DPPX/SP macro definitions. They can also define and use their own macros.

DPPX PL/I

PL/I is a general-purpose, high-level programming language. It is designed for business, scientific, and system programming applications that involve the manipulation of numerical and nonnumerical data. PL/I is the most versatile of the major high-level programming languages.[4]

DPPX PL/I consists of two program products:

- The DPPX PL/I *Compiler*, which translates source statements into machine language.
- The DPPX PL/I *Library*, which contains subroutines that provide services at execution time. The PL/I Library must be installed on a processor that is to execute a DPPX PL/I program.

[4] DPPX PL/I is a subset of American National Standard PL/I (ANSI X3.53-1976). It contains the more important features of ANSI PL/I yet is easier to learn and understand.

The major features of DPPX PL/I are basically the same as those of DPPX COBOL:

- Access to many of the input/output services of DPPX/SP through PL/I input/output language
- Access to DPPX/SP commands through PL/I built-in subroutines
- Access to all DPPX/SP data bases through PL/I input/output language
- Access to DPPX/SP for formatting data on input/output devices
- Reentrant code generation, permitting many users to share a single application program
- Sharing of the subroutines of a single DPPX PL/I subroutine library by multiple application programs running concurrently
- Extensive error checking at execution time
- Interactive Execution Time Debug facility to monitor program execution, display, and change data

Application Support

DPPX/SP Interactive Map Definition

DPPX/SP Interactive Map Definition (DPPX/SP IMD) is a program product that runs *only* with DPPX/SP, not with DPPX Base. It helps programmers specify how an application program should format data on display terminals and printers.

DPPX/SP IMD lets users define and verify data layouts interactively on a keyboard display terminal. It also allows easy access to device capabilities, such as highlighting, multiple partitions, and field validation from DPPX COBOL, DPPX PL/I, and DPPX Assembler languages, and from programs developed with Cross System Product/Application Development for DPPX/SP.

Data Capture and Management System for DPPX

Data Capture and Management System (DCMS) for DPPX allows full-screen data entry. Data entered at display terminals is checked for proper format, and field validation edits are performed to ensure that the data conforms to the user's specifications. Resulting data sets can be processed by user-written application programs.

DCMS for DPPX can be used in any application or industry that requires the input of large amounts of repetitive data in a batch mode. It differs from interactive terminal applications in that selection of next screen or type of operation is not necessary. It is simply input of predetermined data in a predetermined format.

DPPX Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit

DPPX Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit (GEN3644) is a program product that helps programmers efficiently customize the IBM 3644 Automatic Data Unit. Customization consists of defining I/O channel parameters, selecting 3644 functions, and specifying initial values of stored data items. It results in a parameter table that works with the 3644 microcode provided by IBM.

Programmers customize the 3644 by filling in the blanks on preprinted worksheets, entering the data from the worksheets into the system using a terminal, invoking DPPX GEN3644 to create a parameter table, and loading the parameter table into a 3644 terminal.

DPPX Performance Tool

DPPX Performance Tool is a program product for collecting and reporting information about DPPX/SP performance. You can use it to help tune the system both after initial installation and whenever the system changes.

Among the measurements that can be taken are:

- Processor usage by execution level
- Distribution of dispatch queue by execution level
- Main storage usage
- Direct-access storage usage by device
- Data set, environment, and transient module usage
- Dynamic resident module fetches
- Information about transactions processed by DPPX/SP

At user-defined intervals DPPX Performance Tool stores measurement data in a data set. A report can then be generated from this data set.

DPPX Presentation Services for 3640 Terminals

DPPX Presentation Services for 3640 Terminals (PS3640) is a program product that helps programmers develop and run transaction programs for certain 3640 terminals. It helps with transaction invocation, data formatting, data editing, message assembly, message buffering, time-of-day display, time stamping, transaction time out, and other transaction functions for these 3640 terminals: 3641, 3642, 3644, 3646, and 3647.

Network Management (Program Products)

Distributed Systems Executive (DSX)

The Distributed Systems Executive (DSX) program product provides central library support for a distributed system with DPPX/SP. DSX runs in the host.

With DSX at the host, it is possible to:

- Retrieve DPPX/SP programs, maps, command lists, and other data sets from an 8100 and put them into central libraries at the host.
- Send these items from the host to each 8100 in the distributed system, thus adding, deleting, or replacing data sets in these systems. (DSX also maintains records of these items.)
- Retrieve program fixes from DPPX/SP and put them into the host central library.
- Send a command list to DPPX/SP for immediate execution or for batch execution. For example, you can send the command to install fixes after they are transmitted.
- Send a message to the 8100 operator. For example, the message could tell the operator that fixes have been installed.
- Send print data to an 8100 and retrieve dumps produced at an 8100.
- Initiate a DPPX/SP session by releasing it from the DSX hold status. This is done with a DPPX/SP command issued by the operator at the 8100 site. (Requires DSX Version 2.)
- Display sessions by status, destination, or identification.
- Display and respond to certain messages issued by DSX.
- Enter a command to hold or delete data before it is sent.

Using DSX, your organization can assign application program development to one or more 8100 locations with DPPX/SP and control updates or new releases of the programs through an intervening host. DSX can also be used during installation and tuning, to copy data sets from one 8100 system to other 8100 systems.

Host Command Facility (HCF)

Host Command Facility (HCF Versions 1 and 2) is a network management product that runs in the host (a 308x, 303x, 4300, or System/370). HCF lets users issue most DPPX/SP commands from a terminal connected to the host. This enables installation, control, problem determination, and service of distributed 8100 systems from a central (host) location.

Part of HCF called logical connection verification can be used to check whether data can be transferred between an 8100 Information System and a host. With this and other diagnostic functions possible from the host, problems can be diagnosed and controlled from a central location.

Chapter 3. Configuration Possibilities

Configuration Options

DPPX/SP can be used in two different types of 8100 networks:

- **Interconnected 8100s**—where an 8100 can communicate with another 8100 to which it is attached by a data link
- **Host-connected 8100s**—where 8100s are connected to a host computer (308x, 303x, 4300, or System/370)

Interconnected 8100s

Interconnected 8100s, connected by data link (see Figure 2), operate jointly on one application or a group of closely related applications, in which a program on one 8100:

- Initiates tasks on an interconnected 8100
- Reads from or writes to a data file or data base on an interconnected 8100 (with the aid of user programming)

Other characteristics of interconnected 8100s include:

- Interactive use of applications residing in other 8100s
- No subordination of one system to another with regard to communication or applications
- Optional data link to a host processor for batch data or batch submission

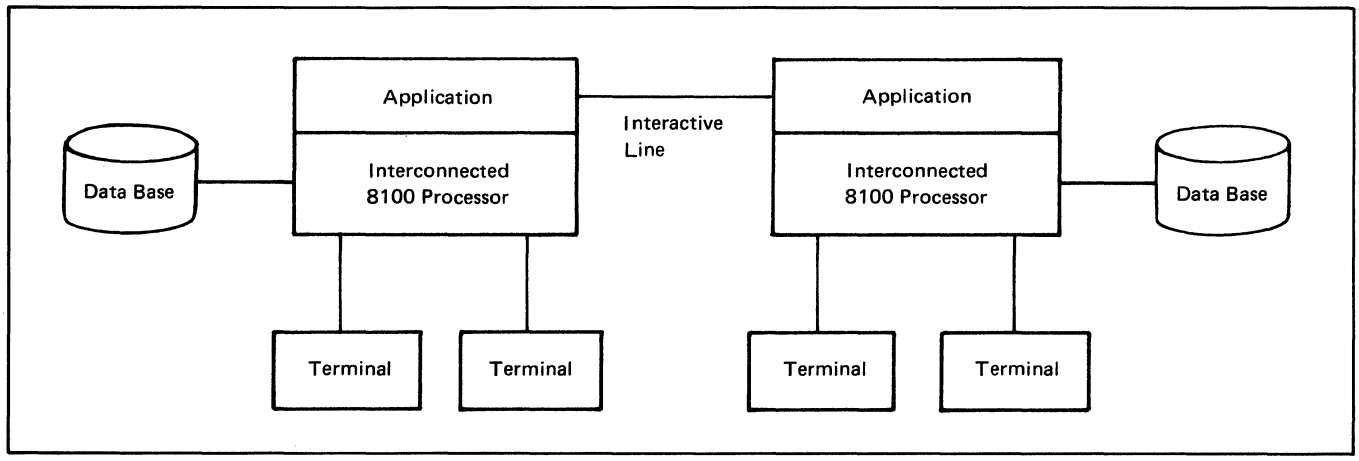


Figure 2. Interconnected 8100s

Host-Connected 8100s

A host computer is the top element in a host-connected system. Distributed processors are satellites to the host site.

A host-connected system (see Figure 3) consists of a host computer and one or more 8100s operating jointly on an application or a group of closely related applications. The host computer can be a 308x, 303x, 4300, or System/370.

A program in one processor:

- Activates application tasks on another system
- Reads from or writes to an online data file or data base on another system (with the aid of user programming)
- Transmits programs to another system for execution

The 8100s may or may not depend on the host for support. You may have an interconnected processor system configured within a host-connected system.

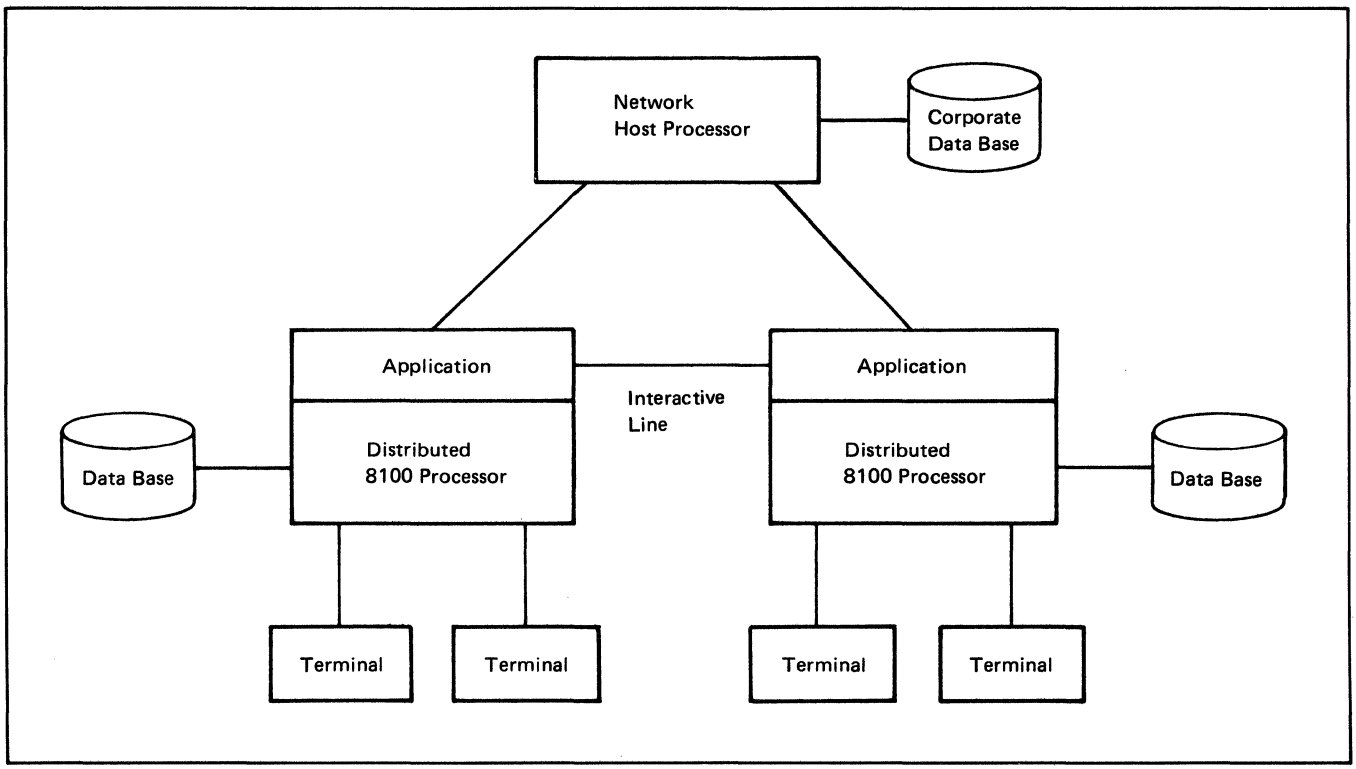


Figure 3. Host-Connected 8100s

Implementing a System with DPPX/SP

There are two general ways to implement distributed systems that have 8100s with DPPX/SP:

- Starting from the host site
- Converting from an existing host-connected system

Starting from the Host Site

This approach can be used for a complex application where various locations must use common data. It requires 8100s connected to a host, as described above.

Programmers develop application programs on an 8100 at the central site. This central site 8100 must have DPPX/SP plus the appropriate program products for application development. The programs are tested with host software to check the connections. When programs have been tested and are ready for use at remote 8100 sites, the application programs can be distributed either on diskettes or transmitted over a data link using the Distributed Systems Executive program product. (DPPX/SP is required in each distributed 8100 to receive and store the production programs.)

This approach uses 8100s with DPPX/SP in two different ways:

- The 8100 at the central site serves as an *application development* system, where users develop application programs that are used later in a production system.

- The 8100s at the distributed sites serve as *production* systems, where application programs already developed and tested are run on a regular basis.

This approach yields a system that is centrally designed and managed from the start through a central data processing department.

Converting from an Existing Host-Connected System

Where installed systems have processing and data concentrated in a host, with terminals at distributed sites, you can add 8100s at the distributed sites and continue to use host programs while developing distributed applications. DPPX/SP in the 8100s makes this possible.

DPPX/SP lets the 8100 function as though it were a communication controller (without modifying data) instead of a distributed processor. So at one time a terminal user could log on through DPPX/SP to a host program. At another time the user could use the same terminal and log on to DPPX/SP and use an 8100 application program.

Application programs could be developed at the host location on an 8100 or at any of the distributed 8100 locations. The only requirement is that the appropriate program products be at the site where applications are developed.

The following chapters illustrate possible configurations as they have been implemented in two different business environments.

Chapter 4. PartsCo—A New DPPX/SP User

PartsCo sells auto parts wholesale through a series of warehouses around the state. Its computer system consists of a 3081 host at company headquarters and terminals and printers in each warehouse.

Dealing with a Problem

The directors of PartsCo must make a decision. Its computer system soon will not be able to handle the needs of a growing company. To serve new warehouses and handle increased volume, the number of terminals and printers connected to the host has tripled since PartsCo installed its first network ten years ago.

The large number of users means a heavy workload for the host computer and for telecommunication lines in the network. Every time someone enters something at a terminal, programs in the host must process it. During a typical day of operation, many problems can occur:

- The system is sometimes unavailable for work because the entire network stops every time a critical application in the host or a telecommunication line fails.
- Response time is poor, reducing employees' productivity and making it hard for them to serve customers promptly and efficiently.
- Vital functions, such as inventory control, are processed too slowly.
- Frequent programming problems arise when such system resources as processing time and storage are overused.
- New application programs cannot be developed because system resources are being used so heavily.

PartsCo has had difficulties maintaining large data bases and controlling access to them. All the data is stored at the central site, taking up disk space and placing an added strain on the host processor when distributed users try to gain access to it. Delays can occur when several people try to use the same information at once.

Finally, PartsCo has had to increase the size of its computer center staff as it has added new equipment. When maintenance or service is needed at a remote site, such as a warehouse, people at the central site are often too busy to provide it immediately; this means increased downtime and lost productivity at the warehouse. People also lose time simply by having to go to a remote site to handle a problem. The computer center staff, then, faces a large workload, and its productivity is reduced.

Deciding on Distributed Processing

The directors at PartsCo decide to see whether distributed processing can help solve their problems. They learn that adding the 8100 Information System does not require them to replace their current 3081 and terminals. Instead, the old equipment becomes an integral part of a new distributed system.

8100 processors at distributed sites can take care of much of the system's workload. Certain jobs are still left to the host. Data bases are maintained in these 8100s as well, freeing resources at the central site. When both processing and data are *distributed* in this way, the system can do more work more efficiently.

The PartsCo directors find that ordering just one product, the DPPX/System Product (DPPX/SP) program product, supplies much of the function they need. They also take advantage of some additional program products to do application development.

When PartsCo installs the the DPPX/SP program product and 8100 processors, they may find that:

- Installation is a simple process because they only go through one installation process with DPPX/SP per site. Because DPPX/SP includes so much function, they need to install fewer program products at each site.

- Customization is simplified because of the easy-to-use panels DPPX/SP supplies.

- System availability increases. With the data processing responsibility spread over several sites, no failure of a single 8100 processor, host processor, or telecommunication line can stop the whole system.

- Response time decreases, improving both productivity and customer service.

- Vital functions are processed quickly and with less stress on resources at the central host.

- Fewer programming problems are likely to occur because system resources are no longer stressed.

- With the addition of a few optional program products to DPPX/SP, new application programs can be developed at an 8100 processor in the central site. This work is independent of the rest of the system and does not disrupt normal processing.

PartsCo finds it easier to control access to data bases with distributed processing. Some data bases, such as inventory, are placed in an 8100, thus moving them closer to the end user. Application programs running under DPPX/SP can access such a data base for normal situations without using resources at the host. Placing data closer to users means less work for the host and reduces communication between distributed and central sites. This results in shorter response times, lower communication costs, and better local control of data.

Distributed processing lets PartsCo manage its system with greater flexibility. Certain data bases—for example, a master file of parts and their prices— are almost autonomous, residing in and being managed by 8100 processors in the warehouses. Others, such as the names and addresses of customers, are kept in the central host and managed there.

With the host's workload reduced through distributed data processing, the system can accommodate increased expansion as PartsCo grows. Expansion is easier: new terminals, printers, subsystems, and 3640 plant communication devices are attached directly to 8100s in warehouses rather than linked via telecommunications lines to the host.

The following sections explain the steps PartsCo goes through once it decides to use DPPX/SP.

Installing DPPX/SP at the Central Site

PartsCo does not install its entire network at once. It starts with a single 8140 processor, connected directly to the 3081 host computer at headquarters. Five 8775 Display Terminals and a 3262 Printer are attached to the 8140.

Some additional program products are added to the system to develop application programs. (These include DPPX COBOL Compiler and Development Management System program products.) Now application programs can be written and tested before they are used in the completed system. New programs will be written and tested here even after the system is fully operational (Figure 4).

Note: PartsCo may be able to use an 8130 or 8150 (with DPPX/SP 2) processor in its system instead of an 8140. From now on, the term "8100" is used to indicate situations where an 8140, 8130, or 8150 of any model may be used.

To install DPPX/SP at the central site, PartsCo's personnel begin by following a set of step-by-step instructions to put all the machinery in place and connect it to the existing host computer.

New customers can install the DPPX/SP program product easily. Utility programs are provided that install the system from diskettes or tapes.

IBM provides panels that appear on the display terminal. These panels guide customers through the customization process. Through the panels, customers can define the configuration for their system.

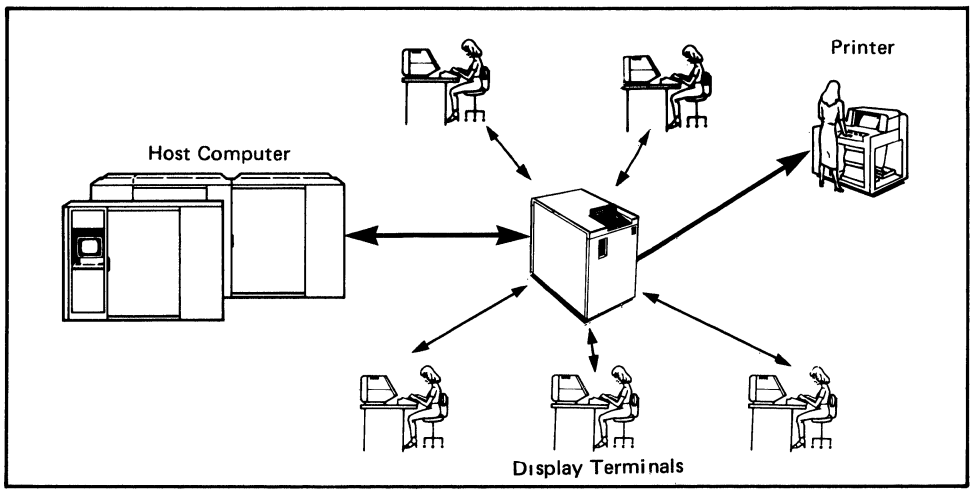


Figure 4. PartsCo's First 8100 System for Developing Application Programs

PartsCo installs several optional program products in addition to DPPX/SP. Each additional program product and its function is described in "Part 3. Additional DPPX and Host-Related Program Products" on page 155.

The products PartsCo installs are:

- **DPPX/SP**
- **Cross System Product/Application Development for DPPX/SP**
- **Cross System Product/Application Execution for DPPX/SP**
- **Data Capture and Management System (DCMS) for DPPX**
- **DPPX/SP Interactive Map Definition (DPPX/SP IMD)**
- **DPPX COBOL Compiler**
- **DPPX APL**
- **DPPX Performance Tool**
- **Distributed Systems Executive** (on the host computer)

Not all of these products will run concurrently in PartsCo's system.

Assuming PartsCo is a new DPPX/SP customer, a PartsCo user, probably a system administrator, installs DPPX/SP by inserting a diskette in the diskette reader and following a procedure that:

1. Starts the system for the first time
2. Asks the user (through the operator panel) for information about certain facets of the system
3. Readies the system to receive any additional diskettes or tapes that contain DPPX/SP
4. Starts a series of panels on the display terminal that guides the user through customization

Writing Programs for the System

After installation, PartsCo's system is ready for use. The programming staff at headquarters begins writing the programs that will handle PartsCo's inventory, control its payroll, and keep track of customers. Certain optional program products help the programmers in their task.

- **Cross System Product/Application Development for DPPX/SP** guides PartsCo's personnel step by step as they create application programs. They write programs in response to prompts (questions or messages that Cross System Product/Application Development for DPPX/SP displays on the terminal screen), without having to learn a programming language or its syntax. After learning Cross System Product/Application Development for DPPX/SP, personnel simply press a key to refresh their memory about aspects of it by reading online tutorials that are available to describe every Cross System Product display and message.

Figure 5 shows a terminal display that a PartsCo programmer sees when writing programs using Cross System Product/Application Development for DPPX/SP.

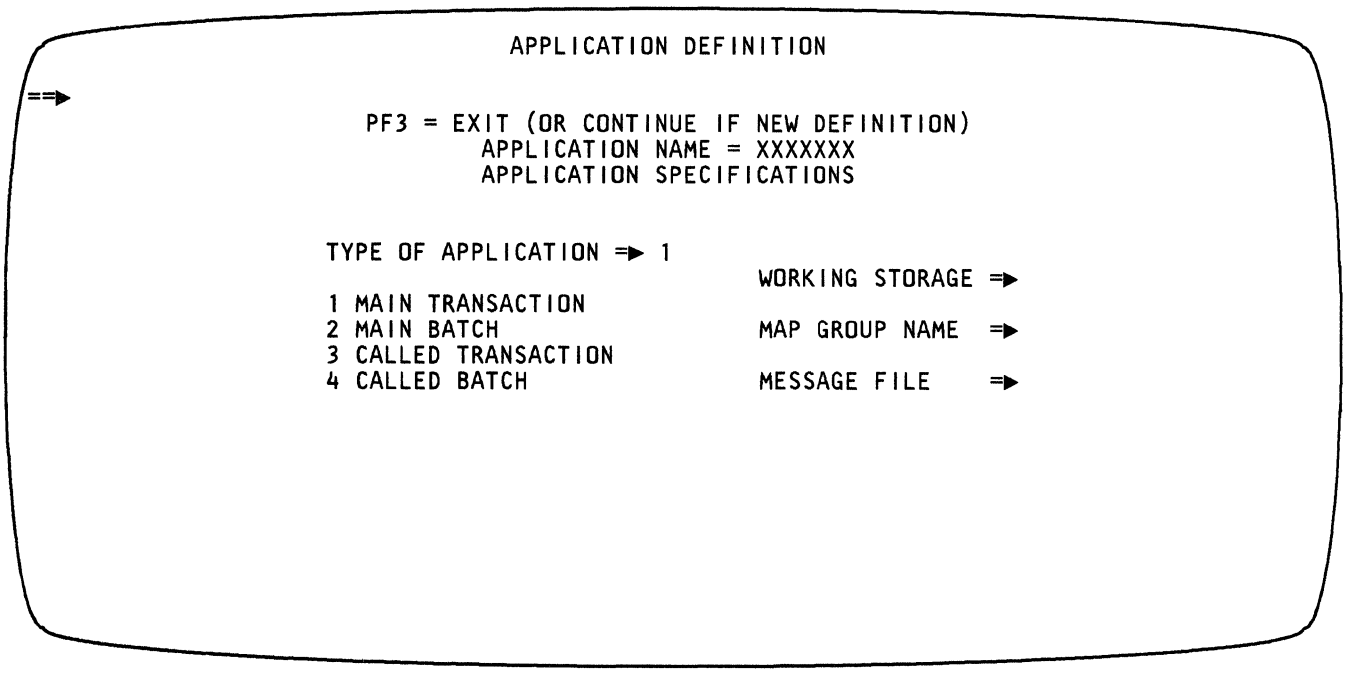


Figure 5. Sample Cross System Product/Application

Programs written with Cross System Product/Application Development for DPPX/SP—especially those that produce maps (panels shown on a terminal screen, with areas for user input)—are usually easier to write and can be developed faster than those written with a programming language.

When the programs are written, Cross System Product/Application Development for DPPX/SP helps the programmer to interactively test, debug, and document them. (They also need the Cross System Product/Application Execution program product to run programs developed with Cross System Product/Application Development for DPPX/SP.)

Cross System Product/Application Development for DPPX/SP works with the **DPPX/SP Interactive Map Definition (DPPX/SP IMD)** program product. But a programmer need not know how to use that product when using Cross System Product/Application Development for DPPX/SP.

- The Cross System Product Set supports DTMS services.
- Because COBOL programs can access more system services than those written with Cross System Product/Application Development for DPPX/SP, PartsCo decides to write some of its application programs in COBOL. They use the DPPX/SP interactive editor to create programs. These programs can be called from a Cross System Product application.

When the programs are written, PartsCo's programmers use the **DPPX COBOL Compiler** program product to compile them. They can also adapt existing COBOL programs from their previous system to their new distributed system simply by recompiling them.

DPPX/SP carries out the instructions in COBOL programs as they run. It does arithmetic, data processing, input (reads), and output (writes).

- PartsCo also uses DPPX APL for application development and interactive problem solving. APL is an interactive language that does not require compilation or link-editing. It contains powerful functions for numerical calculation and data manipulation.

- For programs written in COBOL or another programming language, DPPX/SP and **DPPX/SP IMD** help format and modify the layout of data on display terminals and printers.

  DPPX/SP IMD guides users through designing the printer or screen formats with which programs display information.

- When PartsCo personnel build data bases, the **Data Capture and Management System (DCMS) for DPPX** helps speed the task of entering the data into the system. DCMS works hand in hand with Cross System Product/Application Development for DPPX/SP (see above), providing screen displays (maps) that make it easy for someone to enter data efficiently. DCMS:

  - Saves keystrokes by eliminating repetitive definitions of data items
  - Allows one map to be used for a variety of data entry tasks
  - Lets users decide how their data is formatted
  - Makes sure only properly authorized users can use it
  - Offers "help" displays on the screen to guide and instruct users

  When a program runs using maps created by DPPX/SP IMD, DPPX/SP Format Management controls the transfer of data between the program and a printer or a terminal screen.

Besides the COBOL, Cross System Products for DPPX/SP, and APL selected by PartsCo, DPPX/SP offers these other program products for writing programs:

- The **DPPX Assembler** compiles programs written in Assembler language and processes macro instructions, either defined as a part of DPPX/SP or written by users. Programmers often use Assembler language for writing systems programs, as opposed to programs that process data.

- DPPX PL/I is a versatile language used for business, scientific, and systems programming. A user of DPPX PL/I has access to the services of DPPX/SP as well as other program products. It consists of the **DPPX PL/I Compiler** and the **DPPX PL/I Library** program products.

- DPPX FORTRAN, a programming language best suited to manipulating numbers, is often used for general problem solving. It consists of two program products, the **DPPX FORTRAN Compiler** and the **DPPX FORTRAN Library**.

Testing the Programs

You'll recall that PartsCo's programmers wrote programs in two ways: using Cross System Product/Application Development for DPPX/SP and COBOL.[1]. Someone who uses Cross System Product/Application Development for DPPX/SP can test a program's design, its syntax, and the output it will generate by responding to prompts from Cross System Product/Application Development for DPPX/SP. If the check detects problems in a program, Cross System Product/Application Development for DPPX/SP issues prompts that guide the programmer in fixing the problems. Instead of compiling the program, Cross System Product/Application Development for DPPX/SP prepares the program for execution through a generation facility.

Similarly, someone who uses COBOL can call on:

- The **DPPX/SP interactive editor** to help edit the program and modify it
- The **DPPX COBOL Compiler** to prepare a program for execution
- The **DPPX/SP linkage editor** to combine the various programs in a run unit into a single entity
- The **DPPX/SP COBOL execution services** to perform the program's functions as it runs
- The **DPPX/SP execution debug monitor** to intercept DPPX/SP requests to a display or printer device
- The Map Group Test function of **DPPX/SP Interactive Map Definition (IMD)** to simulate the actions of an application program while still in IMD

Finally, to ensure that the system is running as efficiently as possible, PartsCo uses the **DPPX Performance Tool** program product. The Performance Tool reports how long it takes to perform certain tasks within the 8100 processor. With the data it supplies, PartsCo's personnel can tell:

- Which programs or resources in the 8100 are not working as they should (taking too long to process)
- Whether resources are being overused

The Performance Tool is valuable both when the system is installed and during its entire life.

---

[1] DPPX/SP can run COBOL applications. You need Cross System Product/Application Execution to run Cross System Product/Application Development for DPPX/SP–developed applications. To *develop* applications in Cross System Product/Application Development for DPPX/SP and COBOL, you need the Cross System Product/Application Development for DPPX/SP and COBOL Compiler program products

Installing the Entire Network

When PartsCo finishes writing and testing its programs, it is ready to set up 8100s with DPPX/SP through its entire network. First an 8100 and DPPX/SP is installed in each warehouse and joined by a telecommunication line to the 3081 host. Then existing terminals and printers are attached to the 8100s to make them part of a distributed data processing system. (For a list of all the IBM devices you can attach to an 8150, 8140, or 8130, see "DPPX/SP-Supported Devices by Device Type" on page 104.)

The central site operator can customize the distributed 8100 sites by either:

- Putting a copy of special command lists on diskette or tape and sending them to the distributed site.

  or

- Using **Distributed Systems Executive (DSX)** to send command lists (CLISTs) to the distributed sites. Users at these sites can use the command lists to configure their systems.

Using Distributed Systems Executive, PartsCo can also copy DPPX/SP updates from the central system into distributed 8100s.

Distributed Systems Executive can also increase productivity by letting PartsCo's operators do some maintenance and service from the central site. The system can record problems at distributed sites and report them to personnel at the central site. Some problems that previously would have needed an operator's attention can be handled automatically. Operators may spend less time doing trouble shooting, are free to do other tasks.

Choosing Which Program Products to Use

Besides those it has already installed, PartsCo has other program products available to it. Depending on its special computing needs, it will install them in one or more of the 8100s, using the same procedures with which it installed the program products in the 8140 at headquarters.

The DPPX family of program products is diverse. No one system is likely to use them all. But, depending on your computing needs, you probably will use a combination similar to one of the one presented here or in the following chapter.

Chapter 5. Milwaukee General—An Insurance Company

Milwaukee General Insurance Company, with a home office in Milwaukee and twenty branch offices across the country, has switched from centralized processing with a 3081 host to distributed data processing with the host, 8100s, and DPPX/SP.

The Computing Needs of Milwaukee General

Since the insurance business depends heavily on large, centralized data bases—customer records, actuarial statistics, premium rates—Milwaukee General's new 8100 system with DPPX/SP is similar to the centralized system it is replacing. All important processing is still done by the host at the home office, all data bases are still controlled by the host, and printers, terminals, and systems (such as the Series/1) are still in more or less constant contact with the host. By reducing the system's dependence on host programming and telecommunication lines, however, distributed data processing can:

- Make the system more reliable and accessible by distributing data throughout the network
- Improve the speed with which the system does jobs by accessing transactions at the local 8100 rather than at the host

Figure 6 shows how Milwaukee General's system is set up.

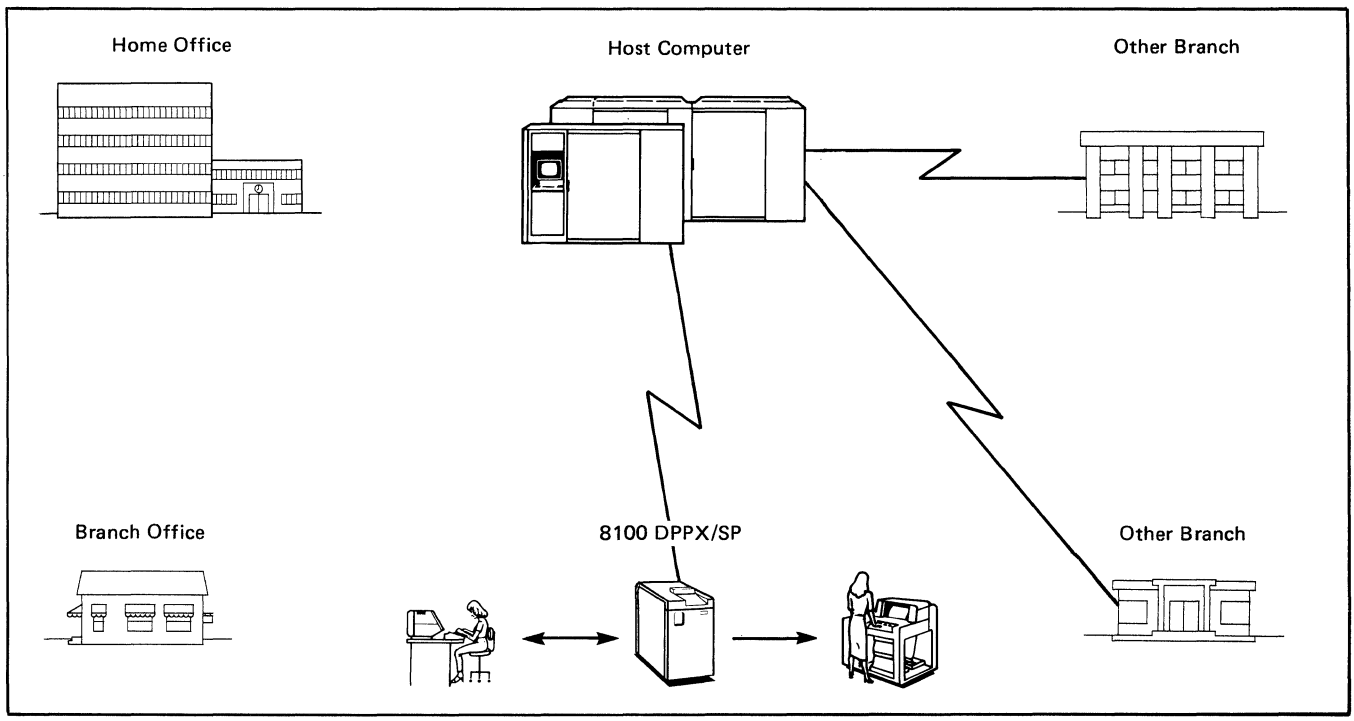


Figure 6. Milwaukee General's Distributed Computing System

In Milwaukee General's former system, terminals and printers in each branch office were directly linked to a communications controller, which in turn was linked by a telecommunication line to the host. Devices such as the 6580 Displaywriter or 5150 Personal Computer worked independently. Every time an employee entered something at a terminal, it was sent to the host, processed there, and sent back. To get a printout of a customer's auto insurance policy, for example, the following process would have taken place:

1. An insurance agent in Detroit entered a print request at a terminal
2. The request was directed to the host in Milwaukee
3. Programs in the host received the request and processed it
4. The host sent two responses back: instructions to a printer in Detroit and an acknowledgment of the request to the agent's terminal

Each 8100 can do certain kinds of processing on its own, freeing the host in Milwaukee for other work. Milwaukee General also connects the 6580 Displaywriter and 5150 IBM Personal Computer to DPPX/SP, making these devices part of the network. Now, when an insurance agent in Detroit orders a printout of a customer's auto policy, the following things happen:

1. The agent enters a print request at the terminal
2. Programs in the 8100 receive the request and process it
3. The 8100 sends instructions to a printer and acknowledgment to the agent's terminal

Notice that the entire job is done in Detroit. This means that:

- The job can be processed and completed faster
- The 3081 host can be free to do other jobs, allowing the system to work more efficiently
- The burden on telecommunication lines can be diminished

The System in Use

The example above shows a very small part of what Milwaukee General's system can do. To see DPPX/SP and associated program products at work, let's look at a different example of processing done at Milwaukee General during a normal workday.

A customer enters the Detroit branch office and asks about an auto insurance policy. A Milwaukee General agent explains the kinds of coverage available to the customer and makes suggestions. When the customer decides what kind of coverage to order, the agent switches on a display terminal and follows these steps:

1. The agent logs on—issues a command to start a session with the computer system.

   As an alternative, the agent could be automatically logged on. A session between the agent at the terminal display and the system is started remotely. When the agent turns the terminal display on, the appropriate logo appears automatically.

2. The system responds by asking for the agent's password.

   The agent types the password, which is not displayed on the screen, and presses ENTER. The password is DPPX/SP's way of ensuring that an unauthorized user does not try to use the 8100 system. The agent is now communicating with DPPX/SP.

3. The agent starts a session with DPPX/SP. DPPX/SP services let the agent use programs in the 3081 host.

4. To use a program in the host—an IMS transaction named PREMIUMS for example—the agent tells DPPX/SP to call it by typing its name.

5. The 3081 program—PREMIUMS—asks for information about the customer: age, marital status, the kind of coverage, and so forth. The insurance agent's screen might look like Figure 7.

   In the picture of the terminal screen, notice that the cursor is to the right of the first arrow. The cursor indicates where the agent is currently typing. Certain parts of the screen are protected. If the agent moves the cursor somewhere other than the position right after the arrows (===>), the terminal will not let the agent type anything. Notice also that the program supplies certain information (the agent's serial number), which will be entered automatically in the proper place when the agent presses ENTER. (The agent could change this number, however, by typing over it.)

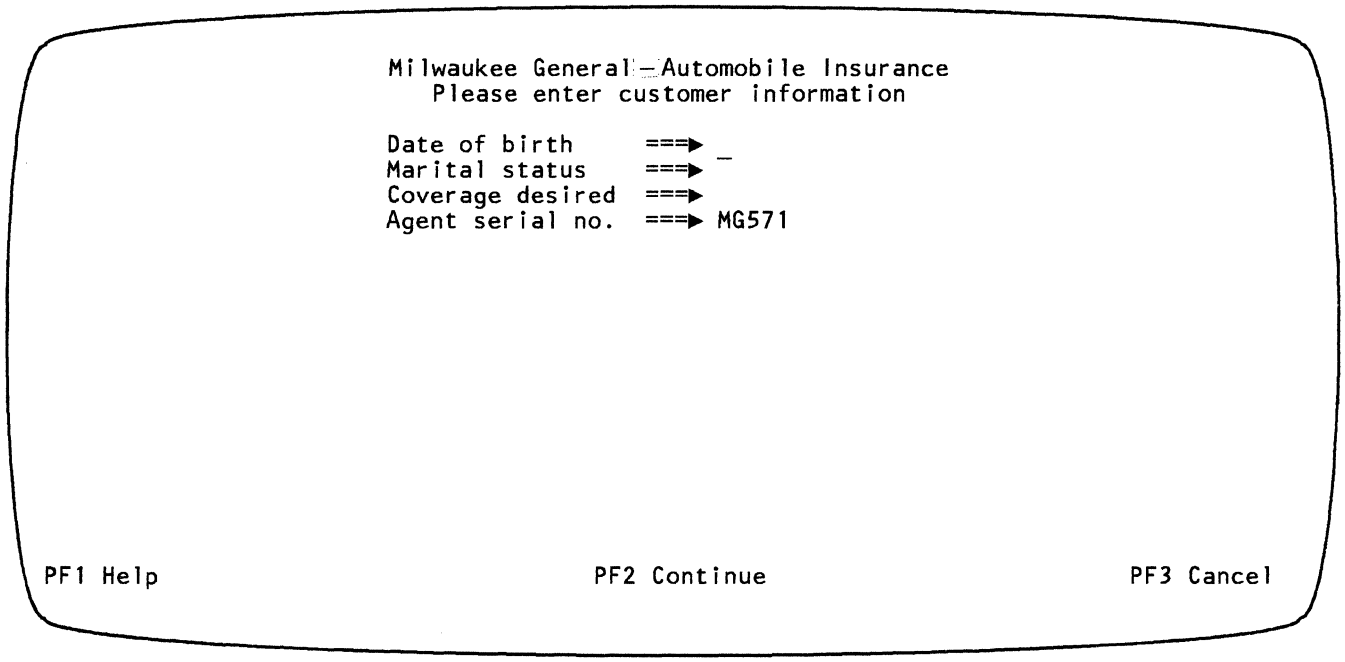


Figure 7. The First Display the Insurance Agent Sees

The program PREMIUMS can:

a. Receive the information when the agent enters it. DPPX/SP transfers information from the terminal to the host through the 8100 in the branch office, making it seem as if the terminal were attached directly to the host.

b. Get actuarial statistics stored at the home office. Program products at the host control the program's access to the data base that contains these statistics.

c. Calculate how much the customer can expect to pay for the insurance.

d. Display the result on the agent's screen. DPPX/SP again transfers the result from the host by way of the 8100 to the terminal.

e. Print the original information and the result on a printer in the agent's office. DPPX/SP sends the print request to the 8100, where it is assigned to the printer. The printer can print the job immediately. If it is busy, it will print the job as soon as it can.

6. If the customer wants to explore other options, the agent can simply call PREMIUMS again and give it new information (for example, a different amount of personal liability coverage).

7. If the customer decides to take a policy, the agent can write the policy with another program in the 3081 host. The agent can even have copies printed both at the branch office and in the home office.

DPPX/SP also makes it easier for the agent to move from one program to another if necessary. For example, suppose another customer comes to Milwaukee General to get auto insurance. This customer already has home owner's insurance through Milwaukee General, and so has personal data on file.

The agent uses the host program PREMIUMS, which asks for much personal data on the customer. Rather than ask the customer all the questions that were asked when he obtained home owner's insurance, the agent seeks the answers in a customer data base, CUSTDATA.

With DPPX/SP facilities, the agent does not have to leave the session with PREMIUMS, log off, and then log back on again to get to CUSTDATA. DPPX/SP lets the agent save the display screen contents from one application when accessing a second.

First, the agent leaves PREMIUMS and requests the selection menu. Included on the menu is a list of the applications the agent can use. (Milwaukee General can change the selection menu as it adds applications.) The agent then chooses CUSTDATA and finds the needed information.

All the agent has to do now is request the selection menu again. The agent chooses PREMIUMS and is returned to the display screen he or she was originally working with.

Operating the System

Occasionally, problems occur in Milwaukee General's system. Some problems can be diagnosed, and often fixed, by personnel at the central site. DPPX/SP and certain other program products make this possible:

1. DPPX/SP can help you perform certain system operator functions. It may function as a "programmed operator." As such, DPPX/SP reduces the work of the operator at each distributed 8100 site. This DPPX/SP service offers several advantages:

   - It handles errors (messages and return codes) in a standard, system-wide way.
   - It allows Milwaukee General to decide what kinds of errors are passed on to its operator, whether that operator is at the central site or a distributed site.
   - It can perform actions automatically at certain times of the day or at certain time intervals.

2. DPPX/SP reports these kinds of events to the 3081 host:

   - Unrecoverable failures in devices or adapters connected to the 8100
   - Certain failures in IBM program products
   - Failures in application programs that Milwaukee General has written
   - ALERT commands issued by application programs

3. **Network Problem Determination Application (NPDA) Version 2**, a program product located in the host, receives information from DPPX/SP and displays it for an operator at the host. (It also reports problems that occur in the host, its links, and devices attached to it.)

   For each error notification it receives, Network Problem Determination Application Version 2 tells the operator

   - The origin of the error notification
   - A description of the error
   - What probably caused the error
   - How the operator can respond to the error

   Network Problem Determination Application Version 2 can notify the operator at the host when DPPX/SP reports errors from any of the branch offices. This tends to reduce the amount of down time, make problems easier to spot, and make the whole system easier to manage.

4. **Host Command Facility (HCF) Version 2**, a program product that runs in the host, allows a host terminal operator to gain access to a distributed 8100 and receive messages from and issue DPPX/SP commands to that 8100.

After NPDA notifies the operator of an error somewhere in the system, the operator can use Host Command Facility to try to fix the problem. If the problem can be solved by the operator at the central site, the time and expense of sending someone to a branch office to fix the problem is eliminated.

With HCF the host operator can control several distributed 8100s simultaneously from one terminal. This makes it easier to monitor several or all 8100s and identify problems that affect all the branch offices.

To see these program products in action, let's imagine this situation at Milwaukee General.

1. A program fails as it runs in the 8100 in one of the branch offices. It issues a series of messages and return codes, which are monitored by DPPX/SP.
2. DPPX/SP receives all the messages and return codes and processes them. For each one, DPPX/SP may do one of several things, determined in advance by Milwaukee General. It may, for instance:
   - Issue one or more DPPX/SP commands
   - Send a message to a user
   - Ignore the situation
3. DPPX/SP receives notification of the program's failure. DPPX/SP then notifies the 3081 host of the failure.
4. In the host, NPDA receives notification of the failure from DPPX/SP. It displays information about the failure, including possible responses, for an operator at the 3081 host. The next step is up to the operator.
5. The host operator can gain access to data bases in the distributed 8100 processor by using HCF. The operator can also try to reenact the problem, take a dump, or do some other procedure by issuing DPPX/SP commands to the 8100 through HFC.
6. After studying the problem, the operator may do other things:

   a. If he can solve the problem on the spot or by talking on the telephone with someone at the branch office, he uses HFC to issue instructions that will fix the problem in the branch office where it occurred. HFC allows the operator to make corrections in the 8100s in the network if they are necessary.

   b. If the operator at the central site suspects the problem is caused by a defect in an IBM product (a machine or program), he can contact the IBM Support Center.

      After he describes the problem and provides certain information about it, IBM Support Center personnel help isolate its cause and fix it. If the cause turns out to be a defect in an IBM product, Milwaukee General may submit an *Authorized Program Analysis Report (APAR)* to have it corrected.

      IBM can then issue a temporary fix, and ensure that the problem is fixed permanently when updates are made to the product in which the problem occurred.

Part 2. DPPX/SP Features and Enhancements

This part summarizes the important functions of the Distributed Processing Programming Executive System Product (DPPX/SP). The next section, Part 3, describes optional program products that are compatible with DPPX/SP. Much of the information included in Part 1 is restated here in greater detail.

Each of these parts can be read in reference fashion. Each chapter describes a separate feature or product.

Note: You should begin with this part if you are familiar with distributed data processing and with the general capabilities of DPPX. However, you might be interested in reviewing "DPPX/SP Releases 1 and 2 share advantages over DPPX Base" on page 6 and "DPPX/SP 2 has advantages over Release 1" on page 7 to acquaint yourself with recent enhancements to DPPX/SP before continuing with Part 2.

Chapter 6. DPPX/SP

DPPX/SP (5660-281) with associated DPPX program products comprises a DPPX operating system. It manages system use, provides for the development of application programs, and can be used in each 8100.

DPPX/SP lets many programs and terminals use the system at the same time. It supports interactive and batch applications. It lets programs and terminals communicate with other programs and terminals, subsystem and workstation attachments, other 8100s, and a host. It has many optional application development tools, and it can be modified to suit unique requirements.

The first few sections of this chapter discuss DPPX/SP in general: how it communicates with users and other systems and how it processes data. The latter sections of the chapter, starting with "DPPX/SP Performs System Management Tasks" on page 58, discuss some of DPPX/SP's specific functions. The DPPX/SP functions highlighted are:

User Interface

DPPX/SP provides two types of user interface: dialogs and commands.

Dialogs display panels on the user's terminal. The panels request the user to choose from a list or to provide information. Dialogs are an easier-to-use interface than commands. After the user fills in the panels, DPPX/SP constructs and issues the DPPX/SP commands necessary to perform the function. The dialog interface is available only for keyboard-display terminals.

Dialog interfaces are available for most system functions. (See "DPPX/SP Provides a Panel Interface to Many Services" on page 64.) Users can also write their own dialogs to perform system tasks.

Dialogs have these characteristics:

- A menu format. The user can select items from a list.
- A fill-in-the-blanks format. Specific fields can be left blank for the user to fill in or can be already filled with default values.
- A help facility to explain any function a user is having trouble performing.
- A tutorial for developing and using dialogs.
- Extensive use of defaults.
- User input retained across sessions.

Commands are one or two words that describe a request. They can be used to access all DPPX/SP functions. With commands, users can define and control the system, initiate and terminate work, and run programs. The command language has these characteristics:

- Users can execute commands interactively or submit them for batch execution.
- Users can invoke commands individually or group command sequences in data sets called *command lists (CLISTs)*. A command list can be invoked interactively or scheduled as a batch job. A command list can use substitution parameters, make unconditional branches forward, and branch on a return code.
- Users can abbreviate commands, add their own commands to the system, and change the names of IBM-provided commands.
- Programs can invoke commands through a call interface.
- Users can invoke many DPPX/SP commands through a help facility. One command lets you display the correct syntax and definition of a requested command. You can use this display to review the correct form of the command or to invoke it directly.

Processing Modes

There are three processing modes for scheduling work: *interactive*, *batch*, and *transaction*. All are invoked by command.

Interactive execution takes place immediately. The terminal user enters a command and waits until execution is completed.

Batch execution is held for execution. Jobs are placed on a queue and the terminal is available for other work. With DPPX/SP 2, 16 batch jobs can be executed concurrently.

Transaction processing provides online response to concurrent requests for services from a number of terminals. See "DPPX/SP Manages Data Bases and Processes Transactions" on page 61 for information on this type of processing.

Addressing Structure

DPPX/SP provides up to 16 million bytes of logical storage (depending on processor model). Real storage is allocated in 2048-byte blocks. A user's address space is made up of logical storage mapped to real storage and appears as a single, contiguous area. A common area is available to all applications for shared programs and data.

Utilities

DPPX/SP has utilities to:

- Initialize tape and disk volumes
- Copy data
- Build indexes for data sets
- Define, change, delete, and list catalogs and data sets
- Dump and restore a disk (this is a stand-alone program)
- Format and print storage dumps
- Trace the system
- Exchange data (via diskette or tape) with other systems

Communication Support

DPPX/SP is designed for the distributed data processing environment. As a result, communication support based on *systems network architecture (SNA)* is an integral part of the product. This includes support for a wide variety of general-purpose and industry-oriented terminals and controllers. (See "Chapter 8. Machine and Program Requirements" on page 103.)

Communications support also includes support for cooperative network management. This allows multiple remote 8100 systems to cooperate with a central site that provides application development and network management using 8100 and host systems at the central site. It also lets you choose to decentralize and provide these functions at the remote 8100 site.

The cooperative network management functions are provided by DPPX/SP communication support operating with certain DPPX program products. (See "Chapter 12. Network Management" on page 135.)

In addition to support of *Synchronous Data Link Control (SDLC),* DPPX/SP also provides limited support for attachment to a host using a *Binary Synchronous Communication (BSC).* The limited support is for the Remote Job Entry service of DPPX/SP and for 3270 BSC. This lets 8100-attached terminals communicate with existing BSC applications in a host. (For more information, see "DPPX/SP Lets 8100-Attached Devices Communicate with the Host" on page 81 and "DPPX/SP Lets Jobs Be Sent to the Host" on page 86.)

Following are some of the specific features of DPPX/SP communication support:

Host Communication: The transaction programs of DPPX/SP in an 8100 can communicate with *Customer Information Control System/Virtual Storage (CICS/VS)* and *Information Management System/Virtual Storage (IMS/VS)* application programs in the host. The programs can be written in DPPX APL, DPPX COBOL, DPPX PL/I, DPPX Assembler language, or by Cross System Product/Application Development for DPPX/SP.

The *Host Transaction Facility (HTF)* of DPPX/SP handles communication. It includes message recovery and resynchronization and is supported by SNA sessions. These functions are provided:

- Send/receive—A DPPX/SP transaction program sends a message to CICS/VS or IMS/VS in the host and receives a reply message. A typical use for this function is to query or update a host data base. The reply message could be an answer to the query or an indication that the data base was updated.

- Dedicated receive—A CICS/VS or IMS/VS application program sends unsolicited data to the 8100 or initiates DPPX/SP transactions. The data received from the host is passed to a DPPX/SP transaction program for processing. The particular transaction program can be specified when the dedicated receiving session is established or when the message is sent by the host.

- Batch capability—A DPPX/SP transaction program sends or receives messages comprised of records in a data base. A DPPX/SP transaction can invoke HTF and request transmission of a batch message. Likewise, in a dedicated receive session, a host system can cause the scheduling of a transaction with a batch message as its input.

In addition to the above features, DPPX/SP 2 features the *HTF Direct* facility. HTF Direct decreases the path length required to process messages within the network by:

- Eliminating a separate *host request complete transaction* to process the host reply message.

- Decreasing the path length required to process a message.

- Transmitting information to a host transaction program without requiring a previous COMMIT.

- Allowing several transmissions to a host transaction program from the same DTMS transaction program.

- Permitting transmissions to different host transaction programs from the same DTMS transaction program.

HTF Direct accepts only synchronous, nonbatch, nonrecoverable messages for transmission.

Adjacent 8100 Communication: SNA terminals, certain non-SNA terminals, and the 3640 devices attached to an 8100 node may establish sessions, via SDLC links, with application programs in adjacent 8100 nodes. Such a terminal connected to an 8100 can access function in another 8100 connected to the first via an SDLC link. (If the DSX and HCF program products are used to service 8100 systems, the serviced 8100s must be host-connected.)

Stand-Alone 8100 Communication: An application program in a stand-alone 8100 can communicate with another application program in the same 8100.

Device-Independent Programming: Because the unit of data transfer is the logical record, programmers need not be concerned with physical device characteristics for communicating with SNA terminals. Application programs simply use READ and WRITE statements (in the high-level languages). (Some non-SNA terminals, such as the 2741 and 2780/3780 BSC-equivalent terminals, are also supported in this manner.)

Link Support: SDLC links may be shared by multiple sessions. Multiple data links from one 8100 to one or more host systems are also supported, as well as the attachment of 8100s via multipoint telecommunication lines to a central 8100 system. In addition, application programs need not be concerned with the physical line to the various attached devices.

Subsystem and Workstation Communication: DPPX/SP supports attachment of a number of systems and devices, including:

- Distributed Processing Control Executive (DPCX) and its Distributed Office Support Facility (DOSF) program product
- 5150 and 5160 IBM Personal Computer
- 5280 Distributed Data System
- Series/1
- 6580 Displaywriter
- 364x Plant Communication Devices
- 4700 Finance Communication System

Problem Determination Tools

Because DPPX/SP users are allocated isolated subsets of system resources (called *environments*), a problem serious enough to cause one user's failure does not affect other users operating with different resources. DPPX/SP attempts to recover from the error. If it cannot, the failing user environment is terminated, and resources in the environment are returned to DPPX/SP.

These tools are available to help find the cause of a problem:

- **Messages and codes.** When an error occurs, DPPX/SP usually issues a message that tells why. If DPPX/SP cannot resolve an error, it issues a code to identify the module that identified the error.

- **Error log analysis.** DPPX/SP keeps an error log containing various types of incident records. The error log may be viewed at a terminal or printed at any time.
- **Connection tests.** An operator can issue a command to make sure that all devices are properly connected. The command causes an IBM-defined or user-defined pattern to be sent from the operator's terminal to the specified device and back.
- **Stand-alone dump.** If the system fails, a stand-alone dump can be taken. An operator loads the dump program from a diskette and executes it. The resulting dump is written to a data set on the diskette. After restarting the system, the operator can print the dump data set.
- **System trace.** A user can start a system trace by issuing a command. The trace records I/O operations and program control flow. Other system activities can continue while the trace is going on, although performance is degraded somewhat. When desired, the user can issue a command to stop the trace. The trace data can be saved in a data set, which can be viewed or printed. The traced records appear in the chronological order in which they were entered.
- **Operator alert.** DPPX/SP alerts the host NCCF/NPDA operator when errors occur at the 8100. These errors include unrecoverable device and adapter errors, failure of an IBM program product or a user application, or an application or user issuing an ALERT command.

Authorization and Passwords

The key element in DPPX/SP data security is its authorization strategy.

For each object known to DPPX/SP (people, programs, data sets, and commands), there is a DPPX/SP structure known as a *profile*. Among other things, profiles contain authorization descriptors called *credentials*. The credentials of each object determine which object it can use and be used by.

One person, such as a system administrator, could selectively assign values to the profiles so that any one object is restricted from access to any other object. When a user without proper credentials attempts to access a data set or use a command, the usage is denied and a message is sent to the system operator. The message can be logged.

The administrator can also control user access of individual system printers and printer queues.

Users can also be assigned passwords that they must enter when logging on. If a user does not enter the correct password, logon is denied. Also a message is sent to the system operator, and the message can be logged.

Data Exchange Using Diskettes

Data exchange is the process in which information is written on a diskette at one system and used in another system. To ensure that the exchange of information can be accomplished efficiently and without errors, certain formats have been established.

DPPX/SP supports data exchange between an 8100 and systems that support these formats:

- **Basic data exchange**, for reading and writing both the IBM Diskette 1 and the IBM Diskette 2
- **Type H data exchange**, for reading and writing the IBM Diskette 2D

DPPX/SP Performs System Management Tasks

DPPX/SP has components that perform various tasks for system management. These components together may be considered the *executive components* that best enhance DPPX/SP operation.

Supervisor

The supervisor is the fundamental part of DPPX/SP. It includes the processor, storage, and program management functions that are necessary to efficiently use the 8100 for distributed data processing. This includes:

- Management of one special-purpose and seven general-purpose hardware interrupt levels.
- Management of up to two processing and control elements.
- Management of logical storage and real storage (see "Addressing Structure" on page 53).
- Advanced techniques for managing storage occupancy by using transient program modules. These techniques are self-tuning, providing ease of use and enhanced reliability.

Environment Management

This function provides dynamic partitioning of 8100 processor resources among multiple programs. Each resource set is called an *environment*. The programs include system-management and application-management functions as well as user applications.

An environment can be protected from others by assigning it a unique logical address space. Alternatively, an address space may be shared among multiple environments for more efficient use of the logical storage space. Dynamic initiation and termination of environments is provided, including automatic recovery of resources upon termination.

Data Management

Data is stored on disk in 256-byte physical blocks. One or more physical blocks make up a *logical block*. A logical block is the unit of a data set that is transferred between main storage and direct-access storage.

All data sets (except stand-alone indexes) have *relative sequential organization* as well as the following characteristics:

- All records in a data set are the same size. For data sets on disk or diskette, minimum record size is 2 bytes and maximum is 4096 bytes. For data sets on tape, minimum record size is 18 bytes and maximum is also 4096 bytes.
- All logical blocks in a data set are the same size. For data sets on disk or diskette, block size may be any multiple of 256 bytes up to 4096 bytes, on tape any size from 18 to 4096 bytes.
- Locking is at the logical block level.

Access can be in one of three ways—**sequential, direct,** or **indexed:**

- **Sequential access** is by record order.
- **Direct access** is by relative record or relative block number.
- **Indexed access** is by key (up to 64 bytes long), as many as 8 indexes may exist concurrently (each accessing the data set with a different key), and indexes and the corresponding data records are kept in separate data sets.

Data sets are grouped into *catalogs*. There are three kinds of catalogs:

- The **system catalog,** which contains profiles of all data sets on the system residence volume, of all user catalogs, and of all volume catalogs.
- **User catalogs,** which contain user data sets.
- **Volume catalogs,** which allow diskettes to be exchanged among DPPX/SP systems.

DPPX/SP 2 features *distributed indexed access method (DXAM)* enhancements that offer more efficient access and storage of indexed files. DXAM improvements include the ability to automatically reorganize data sets (online or offline) so that they demand less storage space. This compression feature does not interfere with normal terminal operations, and the affected data sets may still be accessed and updated during the compression operation. No data deactivation is required. DXAM data set reorganization acts directly on index structure and requires no additional permanent storage.

Keys and Locks

DPPX/SP 2 offers a revised logical storage addressing plan for the 8150 processor. This "Key/Lock" support eliminates redundant *Common Address Space Section (CASS)* previously integrated into each user space. Access to non-CASS storage is controlled by locks and allotted in 32 K blocks. This offers greater customer flexibility in the definition of both CASS and "user-private" storage allotment and allows more address space to be used.

The ACV-based addressing system employed on the 8130 and 8140 is also supported in DPPX/SP 2.

Linkage Editor

The linkage editor converts *object modules* produced by language translators into *load modules* that can be executed. It can also:

- Combine previously link-edited object modules into a single load module
- Combine previously link-edited load modules into a single load module
- Reserve storage for common areas
- Delete program units (control sections) from a load module
- Build an overlay structure

Interactive Debug

Interactive debug allows programmers at terminals to debug programs as they execute. Programmers can place breakpoints in programs (points at which execution is suspended) and display or alter registers and data areas. Programmers must determine the relative storage locations of areas to be debugged. Symbolic debugging is not provided.

Printer Sharing

Printer Sharing allows multiple interactive users, batch users, and application programs to share printers. Print jobs submitted are queued and assigned a priority for printing. Then the system releases them for printing one at a time. The number of printers that can be used depends on the amount of storage available to Printer Sharing.

Interactive Editor

The interactive editor can be used for entering, examining, and changing source programs, command lists, and other data (in character or hexadecimal formats). It operates as a line editor from keyboard-printer and keyboard-display terminals, or as a full-screen editor from keyboard-display terminals with DPPX/SP.

Help panels are also available for new users.

Host Communication

DPPX/SP has components that aid in communication between the 8100 and the host. These components are:

- **Distributed Host Command Facility (DHCF)**—helps create a link between an application program in the 8100 and a terminal user attached to the host. DHCF works under HCF in the host.
- **Host Data Transfer (HDT)**—provides for data transfer between the host and the 8100. HDT works with the Distributed System Executive program product.
- **Host Transaction Facility (HTF)**—lets programs in an 8100 work with programs in the host (for example, CICS and IMS transactions)
- **Problem Determination Application**—alerts the host operator of certain errors in the attached 8100.

DPPX/SP Manages Data Bases and Processes Transactions

DPPX/SP manages access to data bases and processes transactions. This function is called DTMS, short for Data Base and Transaction Management System.

Managing data bases consists of controlling their access and protecting their integrity. Processing transactions consists of providing online response to concurrent requests for services from a number of terminals.

The data base management capability is similar to that in Information Management System/Virtual Storage (IMS/VS) and the transaction processing capability is similar to that in Customer Information Control System/Virtual Storage (CICS/VS).

Data Base Management

A *data base* is a data set described to and processed by the data base manager. The data set can be indexed or nonindexed. DPPX/SP manages data bases in these ways:

- **Controlled access to shared data.** Application programs may be scheduled simultaneously against the same data base. When an application program is updating a record, DPPX/SP can prevent other programs from accessing the record until the updating program has completed its entire update or reached a point of synchronization.

- **Shadow file backup.** The *Shadow file feature* allows the user to maintain a second disk volume as a duplicate of data being written to the primary disk. This feature provides easily accessible backup for crucial data sets and reduces the need for dumping data bases to tape. In the event of primary disk failure, the secondary data base can be accessed without special recovery procedures being performed by the operator. The Shadow file feature duplicates only non-SYSRES files.

- **Re-creation of a data base.** A damaged data base can be reconstructed as long as changes were journaled to an audit file. A utility is provided to do this, using the audit file and a valid earlier copy of the data base. The audit file can reside on disk, diskette, or tape.

- **Reorganization of data records.** Reorganization is not required to recover space freed when data records are deleted. The normal record insertion procedure reuses any freed space.

  After many insertions and deletions, records in a data base might be in an order that no longer allows efficient sequential processing. Therefore, for *indexed data sets*, a DPPX/SP command is provided to reorder records according to the sequence of a frequently used index, to improve sequential processing with reference to that index. Free space is eliminated, and indexes are revised to reflect the new locations.

- **Compression of indexed data bases.** Enhancements to DPPX/SP 2 provide for automatic compression of data sets without data base deactivation. Availability of valuable storage space is maximized. See "DPPX/SP Performs System Management Tasks."

- **Deadlock avoidance.** A *deadlock* occurs when two or more programs try to access the same records at the same time and cannot continue until the records have been accessed. DPPX/SP recognizes potential deadlocks and rejects requests that would cause them. A rejected program is notified and can restart or bypass the request.

Transaction Management

A **transaction** is an exchange between DPPX/SP and a terminal. DPPX/SP provides these transaction services:

- **Transaction program linking.** For applications where performance is not critical, the programmer can write a series of short, related programs and link each to the next by a call to a DPPX/SP service. (This is instead of using READ and WRITE statements in the high-level languages.)

  As each short program, or *segmented conversation*, ends, the system resources assigned to it are freed and reassigned to other users while this user is deciding on and keying a response. DPPX/SP starts the next program in the linked series after the next entry from the terminal is made.

  For applications where performance is more critical than processor storage, the programmer can write *continuous conversational* transaction programs (using READ and WRITE statements in the high-level languages). Such programs do not give up use of system resources during think time and are therefore not as efficient with storage.

- **Concurrent processing of user-written transaction programs.** DPPX/SP handles requests from many terminal users concurrently. The requests can be for the same or for different transaction programs. See "Host Communication" on page 54 for information on how transaction programs communicate with the host.

Testing Aids

DPPX/SP has a *test mode* to let programmers test transaction programs against production data bases without changing them. When a program running in test mode terminates, either normally or abnormally, all changes made in the data base are canceled, and initial values are restored.

Testing can also be done using test data bases.

Debugging Aids

DPPX/SP supplies two facilities to help programmers debug other transactions:

- The DEBUGTR transaction, to execute the next transaction under control of DPPX/SP interactive debug

- The NAMEDUMP transaction, to identify a data set to contain any program dumps during a terminal session

Automatic Logon

In most cases, users log their own terminals on to DPPX/SP. But DPPX/SP also lets a terminal user log another terminal on and lets the logon command be issued from a program or a command list.

This automatic logon capability is useful for starting sessions with output-only devices, such as printers and certain 3640 terminals, from which users cannot log on. It is also useful for logging on terminals and printers when the system is started or when a specific event occurs (such as completion of a certain piece of work).

Control of Operation

An operator can make dynamic adjustments to meet changing user requirements:

- Data bases can be brought online when DPPX/SP is started or later when needed.
- System resources for execution of transactions can be allocated or released without stopping the DPPX/SP transaction monitor.
- Descriptions of data bases and transactions can be added, displayed, altered, or deleted as necessary without halting DPPX/SP.
- The operating status of data bases, transactions, and users can be examined while DPPX/SP is running.
- The audit file can be activated or deactivated while DPPX/SP is running.

Transfer of Data Sets and Programs to Data Base Control

Indexed and nonindexed data sets may be placed under data base management control with no change to or movement of the data sets. Minor command list changes are required, as well as the redefinition of the data sets to data bases.

Existing DPPX/SP conversational and batch programs that do not access data bases may also be defined to execute as transaction programs. No revision or recompilation is required.

DPPX/SP Provides a Panel Interface to Many Services

DPPX/SP lets users request services by responding to questions instead of entering commands. It provides a simple, full-screen interface to most DPPX/SP services.

A DPPX/SP user can enter data in the dialog panels at a terminal. The dialog uses this data to build and issue DPPX/SP commands to operate the system. While the dialog panels do not replace any DPPX/SP commands, they let users issue these commands more easily.

Using Dialogs

Dialogs are easy to use. Most dialogs let users enter information in certain ways. They can:

- Select one task to perform out of several choices. Panels that ask users to select items from a list are called **menu panels.**
- Type information to answer specific questions. Panels that ask users to supply information are called **data panels.**

Some panels supply answers already. These are **default answers.** Users can leave these as is or change them.

Some panels include selection lists and questions for users. These are combined menu and data panels.

DPPX/SP first displays a series of menus and then a set of data entry panels, prompting the user to select the desired function.

The DPPX/SP Dialogs

The DPPX/SP dialog panels help the central site user with system management and application development tasks. They also help the distributed site user, who may have little or no data processing skills, with normal operation and problem determination tasks.

The DPPX/SP dialogs help you with these tasks:

- Using data sets
- Setting up and changing data sets, catalogs, and volumes
- Developing programs and dialogs
- Controlling your system
- Analyzing your system
- Setting Up or changing your system
- Using DPPX/SP commands
- Learning to use DPPX/SP dialogs

Using Data Sets

The dialogs for using data sets let you put information into data sets and review, revise, and print the information. You can list and manipulate the data sets you own.

The dialogs for using data sets let you:

- List the data sets in a catalog
- Look at the contents of a data set
- Edit the contents of a data set
- Copy, move, or replace a data set
- Rename a data set
- Print a data set and handle print requests
- Change the characteristics of a data set
- Delete a data set

Setting Up and Changing Data Sets, Catalogs, and Volumes

The dialogs for setting up data sets, catalogs, and volumes are particularly suited for those key operators and system administrators who control and have general access to data sets on the system. The panels let the user set up and manipulate data sets and catalogs and prepare diskettes and tapes for use by others.

The dialogs for setting up and changing data sets, catalogs, and volumes let you:

- Set up or change your data sets
- Set up or change your catalogs
- Prepare diskettes and tapes for use

Developing Application Programs and Dialogs

The dialogs for developing application programs and dialogs are for application programmers using Cross System Product/Application Development for DPPX/SP, the COBOL Compiler program product, or the DPPX/SP dialog development facility. They let your programmers do most of the tasks associated with application and dialog programming.

Controlling Your System

The dialogs for controlling your system are for system operators, system administrators, and system programmers who will run your system on a daily basis. Users with resource authorization to control shared printers may use the printer dialog.

The dialogs for controlling your system let you:

- Display the status of your system
- Control shared printers and print jobs
- Prepare diskettes and tapes for use
- Set and verify the time and date
- Maintain the operator's shift log
- Submit jobs to the Remote Job Entry (RJE) component of DPPX/SP and manage the RJE queue
- Use your command lists and DPPX/SP commands
- Activate and deactivate adapters
- Control batch processing

Analyzing Your System

The dialogs for analyzing your system are for system operators and help desk personnel who will be gathering information about the operation of your system or about a problem with its operation.

The dialogs for analyzing your system let you:

- Start, stop, and display system traces
- Test device operation
- Print dump data sets
- Display error logs
- Test your 386x modem subsystem

Setting Up or Changing Your System

The dialogs for setting up or changing your system are for system planners and programmers who are responsible for planning, installing, and managing changes on your system.

The dialogs for analyzing your system let you:

- Maintain user profiles
- Maintain environment profiles
- Customize software components
- Set up operational data sets
- Maintain your central configuration

- Display and maintain the list of commonly used DPPX/SP commands
- Manage Printer Sharing resources
- Save and retrieve data bases, indexes, and multiple diskette data sets
- Maintain dialog tables
- Maintain program products
- Maintain user session options
- Generate a remote system

Using DPPX/SP Commands

The dialog for using DPPX/SP commands is for systems or operations people who need to use and save DPPX/SP commands as an alternative to the dialog function. This dialog lets users access the DISPLAY.COMMAND interface.

Learning to Use DPPX/SP Dialogs

The dialog for learning to use DPPX/SP dialogs is a tutorial designed for the first-time user. This tutorial covers:

- An introduction to DPPX/SP dialogs
- Things you can do with DPPX/SP dialogs
- DPPX/SP dialogs and how to use them
- The help panels and how to use them
- Suggested reference manuals

Help Panels

Help panels provide an online user's manual. Help panels can be used in two ways:

- To learn how to use the panels initially
- To get more information about a particular panel function later on

To learn how to use the panels, users can select the first-use tutorial from the initial menu. Inexperienced users should look at this tutorial before attempting any other activity.

To get help with a function any time later, a user can press program function key 1 (PF1) or enter HELP on the command input line. An explanation panel is displayed:

- Help panels for the function selection panels describe the objectives and rationale of the particular function.
- Those that correspond to data entry panels offer guidelines for the data to be entered, instructions for modifying an entry, and references to documents providing more detailed information.

Benefits of Panels

DPPX/SP panels offer a convenient way to accomplish many of the data processing tasks that previously have been handled by a system programmer or administrator. Specific benefits include:

- **Less skill required.** Users do not have to initiate activities. DPPX/SP asks users to enter information by filling in data entry panels. Help panels are available when a user is not familiar with a function or the requested parameters. Constant reference to manuals is not normally required.

- **Increased productivity.** Even experienced professionals can use DPPX/SP to handle ordinary chores more quickly. For example, when maintaining catalogs, a system programmer can browse the contents of various catalogs and data sets, determine which of them are no longer required, and delete them.

  The extensive use of defaults provides another source of productivity. Defaults are provided for initial use by each user. For repeated use, the last values entered on the panel are used as defaults.

- **Fewer errors.** The potential for error is reduced because DPPX/SP minimizes the amount of information requested from the user and uses default values whenever possible. In addition, extensive editing of user input and online instruction in the form of help panels eliminate guesswork.

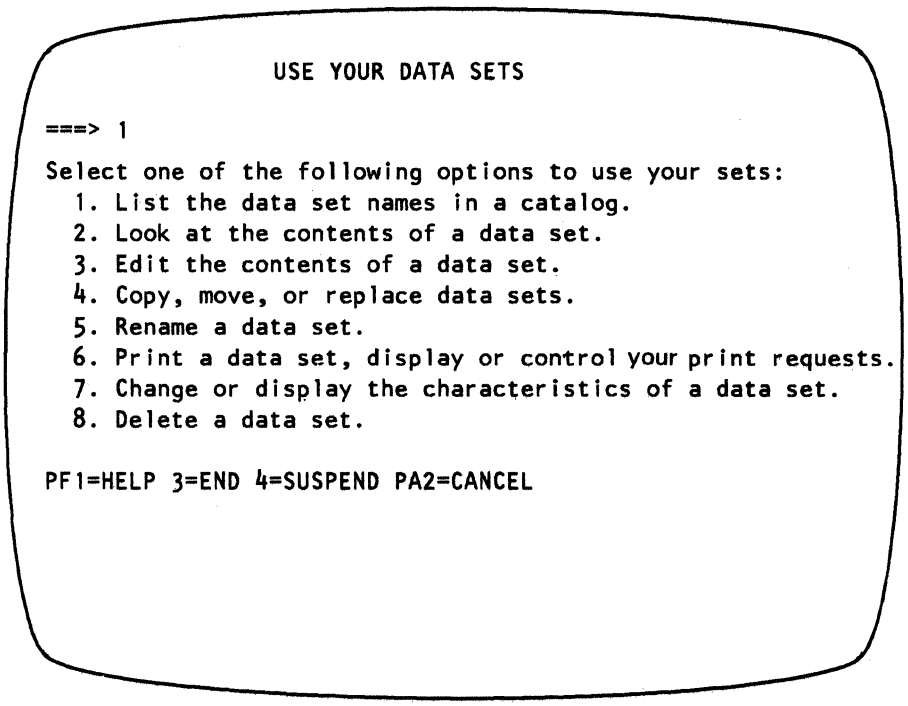
- **Increased function.** The panels accomplish most of their functions through DPPX/SP commands, but they also offer capabilities not provided by the command language. For example, the browse option allows a user to scan a data set by moving it up, down, left, and right to view records longer than 80 characters. The data set being browsed may contain source program code, application data, or print data sets. The ability to scan print data sets is particularly valuable, since the user might decide to delete rather than print the data set.

Examples of Panels

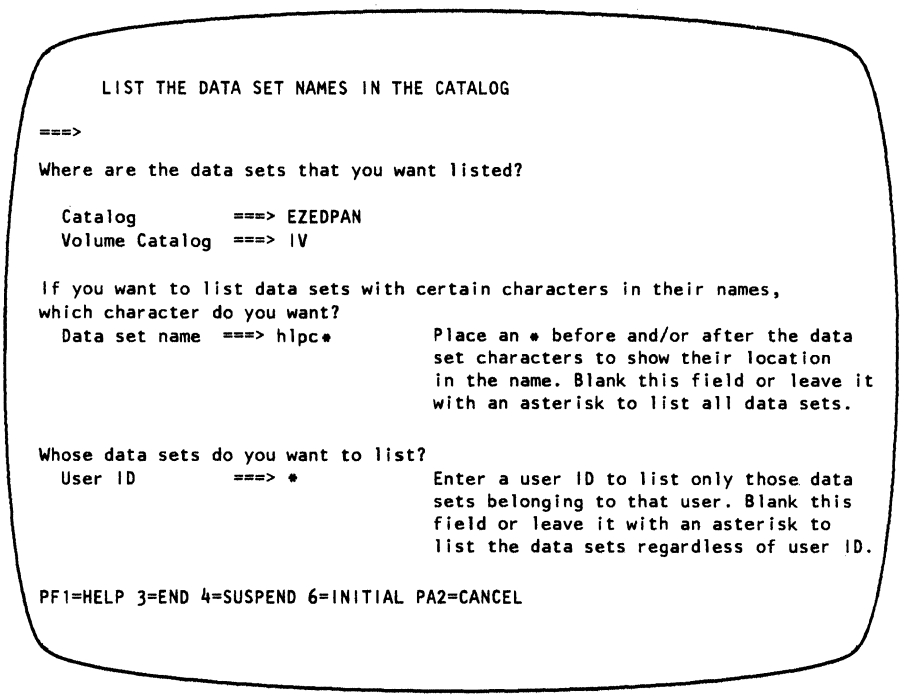
This is the initial menu, which a user receives after entering the IPF command:

```
     USING DPPX/SP - INTERACTIVE PRODUCTIVITY FACILITY

===> 1

Select one of the following options:
  1. Use your data sets.
  2. Set up or change your data sets, catalogs, and volumes.
  3. Develop an application.
  4. Control DPPX/SP.
  5. Analyze the system.
  6. Set up or change your system.
  7. Use and save your DPPX/SP commands.

  T. Learn to use Interactive Productivity Facility.
PF1=HELP 3=END 4=SUSPEND PA2=CANCEL
```

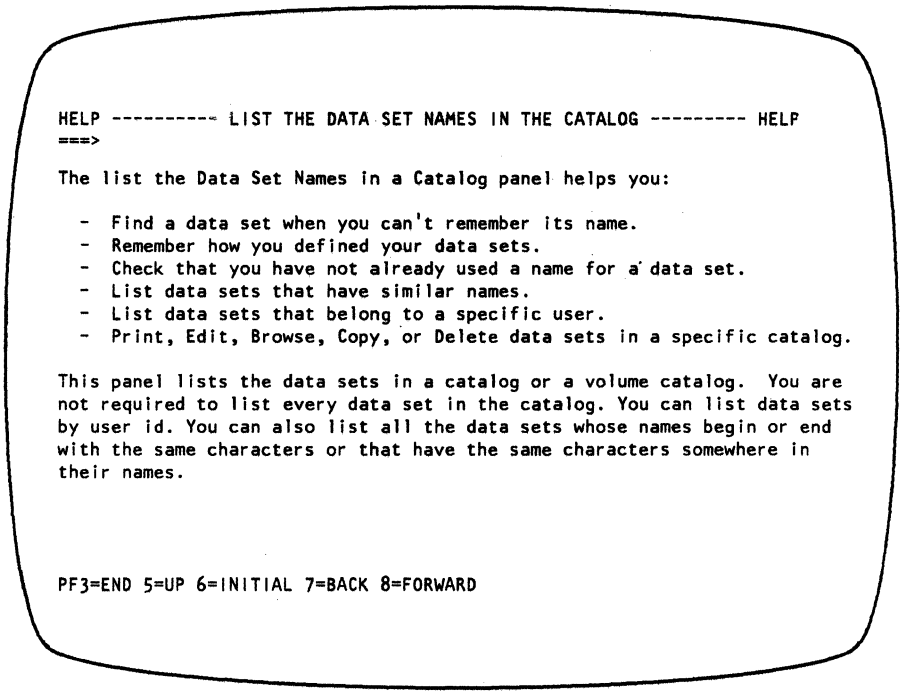
If the user selects option 1, this panel is shown:



To list the data set names in a catalog, the user again selects option 1:

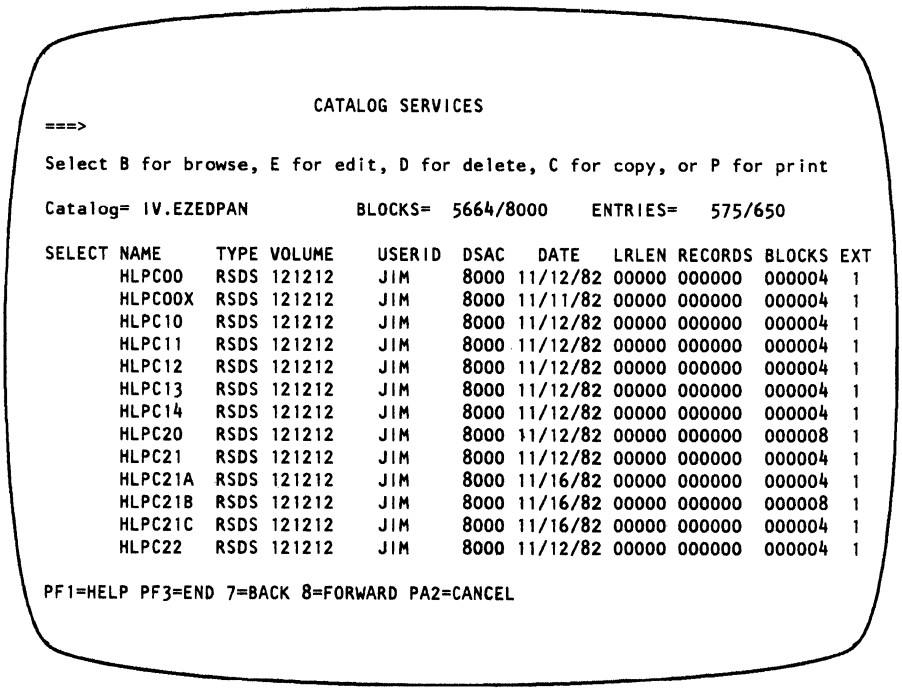


The user who needs help can press PF1 and receive this panel:



```
HELP ---------- LIST THE DATA SET NAMES IN THE CATALOG --------- HELP
===>

The list the Data Set Names in a Catalog panel helps you:

  - Find a data set when you can't remember its name.
  - Remember how you defined your data sets.
  - Check that you have not already used a name for a data set.
  - List data sets that have similar names.
  - List data sets that belong to a specific user.
  - Print, Edit, Browse, Copy, or Delete data sets in a specific catalog.

This panel lists the data sets in a catalog or a volume catalog.  You are
not required to list every data set in the catalog. You can list data sets
by user id. You can also list all the data sets whose names begin or end
with the same characters or that have the same characters somewhere in
their names.



PF3=END 5=UP 6=INITIAL 7=BACK 8=FORWARD
```

The final panel of this dialog sequence is shown below:



```
                          CATALOG SERVICES
===>

Select B for browse, E for edit, D for delete, C for copy, or P for print

Catalog= IV.EZEDPAN          BLOCKS=  5664/8000    ENTRIES=   575/650

SELECT NAME     TYPE VOLUME    USERID  DSAC   DATE    LRLEN RECORDS BLOCKS EXT
       HLPC00   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC00X  RSDS 121212    JIM     8000 11/11/82 00000 000000  000004  1
       HLPC10   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC11   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC12   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC13   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC14   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC20   RSDS 121212    JIM     8000 11/12/82 00000 000000  000008  1
       HLPC21   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1
       HLPC21A  RSDS 121212    JIM     8000 11/16/82 00000 000000  000004  1
       HLPC21B  RSDS 121212    JIM     8000 11/16/82 00000 000000  000008  1
       HLPC21C  RSDS 121212    JIM     8000 11/16/82 00000 000000  000004  1
       HLPC22   RSDS 121212    JIM     8000 11/12/82 00000 000000  000004  1

PF1=HELP PF3=END 7=BACK 8=FORWARD PA2=CANCEL
```

DPPX/SP Lets You Write Your Own Panel Dialogs

In addition to providing panels to do specific DPPX/SP services, DPPX/SP lets you write your own panel dialogs. These user-written panels can take advantage of the DPPX/SP panel facilities, including the use of help panels and function keys. You can use the standard set of PF keys in your panels, and you can write your own help panels to tell the user how to use the dialog.

For example, you might want to write a panel dialog to:

- Start and stop parts of the system or network
- Set the time and date
- Handle an error log
- Start application programs
- Associate output to different devices

DPPX/SP Lets Users Switch Smoothly between Applications

DPPX/SP lets DPPX/SP terminal users move from one application to another without having to log off or on each one. This function is called the *Router*. The Router provides a simple interface through which operators can access different applications. These applications can be anywhere in the network. The Router does *not* interfere with the connection between an application and the operator.

System administrators can define the Router interface with the panels DPPX/SP provides or through DPPX/SP commands entered normally and stored in command list data sets. The first method is recommended. Administrators then do not have to predefine data sets.

Selection Menu

The Router function of DPPX/SP communicates with operators through a *selection menu*. This menu can be changed to fit an installation's needs. The Router selection menu fits in well with installations whose systems use menus heavily. It includes:

- A list of the applications the device or operator can access, as defined by the system administrator
- The status of each application, indicating its availability
- Notification if any output is pending on the application

The operator must choose which application to work with. The selection menu does not supply a default application.

Users work with that application until they request the selection menu again. They can use the Program-attention key on the display terminal keyboard to get the selection menu back again, or can set a PA or PF key.

The administrator can define a different set of applications and different terminal characteristics for each user. When a user's needs change, the administrator can again change their definitions with the DPPX/SP panels.

With the selection menu, users can access a new application without ending the current one. For example, an operator can work with both a program in a host and a program in an 8100 during the same session.

What the Router Function of DPPX/SP Lets You Do

The Router function lets an operator access many different applications without logging on and off between each application. However, the Router also has these functions:

- Applications can automatically log terminals on to DPPX/SP. There are two types of automatic logon situations. In the first, the terminal is automatically logged on and DPPX/SP starts all sessions with applications. A terminal user sees a logo or the selection menu when the terminal is turned on.

  In the second, the terminal is automatically logged on by applications. The applications start all sessions. The terminal user sees the selection menu

when the first application initiates a session. (If the terminal is powered off, all application initiations will fail. They must be restarted when the terminal is powered on.)

- Sessions can be initiated by an application or by the operator. If the host link fails, DPPX/SP does not disconnect from the terminals. Instead, it shows an UNAVAILABLE or SELECTABLE status on the selection menu for all host applications.

  Application-initiated sessions show UNAVAILABLE until the host recovers. Then they are listed as WAITING. Operator-initiated sessions show SELECTABLE until they are selected. Then they either show WAITING or will show it when the host recovers. The operator does not have to take special action to reconnect to the host. The operator simply reselects the application.

- DPPX/SP lets the operator save the screen contents from one application (and still accept data for it) when switching sessions even if the operator has started working with another application. (However, this technique of "suspending and restoring" by switching sessions costs storage.)

  A user might want to suspend and resume a session when answering customer questions. Suppose an insurance operator were working with a local data base and received a customer inquiry. The operator could suspend the data base session and switch (through the selection menu) to the application that contained the customer's information. When finished, the operator could select the first application and resume working.

- Operators can use a *control session interface*. This lets operators have dynamic control of the sessions. They can request the *shared screen operation* and issue control session commands. For more information, see "Control Session" below.

- A device or operator can have many sessions with multiple local and remote applications throughout the network.

- Two systems with DPPX/SP can communicate with each other. See "Routing Communication" on page 77 for more information.

- There is no limit to the number of sessions a single device can have with DPPX/SP. However, a user may want to restrict this number because of storage or performance.

- DPPX/SP provides a single session passthrough support.

Control Session

The control session interface is for experienced programmers and system operators. It is also available to users who need this more dynamic session control.

The control session interface allows operators to have full dynamic session control, including defining access to sessions.

Operators can enter DPPX/SP routing commands. These are listed under "Routing Commands". They can also request the *shared screen* option on menu selection.

The main users of the control session interface are:

- System operators communicating with several DPPX/SP-like subsystems to control multiple systems with DPPX/SP.
- Programmers who need the session control commands that the shared screen makes available.

An operator with a shared screen can receive system operator messages from several interconnected systems with DPPX/SP on the screen. The operator could also enter commands to each of these systems through the shared screen.

Routing Communication

DPPX/SP supports sessions between peer 8100s. However, to do this you need:

- A copy of DPPX/SP in both 8100s (in each domain)
- An established session between the two systems

These sessions must be started by the local system, not the remote one.

Routing Commands

DPPX/SP routing commands are available to operators under the control session interface. Commands are in three groups:

- Those the control session operator can issue from a menu or from the shared screen
- Those the control session operator can issue from the shared screen
- Those the control session operator can issue through the NOTIFY facility of DPPX/SP

Hardware Supported

You can use the selection menu only on LU type-2 (3270-compatible) terminal displays with a minimum of 80-character-wide displays.

DPPX/SP also supports nondisplay devices, such as printers and downstream controllers in passthrough mode. See the list of supported printers and controllers under "DPPX/SP Lets 8100-Attached Devices Communicate with the Host" on page 81 for which ones are supported here.

With DPPX/SP, a user does not need *Data-Stream Compatibility/Co-Domain Passthrough (DSC/CDP).* However, the routing function of DPPX/SP requires users to define devices differently from DSC/CDP.

Software Supported

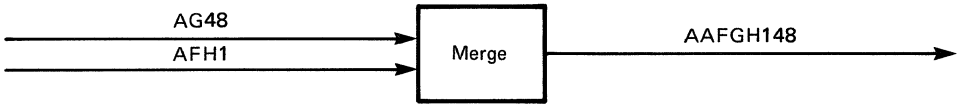
In the host, CICS/VS and IMS/VS are both supported. DPPX/SP supports any other host application that communicates with LU type-2 display terminals.

DPPX/SP Sorts, Merges, and Copies Data

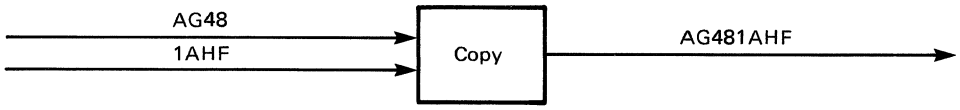
The DPPX/SP Soft/merge facility lets users sort, merge, and copy data. In the following examples each character represents a record.

Sorting rearranges records into a specified order. For example:

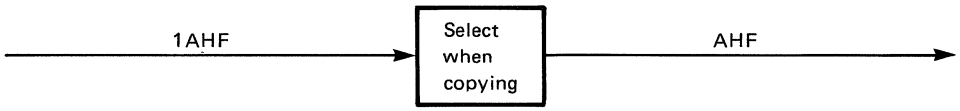
Merging combines ordered records. For example:



Copying reproduces records. For example:



When sorting, merging, or copying, records can be selected based on a comparison with a constant (less than, equal to, greater than, and so forth). For example:



Methods of Invoking Sort/Merge

Users can invoke this DPPX/SP service in any of these ways:

- Conversationally, by entering the SORT command and being prompted for subcommands. DPPX/SP detects any errors as the subcommands are entered, and the user can immediately correct them
- Immediately but not conversationally—by putting the SORT command and its subcommands in a command list and invoking the command list
- As a batch job—by putting the SORT command and its subcommands in a command list and submitting the command list to batch
- In combination with an application program:
  - By putting the SORT command within an application program.
  - By invoking an application program as an exit routine. Records are passed one at a time between DPPX/SP and the application program.
  - By setting up a command list or otherwise arranging for application program routines and this service to be invoked sequentially.

Allowable Input Sources

Records to be sorted, merged, or copied can come from:

- As many as seven data sets on disk, diskette, or tape
- A card reader
- A terminal
- An application program

Allowable Output Destinations

Sorted, merged, or copied records can be put in:

- A data set on disk, diskette, or tape
- An application program

Work Data Set

When copying or merging, DPPX/SP does not use auxiliary storage. When sorting, however, it requires auxiliary storage for a work data set only if there is not enough primary storage.

Control Fields (Keys)

Control fields are the fields in records that DPPX/SP uses to order the records. A control field:

- Can be anywhere within a record.
- Can be contiguous or noncontiguous to another control field.
- Can overlap another control field.
- Must be a whole number of bytes long.
- Can be character, signed decimal zoned, signed decimal packed, or numeric (fixed-point). It cannot be floating-point.
- Need not be in the same place in every record.
- Can be as long as 255 bytes, unless the field is fixed-point. Then, 4 bytes is the maximum.

DPPX/SP can sort or merge on more than one control field. As many as 16 control fields can be specified. Such a composite control field can be as long as 512 bytes.

Record Length

All input records must be the same length.

The longest record that can be copied or merged is 4096 bytes. The longest record that can be sorted is 4096 bytes, minus the length of the composite control field, minus 10.

Maximum Number of Records

DPPX/SP can sort, copy, or merge as many records as can fit on the output device, with one exception. For sorts that require a work data set, DPPX/SP can sort only as many records as will fit in the work data set.

Collating Sequence

Character fields can be ordered according to EBCDIC, ASCII, or a user-specified collating sequence, in either ascending or descending order. A user-specified sequence is helpful, for example, to order data containing characters not in the English alphabet.

Numeric fields are ordered numerically, with all zero fields treated as equal regardless of their signs.

Performance Considerations

DPPX/SP's execution speed when sorting and merging data depends on the system resources available (such as main storage) and the presence or absence of other programs competing for use of those resources.

Specific ways to improve performance are:

- Cut down on the time used for input and output. This is accomplished when:
  - Record and block sizes of input and output data sets are large.
  - The composite control field is short.
  - Input and output devices with fast data transfer rates are used.
  - Input and output data sets are on separate volumes.
  - A work data set, when required for sorting, is not on the same volume as the input and output data sets.
- Avoid use of a work data set.
- Give DPPX/SP more main storage.

DPPX/SP Lets 8100-Attached Devices Communicate with the Host

DPPX/SP lets certain display terminals, printers, controllers, and processors attached to 8100s communicate with host or 8100 DPCX application programs as though the devices were directly attached by data link to the host processor. This function is called Data-Stream Compatibility.

DPPX/SP as a Transition Aid

DPPX/SP makes it possible for a centrally organized enterprise to ease into distributed processing. 8100s with DPPX/SP give users of certain terminals and controllers and processors the ability to log on to a host application or to DPPX/SP.

In this way, terminal users can access these host programs:

- VTAM applications
- TCAM direct applications
- IMS/VS
- CICS/VS
- TSO/TCAM or VTAM
- BTAM
- VM/370

In addition, you can log on to DOSF in another 8100 system.

Aside from logon, the connection through the 8100 is not apparent to users.

SDLC and BSC Links Supported

DPPX/SP supports the *Synchronous Data Link Control (SDLC)* and *Binary Synchronous Communication (BSC)* line-control disciplines for communication between host computers. DPPX/SP supports only SDLC for communication between a host and a downstream controller or processor and a keyboard-display or printer. BSC support is limited to the Remote Job Entry function of DPPX/SP and to 3270 BSC.

DPPX/SP also supports SDLC links to an 8100–Distributed Processing Control Executive (DPCX) host.

When using an SDLC link, the request for host connection can come from the keyboard display at the 8100 system, a program in a downstream controller or processor, from the host, or from the 8100/DPCX system. Transmissions from DPPX/SP can share the same data link with transmissions from other 8100 applications. Functions that use structured fields, such as color and programmed symbols, are supported only for SDLC host connections.

Devices Supported

These keyboard-display terminals are supported for this host communication function:

- IBM 3104 Display Station attached via directly attached loop
- IBM 3178 Display Station attached to a 3274 or 3276 attached via data link, directly attached loop, or data-link-attached loop.
- IBM 3276 Control Unit Display Station attached via data link, directly attached loop, or data-link-attached loop
- IBM 3277 Display Station attached via display and printer attachment
- IBM 3278 Display Station attached to a 3274 or 3276 attached via data link, directly attached loop, or data-link-attached loop
- IBM 3279 Color Display Station, attached to a 3274 or 3276 attached via a data link, directly attached loop, or data-link-attached loop
- IBM 3290 Information Panel attached to a 3274 attached via a data link, directly attached loop, or data-link-attached loop
- IBM 8775 Display Terminal attached via directly attached loop or data-link-attached loop

These printers are supported for host communication:

- IBM 3230, 3262, or 3287 attached to a directly attached or data-link-attached loop
- IBM 3284, 3286, 3287, and 3288 attached to a display and printer attachment
- IBM 3268, 3287, or 3289 attached to an IBM 3276 Control Unit Display Station

These controllers and processors are supported:

- IBM 3601, 3602, 3605 (U.S. only), and 5995 (World Trade only) Finance Communication Controllers
- IBM 3651-25,75 Store Controller
- IBM 3684-1,2 Point of Sale Control Unit
- IBM 4700 Finance Communication System
- IBM 8130, 8140, and 8150 Processors with DPPX/SP or DPPX Base
- IBM 8130, 8140, and 8150 Processors with Distributed Processing Control Executive (DPCX), including the Distributed Office Support Facility (DOSF) program product

- Series/1 system
- 5280 Distributed Data System
- 3640 plant communication devices
- 6580 Displaywriter
- 5150 Personal Computer
- 5160 Personal Computer

The Terminal User's View

A DPPX/SP user starts a session by logging on. Afterward, the user is unaware of DPPX/SP because his or her session is with a host or an 8100/DPCX application.

If the user is already familiar with the terminal and the application, no additional education is necessary.

Data Security

DPPX/SP does not alter host application validation procedures but merely passes the information to the host application.

In addition, the user can be required to provide a password when logging on to DPPX/SP.

Copy Support

Users can print a copy of their terminal screen image. DPPX/SP supports such "local copy" from:

- 3178s, 3276s, and 3278s to printers attached via the same 3274 or 3276.

Users can print host data on their DPPX/SP-attached printers. DPPX/SP supports such "host-initiated copy" from:

- 3277s to printers attached via the same display and printer attachment.
- 3178s, 3276s, and 3278s to printers attached via the same 3274 or 3276.
- 3104s and 8775s to printers attached via the same or another loop. To use host-initiated copy from the 8775, the associated 3287 must be dedicated for host-initiated copy.
- 3290s to printers attached via the same 3274.

APL Support

DPPX/SP accepts input from the 8775, 3276, 3278, 3279, and 3290 with APL feature installed and creates the necessary data stream for host communication.

Performance Consideration

Response times will be longer when using an intermediate 8100 with DPPX/SP than when the terminals, controllers, or processors are directly attached to a host computer or to 8100/DPCX.

Compatibility Consideration

Most existing host application programs written to:

- the 3270 BSC interface

  or

- the SNA interface of the host software supported by the Data-Stream Compatibility component of DPPX

will run with little or no change when using DPPX/SP.

Restrictions

The Data-Stream Compatibility function of DPPX/SP has these restrictions:

- Character sets are limited to EBCDIC only.
- The data stream to devices attached via display and printer attachment is limited to new line and form feed.
- BSC dial is not supported.
- BSC communication for the 8140C and 8150 Model B in dual mode requires the Programmed Communications hardware feature.
- No 3270 structured field support with a BSC line (no support for multiple screen partitions, extended color, or programmed symbols).
- Display-to-display copy is not supported for the 8775 or for displays attached via display and printer attachment.
- A BIND parameter with MAX RU X'00' is not acceptable and requires a new BIND image.

For downstream controllers and processors, these functions are not supported or may require modifications to host or downstream programs:

- Remote power-off capability is not supported.
- The LAST option is not supported for session termination, and therefore the NOT LAST option is assumed.
- Remote diagnostics and application fixes are available only when supported by active DPPX/SP sessions.

- The *physical unit (PU)* services available are those supported by the current level of DPPX/SP.
- *Threshold and remote analysis (TARA)* support is not available for 3600 control units attached to an 8100 system with DPPX/SP.
- *Advanced communication function (ACF)* enhanced recovery capability (ACTPU/ACTLU) is not supported.

Migration

Host application program upgrading is not necessary. Customers with an installed systems network architecture (SNA) host who are using *request/response unit (RU)* sizes greater than 1024 bytes might have to regenerate their host CICS/VS system if using an 8775 as the target display terminal. The same is true for those who use RU sizes greater than 512 bytes and have 3277s as the target display terminal.

DPPX/SP Lets Jobs Be Sent to the Host

DPPX/SP lets OS/VS, DOS/VS, VSE, and VM/370 jobs be submitted by communication lines to a host computer from an 8100. A job can come from a data set or card reader at the 8100. Output from a job is returned from the host to a data set or printer at the 8100. This is done by the Remote Job Entry service of DPPX/SP.

Configuration Support

The configuration to support the Remote Job Entry service of DPPX/SP consists of:

- The DPPX/SP program product.
- One or more workstation input devices. (A *workstation* is one or more I/O devices in a group.) The input devices can be card readers, disks, diskettes, or magnetic tape.
- One or more workstation output devices. The output devices can be printers or card punches.
- An optional workstation console to control this DPPX/SP service. The console can be a keyboard-display terminal or the DPPX/SP operator function. (DPPX/SP provides a type of "programmed operator." See "DPPX/SP Performs Many System Operator Functions" on page 92 for information on that function.)

A workstation console is not required because the Remote Job Entry service can run unattended.

A workstation's I/O devices need not be physically next to the 8100 or to one another.

Using DPPX/SP's Remote Job Entry Service

The DPPX/SP Remote Job Entry service may be used in these ways:

- To support job output from existing host application programs. The host programs may be used from 8100 terminals through DPPX/SP. Printer output could be stored at the host for later transmission to the remote 8100. Overnight, the host subsystem could transmit the output to the unattended 8100 where it is stored for printing the next day. In the morning the 8100 operator prints the output using DPPX/SP Printer Sharing facilities.
- Similar to the above, but using direct printing of job output during the day, while the 8100 is attended. (Direct printing at an unattended 8100 is possible. See "Controlling the Remote Job Entry Service" on page 88.)
- Output from DPPX/SP application programs may be stored as data sets at the 8100. The output could subsequently be transmitted to the host as input to a host application program. The results could then be returned to the 8100, as in the first two examples.

Devices Supported

DPPX/SP supports these devices for the Remote Job Entry function:

For Input

- Disk storage
- Diskette type 2D (must have format created by DPPX/SP)
- IBM 3501 Card Reader
- IBM 2502 Card Reader
- IBM 3521 Card Punch with Card Read feature
- IBM 8809 Magnetic Tape Unit

For Output

- Disk storage
- Diskette storage
- IBM 3262-2,12 Line Printer
- IBM 3289-3 Line Printer
- IBM 3521 Card Punch

SDLC or BSC Transmission to Host

Jobs can be transmitted between the 8100 and the host using either Synchronous Data Link Control (SDLC) or Binary Synchronous Communication (BSC) multileaving.

With SDLC, DPPX/SP allows more than one host session at the same time on a link. For example, output from the host could be printed at the same time data is being read and transmitted to the host. The maximum number of concurrent sessions is dictated by your site's resource and performance requirements or by the host job entry subsystem. Transmission can be via point-to-point switched lines, point-to-point nonswitched lines, or multipoint nonswitched lines. With SDLC, you can have a request unit size of 3840 bytes. This lets data be transmitted more quickly.

With BSC, support is for multiple input and output data streams (maximum of seven for a printer or punch). Transmission can be via point-to-point nonswitched lines.

Nondedicated Workstation Devices

Nondedicated Input and Output Devices

Workstation input/output devices are not permanently assigned to this DPPX/SP service. They can be reassigned by command to other program products or to user programs as business requirements change.

Nondedicated Console Device

You do not need to have a workstation console to run the Remote Job Entry service. You can run this service unattended. If you have a workstation console, the operator of this workstation can log off or let another user become the operator without disrupting the Remote Job Entry service.

Job Input

Jobs can be submitted to a host from a workstation card reader or from a data set on disk, diskette, or tape.

Multiple jobs can be submitted from the card reader for immediate transmission to the host without operator intervention between job streams. (A card reader working this way is often called a "hot card reader.")

Multiple data sets can be concatenated (logically connected together) for transmission to the host. Each data set may contain data alone, job control statements alone, or both.

Each data set submitted to the host can be assigned one of two priorities. The higher-priority data sets are transmitted first.

Jobs submitted to the host can be displayed or canceled any time before transmission.

Job Output

Using SDLC, job output from the host can be printed or punched as it is received, or job output can be stored on disk or diskette for future printing or punching. Based on information included in the data stream from the host, DPPX/SP can assign different print classes to output data saved on disk. Future printing or punching is done when the jobs are submitted to DPPX/SP Printer Sharing. DPPX/SP supports up to 255 print classes and multiple print queues.

Using BSC, job output from the host is printed or punched as it is received.

Controlling the Remote Job Entry Service

An operator can enter commands to and receive messages from DPPX/SP and the host job-entry subsystem. Specifically, an operator can:

- Start and stop the workstation.
- Log workstations on and off the host job-entry subsystem.
- Allocate workstation devices.

- Send commands to the host job-entry subsystem. (DPPX/SP transmits the commands to the host system without interpreting or modifying them.)
- Submit jobs to the host system for processing.
- Request the status of the operation, previously submitted jobs, and the host job-entry subsystem.
- Control printing and punching for output jobs from the host system.

The operator controls operation through DPPX/SP.

The optional workstation console can be the same device as the DPPX/SP operator's console.

You can keep a record of messages sent to the console and commands issued from it. Such a console log can be kept in a data set and later displayed or printed.

Host sessions can be restarted automatically if a link fails.

When running unattended, DPPX/SP lets jobs and output data continue flowing in overnight transmission.

Forms Control

Vertical alignment of forms (paper) at workstation printers is controlled by *forms control blocks (FCBs)*. Each printer has a default FCB stored in the 8100. Users can write their own FCBs.

Under BSC or SDLC, when a printer is started, DPPX/SP uses the default FCB unless a user-defined one is loaded. Under SDLC, forms control can be accomplished automatically based on information included in the data stream from the host.

Prestored Job Control Statements

Job control statements may be prestored at either the 8100 or the host for concatenation to their appropriate data files.

Host Control of Workstations

With SDLC only, the Host Command Facility (HCF) program product can be used to control the 8100 from the host. The host system can then start and control workstations.

Preparing for Remote Work

Preparing for this DPPX/SP function requires definitions at both the host system and the 8100 system:

- At the host system, definitions for the system control program, job-entry subsystem, network control program or emulation program, and access method
- At the 8100, definitions during DPPX/SP installation

DPPX/SP Alerts a Host Operator of 8100 Errors

DPPX/SP monitors activity in an 8100 processor and alerts the operator of an SDLC-connected host when errors occur. It can alert a host operator of:

- Unrecoverable device and adapter errors
- Certain failures of an IBM program product or a user-written application program
- An ALERT command issued by an application program or terminal user

DPPX/SP helps you manage a network that contains a host and one or more 8100 processors.

DPPX/SP receives a copy of all events recorded by the DPPX/SP error log manager. For the events listed above, it generates "alerts" for transmission to the host. An alert is a certain type of systems network architecture (SNA) error indicator (RECFMS RU type 00).

In the host, the Network Problem Determination Application Version 2 (NPDA) program product records the information received from DPPX/SP. In addition, the host operator can display this information.

Who Can Generate Alerts?

Alerts can be generated by:

- **DPPX/SP.** DPPX/SP generates an alert for an unrecoverable device or adapter error, certain failures of an IBM program product, or failure of an application program.
- **A user.** A user-written application program or an authorized terminal user can issue an ALERT command to notify the host of some condition in the 8100.

  Although alerts are often thought of as problem warnings, this ability offers other possible uses. For example, when the last input transaction of the day is received by an application program, that program could send an alert to the network operator to tell him or her to schedule a Distributed Systems Executive (DSX) session. That session could be used to transfer the transaction data from the 8100 to the host.
- **A subsystem.** DPPX/SP passes along to the host any alerts received from systems attached to the 8100 (such as the 3600 Finance Communication System or Series/1).

The Benefits of the Problem Alert Function

DPPX/SP helps you manage your system more efficiently with these advantages:

- Early error detection. DPPX/SP reports problems in the 8100 processor and in devices attached to programs running in the host. This early error detection may reduce downtime by making the whole process of problem determination go faster.

- Faster problem determination. Because DPPX/SP detects errors, you should need less time and effort to diagnose problems in the system.

- Greater control in managing large configurations. Programs in the host can monitor problems in certain parts of the 8100 network and disregard those in other parts of the network.

  Because error information arrives at a central location—the host—you should have less need for skilled personnel to detect and describe problems at remote sites.

DPPX/SP Performs Many System Operator Functions

DPPX/SP performs many functions otherwise requiring system operator action. This is done by the *Programmed Operator Function* of DPPX/SP. The Programmed Operator Function reduces effort required to operate an 8100.

DPPX/SP intercepts messages and codes targeted for the 8100 system operator and applies predefined responses to them. The responses can be user-specified or IBM-specified. (IBM supplies default responses that users can change as they decide on their own responses.)

This "programmed operator" action can also intercept information for the operator handling work at distributed DPPX/SP sites.

Programmed responses are kept as action statements in an indexed data set. Programmers define, display, delete, and change the action statements by issuing commands.

In the action statement data set there are entries to handle all system messages and return codes that can be issued to the 8100 operator. Each message and return code may have a unique programmed response or may be grouped with messages and codes that have a common response.

DPPX/SP, providing for a programmed operator, can also be used to perform actions at specified times or at the end of specific time intervals.

Many Programmed Responses Possible

Users can define these actions for DPPX/SP to take in response to a message or return code:

- Send the message or return code:
  - To a DPPX/SP Command Facility user who has been designated lead operator
  - To a user who is logged on at the host system through the Host Command Facility (HCF) program product and who has been designated host operator
  - To all users logged on to the DPPX/SP Command Facility
- Take no action (only log the message)
- Execute a command list
- Execute a program
- Solicit a response from a lead or host operator
- Reply to a prompt from the system

Users can also program the above actions to occur at specified times of the day or at the end of specific time intervals.

Responses Can Be Grouped

As previously mentioned, action statements define not only the type of action to be performed, but also whether the action applies to a specific message or return code or to a category of them. Such categories are:

- Message type—information, action, problem determination, return code, or timer event. For example, all problem determination messages, no matter what the source, could be routed to the host operator.
- Message type and issuing DPPX/SP segment. For example, all information messages from the Printer Sharing component of DPPX/SP could be routed to a terminal operator at the 8100. Or all information messages from the DPPX COBOL Compiler program product could be discarded after being logged.
- Message type and issuing DPPX/SP subcomponent.
- One specific message or return code.
- Any message or return code for which no other action is defined. In other words, a default action can be specified for all messages and return codes not otherwise defined by action statements.
- User messages (without an ID) may optionally be grouped together or processed individually.

Messages and Responses Can Be Logged

As the system operates, a record is kept in the 8100 operator session log of all messages and return codes issued and the system responses applied to them. The messages and responses can later be analyzed if desired.

Messages and Responses Can Be Monitored

A terminal user can track all messages and return codes going to, and all actions taken by, the DPPX/SP Programmed Operator Function. This ability is useful when defining and debugging a new action.

Benefits of DPPX/SP as "Programmed Operator"

DPPX/SP offers these benefits when it provides for a programmed operator:

- The method in which messages and return codes are handled becomes standardized. This can eliminate many of the inconsistencies normally involved in the management of a network of distributed processors.
- IBM customers can establish their own criteria for determining which messages require routing to a human operator and whether that operator should be at the 8100 or at the host system.
- Users can define actions to be performed at specific times of the day or at the end of specific time intervals.
- Customers can reduce the operation activity required at remote sites.
- DPPX/SP helps develop more centralized control of system operations, especially in problem determination.

DPPX/SP Runs COBOL Application Programs

COBOL is a programming language suited to developing business applications. It is especially efficient in the description and manipulation of files of formatted data. The name COBOL stands for COmmon Business-Oriented Language.

DPPX COBOL is a level-1 implementation of 1974 American National Standard (ANS) COBOL (ANSI X3.23-1974). It also has most of the features in the December 1975 Federal Information Processing Standard (FIPS), PUB 21-1, low-intermediate level.

DPPX/SP contains reentrant routines that perform arithmetic, logic, data conversion, and input/output operations. Another program product, the DPPX COBOL Compiler (5760-CB1), checks syntax and generates object code.

The DPPX COBOL Compiler program product does not have to be installed in every 8100 that has DPPX/SP. A development 8100 could have DPPX/SP and the Compiler product to compile and test application programs. A production 8100 could have just DPPX/SP to run programs that were compiled and tested on the development 8100.

Application-to-Application Communication

Two user-written COBOL application programs can communicate with each other. The programs can be in the same or different 8100 processors. After certain configuration commands are issued, READ and WRITE statements within the COBOL programs control the sending and receiving of data between the programs.

Command Invocation from within a Program

Through a call to a DPPX/SP library routine, DPPX/SP or user-written commands can be issued from a COBOL program.

Calls to Non-COBOL Programs

A COBOL program cannot directly call a program coded in another high-level language. But a COBOL program can call an Assembler language program, which in turn can call a program coded in another high-level language or the Cross System Product.

Shared Storage

DPPX/SP library routines and user programs can be made resident. They can then be shared by all DPPX COBOL programs running concurrently. This improves performance.

OS/VS COBOL Comparison

Most differences between DPPX COBOL and OS/VS COBOL have to do with the language definition.

The major differences other than language content are:

- DPPX COBOL will generate reentrant code. All its library subroutines are reentrant.
- In DPPX COBOL, all the library subroutines are handled dynamically (being either in shared storage or loaded during program initialization). Users cannot choose to link them with their load modules.
- There is no execution-time symbolic dump facility (SYMDMP option) in DPPX COBOL.
- Options for run time (execution time) differ.
- Arithmetic results in DPPX COBOL will not always be the same as those in OS/VS COBOL, for the following reasons:
  - If diagnostic messages are issued during compilation, arithmetic results may not be the same.
  - For a divide operation (not the DIVIDE verb), DPPX COBOL may carry more precision than OS/VS COBOL.
  - Truncation, when necessary, affects the rightmost digits for DPPX COBOL.

DPPX/SP Presents Data according to User-Developed Maps

DPPX/SP helps programmers specify how an application program should present data on display terminals and printers.

DPPX/SP controls the transfer of data between the application program and the display screen or printer, based on the information defined using the Interactive Map Definition (DPPX/SP IMD) program product. Programmers use DPPX/SP IMD during the development stage of an application program. DPPX/SP IMD guides the user through the process of designing screen and printer layouts that DPPX/SP uses when the application program is run.

DPPX/SP IMD makes it quicker and easier to format screens for displays and printers. The programmer uses a series of menus and questions to design the screen. The programmer can see the screen layout as the process proceeds. If changes are necessary, they can be added easily.

Separation of Logic from Data Formatting

With DPPX/SP and DPPX/SP IMD, an application program's logic is separated from data formatting information, making coding simpler.

For example, it is irrelevant to the logic of an application program whether a particular piece of information appears at the top, bottom, or middle of a screen.

The formatting information is kept separately in maps.

Using Maps

Maps contain information about the format in which data should appear on a screen or printer. Programmers create maps before compiling their programs.

Before DPPX/SP can use a map, a user must generate that map using DPPX/SP IMD. Generation also produces DPPX COBOL, DPPX PL/I, or DPPX Assembler language declarations, called *application data structures*. These correspond to the layout descriptions stored in the map. They can be retrieved from the appropriate source statement library when the application program is compiled. Examples of statements to do this are the COPY statement in DPPX COBOL and the %INCLUDE statement in DPPX PL/I.

These statements would also be used to include a *control structure* for passing control information between the application program and DPPX/SP during execution.

Run-Time Use of Maps

When the application program is run, DPPX/SP generates correct data formats based on the application data structure generated by DPPX/SP IMD.

The programmer can also write exit routines to do such special processing as validate data or transform encoded data for a specific map.

Maps Can Be Distributed

Maps created under DPPX/SP IMD at one 8100 can be sent for execution by DPPX/SP to other 8100s where DPPX/SP IMD is not installed. The maps can be sent using the Distributed Systems Executive (DSX) program product or Cross System Product Set for DPPX/SP export/import facilities.

Outboard Formatting and DPPX/SP Data Formatting

Outboard formatting is a means of supporting distributed data processing for applications on systems network architecture (SNA) connections.

This can reduce line loading when an 8100-attached terminal is communicating with a Customer Information Control System/Virtual Storage (CICS/VS) program in the host. The application programmer using CICS/VS Basic Mapping Support (BMS) can redistribute the display services work so that some is performed by the host (inboard) and the remainder by the 8100 (outboard).

Display formatting is controlled by maps (in the host by BMS, in the 8100 by DPPX/SP), with the constant data held outboard in the 8100. The variable data from the BMS-generated data stream is merged with the constant data to complete the presentation formatting.

Existing host application programs do not need changing, but the BMS map must have a flag set to indicate that outboard formatting is required, and the CICS/VS terminal control table (TCT) must be changed to indicate that the terminal accepts outboard maps. DPPX/SP accepts such host output for both basic and advanced display devices, but outboard formatting is not available for printers.

With outboard formatting, DPPX/SP IMD and DPPX/SP offer the user other benefits:

- The host application program can use display features, such as programmed symbols and trigger fields, even though it was not written originally to take advantage of those features.
- Input data can be locally validated by means of user-written exit routines, thereby further reducing line traffic to the host, since only valid data is transmitted.
- The Screen Definition Facility/CICS (SDF/CICS) unload utility provides support for moving BMS source maps outboard to the 8100, where they can be imported by DPPX/SP IMD, and the outboard maps generated.

Typically, an application using outboard formatting would have displays attached to one or more 8100s, linked to a host computer with a Synchronous Data Link Control (SDLC) connection. At least one of the 8100s would require DPPX/SP IMD to import SDF/CICS maps into the map specification library, to carry out any editing that may be required and to generate the maps.

DPPX Users Migrating to DPPX/SP

If you are migrating from DPPX base to DPPX/SP, you may be familiar with a program product called *Distributed Presentation Services (DPS).* Application programs that run under any version of DPS will run under DPPX/SP without modification.

DPPX/SP IMD accepts source maps created by DPS Versions 1 and 2. DPS Version 1 maps are converted to DPPX/SP IMD format before they are edited by DPPX/SP IMD. DPS Version 2 source maps can be used without conversion, but you should convert them if you want to use all the functions and get the best results. You can convert source maps back to DPS Version 2 format, but not to Version 1 format. Object maps generated by DPPX/SP IMD require a DPPX/SP system for execution.

DPPX/SP provides all the existing format management functions and application interfaces for DPS, plus the enhancements listed below. If you are currently using DPS Version 1, read both "Differences between DPS Version 2 and DPPX/SP" and "Differences between DPS Version 1 and DPS Version 2." If you are using DPS Version 2, read "Differences between DPS Version 2 and DPPX/SP."

Differences between DPS Version 2 and DPPX/SP

DPPX/SP includes all the function of DPS Version 2 plus:

- Improved magnetic stripe reader support.
- Execution debug monitor extended for printers.
- Faster map processing.
- Stored logical records. These can be used to:
  - Let the operator redisplay the contents of a screen.
  - Merge the output data and the corresponding keyed input data before passing this information to the application program.

Differences between DPS Version 1 and DPS Version 2

DPS Version 2 has all the functions of Version 1 plus:

- Outboard formatting.
- Support for extended highlighting (underscore), color, and programmed symbols on printers that support them.
- Interactive programmed symbol definition, generation, and run-time loading.
- Improved programmer productivity by reducing the keying necessary during IMD field naming and map modification, and simplified application programming for data arrays.
- Improved support for Kanji devices.
- Improved performance support to allow line printers to run closer to rated speeds.
- Support for DPPX PL/I.

Chapter 7. Choosing the Right Program Products

The Program Products by Task

DPPX/SP works with associated program products to form an operating system or with host program products to form a network. It manages system use and can be used in each 8100. The other program products you choose depends on your application. To help you decide, the program products are listed below by task. (Summaries of all program products appear in "Part 3. Additional DPPX and Host-Related Program Products" on page 155.)

To develop application programs that process transactions and access data bases:

- *DPPX COBOL Compiler.*
- *DPPX PL/I Compiler* and *DPPX PL/I Library.*
- *DPPX/SP Interactive Map Definition (DPPX/SP IMD).* To help APL, Cross System Product, COBOL, PL/I, and Assembler language programmers develop display screen and printer layouts during program development.
- *DPPX Presentation Services for 3640 Terminals (PS3640).* With the Interactive Transaction Generator (ITG) feature, to provide transaction definition services for certain 3640 terminals.
- *Data Capture and Management System (DCMS) for DPPX.* To develop data entry panels.
- *Cross System Product/Application Development for DPPX/SP.* Differs from COBOL and PL/I. It is an online, menu-driven tool, not a programming language. It has these advantages over a programming language:
  - It requires less programming experience and skill and time.
  - It is compatible with other IBM Cross System Products, thus providing application transportability.
  - It takes the initiative by prompting the user for information that it can turn into a program. Contrast this with high-level language programming, where the programmer must determine which statements to code and in which order.
  - It makes technical system interfaces transparent.

Note that Cross System Product/Application Development for DPPX/SP might not be advantageous when:

- You have existing host COBOL or PL/I programs that you want to recompile and run on the 8100
- You need access to more system services than Cross System Product/Application Development for DPPX/SP allows

If you use Cross System Product/Application Development for DPPX/SP, you need the DPPX/SP IMD program product. You also need Cross System Product/Application Execution for DPPX/SP to run applications developed with Cross System Product/Application Development for DPPX/SP.

To develop other types of programs:

- *DPPX APL*, for interactive problem solving and application development
- *DPPX FORTRAN Compiler* and *DPPX FORTRAN Library*, for general problem solving
- *DPPX Assembler*, for system programming
- *DPPX PL/I*, for general problem solving and for system programming.
- *DPPX Parameter Table Generation Facility for IBM 3644 Automatic Data Unit (GEN3644)*, to help customize the 3644 terminal

To run application programs:

- *Cross System Product/Application Execution for DPPX/SP*, to run Cross System Product/Application Definition–developed programs.
- *DPPX PL/I Library*, to run PL/I-coded programs
- *DPPX Presentation Services for 3640 Terminals (PS3640)* with the Execution Manager (EM) component, to run programs that use transaction maps generated by PS3640 Interactive Transaction Generator (ITG)
- *Data Capture and Management System (DCMS) for DPPX*, to control the entry of data batches

To increase productivity:

- *Cross System Product/Application Development for DPPX/SP*, described above

To make an 8100 system with DPPX/SP work better:

- *DPPX Performance Tool*, which supplies information helpful when tuning a development or production system

To manage a network:

- *Host Command Facility*, which gives a user of a host-connected terminal access to DPPX functions (this program product runs in the host)
- *Distributed Systems Executive*, which provides library control from a host-connected terminal (this program product also runs in the host)

The Host Command Facility and Distributed Systems Executive program products are in the host. They do not reside in the 8100.

The Program Products by System Type

Another way to consider the program products is in terms of which ones might be used in an application development system and which ones in production systems.

Although any of the program products could be used in either type of system, they are listed under the system in which they are most used.

You can use these program products in an application development system:

- DPPX/SP.
- Cross System Product/Application Development for DPPX/SP.
- DPPX COBOL Compiler.
- DPPX APL.
- DPPX PL/I Compiler.
- DPPX PL/I Library (for test-executing PL/I programs).
- DPPX FORTRAN Compiler.
- DPPX FORTRAN Library (used when compiling FORTRAN programs).
- DPPX Assembler.
- DPPX Parameter Table Generation Facility for IBM 3644 Automatic Data Unit (GEN3644).
- DPPX/SP Interactive Map Definition (DPPX/SP IMD).
- DPPX Presentation Services for 3640 Terminals (PS3640) with the Interactive Transaction Generator feature.
- Data Capture and Management System (DCMS) for DPPX (for developing data entry panels and controlling the entry of data batches).
- DPPX Performance Tool. (You can use information gained from using the Performance Tool to tune other production systems that don't have the Performance Tool installed.)

You can use these program products in production systems:

- DPPX/SP
- Cross System Product/Application Execution for DPPX/SP
- DPPX FORTRAN Library
- DPPX PL/I Library
- DPPX APL
- DPPX Presentation Services for 3640 Terminals (PS3640) with the Execution Manager component
- Data Capture and Management System (DCMS) for DPPX

Chapter 8. Machine and Program Requirements

This chapter lists storage requirements, devices supported, and host programming supported.

Storage Requirements

The storage requirements of any configuration of program products and user application programs depend on the size and complexity of programs to run concurrently. Figure 8 shows some sample application-development configurations that perform well.

Disk Size Requirements

The system residence volume requires a disk size of 58 megabytes or greater.

Processor*	Environments	Terminals
8150 Model B with 4M bytes of processor storage	8 to 12 Command Facility or Cross System Product users Printer Sharing Batch jobs DTMS testing	12
8130 or 8140 (A or B) with 768K or 1M bytes of processor storage	4 or 5 Cross System Product users Printer Sharing DTMS testing	6 to 8
8130 or 8140 (A or B) with 768K or 1M bytes of processor storage	4 to 6 Command Facility users or 3 to 4 Cross System Product users Printer Sharing Batch jobs DTMS testing	6 to 8
8140C with 1M, 1.5M, or 2M bytes of processor storage	6 to 9 command Facility users or 5 to 6 Cross System Product users Printer Sharing Batch jobs DTMS testing	9 to 12
* Where the processor is either 8130 or 8140, either one can support the configuration shown. However, the 8140 generally provides faster response times and supports more terminals.		

Figure 8. Sample Application-Development Configurations

DPPX/SP-Supported Devices by Device Type

Processors and Additional Storage

- 8130 Processor
- 8140 Processor
- 8150 Processor
- 8101 Storage and Input/Output Unit

Disk and diskette storage are included in the 8130 and 8140. The 8150 provides for no internal auxiliary storage and must be used in conjunction with an 8101 Storage and Input/Output Unit. You must have at least one disk with a capacity of at least 58 megabytes.

Keyboard-Display Terminals

These keyboard displays can be used for all program development, production, operation, and administration activities.

- IBM 3101 Display Terminal Models 10,12,13,20,22,23 (character mode only)
- IBM 3104 Display Terminal Models B1,B2
- IBM 3178 Display Station Models C1 and C2
- IBM 3276 Control Unit Display Station Models 1-4,11-14
- IBM 3277 Display Station Models 1,2
- IBM 3278 Display Station Models 1-5,52
- IBM 3279 Color Display Station Models 2A,2B,3A,3B
- IBM 3290 Information Panel
- IBM 3643 Keyboard Display Models 2,3,4
- IBM 8775 Display Terminal Models 1,2,11,12

However, the 3101, 3276-1, 3276-11, 3277-1, 3278-1, and 3643 terminals Cannot be used to run the following:

- Interactive Map Definition (DPPX/SP IMD)
- Cross System Product/Application Definition for DPPX/SP
- Data Capture and Management System (DCMS) for DPPX
- Interactive Transaction Generator of DPPX Presentation Services for 3640 Terminals (PS3640)
- The Router component of DPPX/SP

These program products require a screen width of at least 80 characters, a screen depth of at least 24 lines, and at least 12 program function keys. You can, however, define maps or data under these program products that will run on the devices listed above.

Keyboard-Printer Terminals

- IBM 2741 Communications Terminal
- IBM 3232 Keyboard Printer Models 1,11
- IBM 3767 Communication Terminal

These keyboard printers can be used for all program development, production, operation, and administration activities with these exceptions: they cannot be used with DPPX/SP IMD, with the Data-Stream Compatibility component of DPPX/SP, with Cross System Product/Application Development for DPPX/SP, with Data Capture and Management System (DCMS) for DPPX, and with DPPX Presentation Services for 3640 Terminals (PS3640). Those program products require a display terminal.

Printers

- IBM 3102 Printer
- IBM 3230 Printer Models 1,2
- IBM 3262 Line Printer Models 2,3,12,13
- IBM 3268 Printer Models 1 and 2
- IBM 3283 Printer Model 52
- IBM 3284 Printer Models 1,2
- IBM 3286 Printer Models 1,2
- IBM 3287 Printer Models 1,1C,2,2C,11,12
- IBM 3288 Line Printer Model 2
- IBM 3289 Line Printer Models 1,2,3
- IBM 5210 Printer Models E1, E2 (character mode only)

These printers can be used for all printing activities except for the following: the 3102 is for hard copy of 3101 output only; the 3268, 3283, 3284, 3286, 3287, and 3288 cannot be used for output from the remote job entry component of DPPX/SP. For output containing APL characters, you may use the 3287 Printer Models 1 and 2, or the 3268 Printer.

3640 Plant Communication Terminals

- IBM 3641 Reporting Terminal
- IBM 3642 Encoder Printer
- IBM 3643 Keyboard Display
- IBM 3644 Automatic Data Unit
- IBM 3645 Printer
- IBM 3646 Scanner Control Unit
- IBM 3647 Time and Attendance Terminal

The program product that helps manage transaction flow on a 3643 or 3645 is DPPX/SP IMD. The program product that helps manage transaction flow on a 3641, 3642, 3644, 3646, or 3647 is DPPX Presentation Services for 3640 Terminals (PS3640).

Card Readers and Punches

- IBM 2502 Card Reader Model A1
- IBM 3501 Card Reader
- IBM 3521 Card Punch
- IBM 3782 Card Attachment Unit Models 1,2

Magnetic Tape Units

- IBM 8809 Magnetic Tape Unit Models 1A,1B,2,3

Modems

- IBM 3863 Modem
- IBM 3864 Modem
- IBM 3865 Modem
- IBM 3872 Modem
- IBM 3874 Modem
- IBM 3875 Modem
- IBM 3976 Modem Models 1,3 (Not available in U.S.)

Control Units and Controllers

- IBM 3274 Control Unit Model 51C,52C
- IBM 3276 Control Unit Display Station Models 1-4,11-14
- IBM 3631 Plant Communication Controller Models 1A,1B
- IBM 3632 Plant Communication Controller Models 1A,1B
- IBM 3843 Loop Control Unit

Subsystems and Workstations

- Distributed Processing Control Executive (DPCX)
- IBM 4700 Finance Communication System
- IBM 3600 Plant Communication Devices
- IBM 6580 Displaywriter
- IBM 5150 Personal Computer
- IBM 5160 Personal Computer
- IBM Series/1
- 5280 Distributed Data System

Other Devices

- Devices conforming to the Western Union Teletypewriter Exchange Services M33/35 start/stop line protocol
- Devices conforming to the 2780/3780 BSC line protocol
- IBM 3650 Programmable Store System
- IBM 3680 Programmable Store System
- IBM 3750 Switching System (Not available in U.S.)
- IBM 8100 Information System
- IBM 7426 Terminal Interface Unit
- IBM 308x series processors
- IBM 303x series processors
- IBM 4300 series processors
- IBM System/370 processor

All these devices and systems are supported by DPPX/SP.

DPPX/SP-Supported Devices in Numerical Sequence

- IBM 2502 Card Reader Model A1
- IBM 2741 Communication Terminal
- IBM 3101 Display Terminal Models 10,12,13,20,22,23 (character-mode only)
- IBM 3102 Printer
- IBM 3104 Display Terminal Models B1 and B2
- IBM 3178 Display Station Models C1 and C2
- IBM 3230 Printer Models 1 and 2
- IBM 3232 Keyboard Printer Models 1 and 11
- IBM 3262 Line Printer Models 2,3,12,13
- IBM 3268 Printer Models 1 and 2
- IBM 3274 Control Unit Model 51C,52C
- IBM 3276 Control Unit Display Station Models 1-4,11-14
- IBM 3277 Display Station Models 1,2
- IBM 3278 Display Station Models 1,2,3,4,5,52
- IBM 3279 Color Display Station Models 2A,2B,3A,3B
- IBM 3290 Information Panel
- IBM 3283 Printer Model 52
- IBM 3284 Printer Models 1,2
- IBM 3286 Printer Models 1,2
- IBM 3287 Printer Models 1,1C,2,2C,11,12
- IBM 3288 Line Printer Model 2
- IBM 3289 Line Printer Models 1,2,3
- IBM 3501 Card Reader Model 1
- IBM 3521 Card Punch
- IBM 3600 Plant Communication Devices
- IBM 3631 Plant Communication Controller Models 1A,1B
- IBM 3632 Plant Communication Controller Models 1A,1B
- IBM 3641 Reporting Terminal Models 1,2
- IBM 3642 Encoder Printer Models 1,2
- IBM 3643 Keyboard Display Models 2,3,4
- IBM 3644 Automatic Data Unit Model 1
- IBM 3645 Printer
- IBM 3646 Scanner Control Unit (with IBM Magnetic Hand Scanner (PN 4123495), IBM Magnetic Slot Reader (PN 4123500), and IBM Dual Entry Magnetic Slot Reader (PN 4123520))
- IBM 3647 Time and Attendance Terminal
- IBM 3650 Programmable Store System
- IBM 3680 Programmable Store System
- IBM 3750 Switching System
- IBM 3767 Communication Terminal
- IBM 3782 Card Attachment Unit Models 1,2
- IBM 3843 Loop Control Unit
- IBM 3863 Modem
- IBM 3864 Modem
- IBM 3865 Modem
- IBM 3872 Modem
- IBM 3874 Modem
- IBM 3875 Modem
- IBM 3976 Modem Models 1,3
- IBM 4700 Finance Communication System
- IBM 5150 Personal Computer
- IBM 5160 Personal Computer

- IBM 5210 Printer Models E01,E02 (character mode only)
- IBM 5280 Distributed Data System
- IBM 6580 Displaywriter
- IBM 7426 Terminal Interface Unit
- IBM 8100 Information System
- IBM 8775 Display Terminal Models 1,2,11,12
- IBM 8809 Magnetic Tape Unit Models 1A,1A,2,3
- IBM Series/1

Host Programming Supported

DPPX/SP supports attachment to a 308x, 303x, 4300, or System/370 host processor and supports the following host programming:

OS/VS1 and OS/VS2 (MVS)	SDLC	BSC
CICS/VS		
VTAM	X	X
ACF/VTAM	X	X
BTAM		X
TCAM	X	X
IMS/VS		
VTAM	X	X
ACF/VTAM	X	X
BTAM		X
VTAM	X	X
ACF/VTAM	X	X
BTAM		X
TCAM*	X	X
TSO		
As part of OS/VS2 (MVS)	X	
With VTAM and TSO/VTAM	X	
As part of ACF/VTAM	X	
As part of TSO/VTAM*	X	
*The user must provide system-level programming to communicate with this program.		

DOS/VS and VSE	SDLC	BSC
CICS/VS		
VTAM	X	
ACF/VTAM	X	
BTAM		X
VTAM	X	
ACF/VTAM	X	
BTAM*		X
*The user must provide system-level programming to communicate with this program.		

Displays and printers connected to DPPX/SP can communicate with these host systems:

SDLC or BSC	
VTAM applications	under DOS/VS, VSE, OS/VS1, and OS/VS2 (MVS)
TCAM applications	under OS/VS1 and OS/VS2 (MVS)
IMS/VS	under OS/VS1 and OS/VS2 (MVS)
CICS/VS	under DOS/VS, VSE, OS/VS1, and OS/VS2 (MVS)
TSO/TCAM	under OS/VS2 (MVS)
TSO/VTAM	under OS/VS2 (MVS)
VSPC	under DOS/VS, VSE, OS/VS1, and OS/VS2 (MVS)
BSC only	
BTAM applications	under DOS/VS, VSE, OS/VS1, OS/VS2 (MVS), and VM/370

These host operating systems can handle jobs submitted by DPPX/SP:

SDLC or BSC	
OS/VS1	via RES
OS/VS2 (MVS)	via JES2, JES2/NJE, and JES3
SDLC only	
DOS/VS	via POWER/VS
DOS/VSE	via VSE/POWER with RJE feature
BSC only	
VM/370	via RSCS and RSCS Networking
VM/SE	via RSCS Networking
VM/BSE	via RSCS Networking

Chapter 9. Installation, Tuning, and Administration

After you receive DPPX/SP and associated program products, putting your system into operation involves installing the products and then tuning the system for best performance. These and other administrative tasks are described below.

Installation

The installation process for DPPX/SP differs depending whether:

- You are a user new to the 8100 Information System

  or

- You are a current DPPX user installing the DPPX/SP program product

The customer is responsible for installing DPPX/SP and associated program products. Documentation supplied with the products tells how to install them.

Installation for a New User

Installation is a step-by-step process. Before installing DPPX/SP, you must install the 8100 hardware. The installation described here assumes hardware setup has been completed successfully.

For installation, you will receive either a set of diskettes or tapes. Installation involves inserting diskettes into the diskette reader (or mounting tapes) and making selections on the basic operator panel.

Installation results in data being copied from the diskette or tape to the system residence volume and in an initial program load (IPL) being done. This IPL initializes the operational system, including a display terminal for the operator. The operator works with the DPPX/SP configuration panels at this terminal.

Installation for a DPPX User Migrating to DPPX/SP

A special migration aid is available to help DPPX users migrate their 8100 system to DPPX/SP. The migration aid helps you:

- Determine the amount of extra space you need for DPPX/SP
- Expand or contract system catalogs, if necessary

Installation for DPPX users migrating to DPPX/SP is a step-by-step process. The main tasks you must perform to migrate to DPPX/SP are:

- Use the migration feature to determine if the volume has room for DPPX/SP
- Issue redefine catalog requests and invoke the redefine catalog utility
- Install the DPPX/SP package
- Install any user-written code
- Test the installation
- Replicate this installation and send copies to remote locations

There are four ways to replicate DPPX/SP. Choose the method that best fits your system needs:

- You can do a separate installation at each remote site. This approach assumes that each site has people with the necessary installation skills.
- You may decide to do a separate installation at each remote site, but let the administrator and system programmer control the migration process directly from the central site. This central site control is done through the Host Command Facility (HCF).
- If you have the Distributed Systems Executive (DSX) program product, you can use this product to help replicate DPPX/SP. DPPX/SP includes a Network Installation Option, which uses DSX to transmit all code changes needed to migrate from a DPPX site to a DPPX/SP site.
- You may choose to replicate DPPX/SP by dumping a copy of your installed central system to diskettes or tape. You can customize each remote site either before creating the dump copy or after the remote site has restored the central site DPPX/SP system.

Installation for a Release 1 User Migrating to DPPX/SP 2

DPPX/SP 2 will be distributed to current DPPX/SP 1 users as a DPPX/SP 2 Revision Package, which will allow the implementation of DPPX/SP 2 at the base site.

The customer may then distribute the DPPX/SP 2 update to remote sites.

Customizing

Customization involves determining which network devices you want as part of your system and then using special customization panels to supply this information to DPPX/SP. The customization panels prompt you for information on which devices you have and then create command lists to define a DPPX/SP system based on the information you supply.

If you later want to add or delete terminals, disks, or diskettes or add program products, a system administrator can make the required changes at any time from a terminal. This is called *reconfiguration.* The changes are entered using another set of panels. You do not need to redefine the entire network.

An administrator at the central site can send configuration information to the distributed sites. To do this, the administrator can:

- Use the Distributed Systems Executive (DSX) program product to send command lists (CLISTs) to the distributed 8100 sites. Users at these sites can use the CLISTs to configure their systems.
- Send a configured system on diskettes or tapes to the distributed site.

If you are migrating from DPPX, you can either continue using your existing definition command lists and use commands to make any necessary modifications for DPPX/SP, or you can redefine your system using the customization panels.

Tuning

The procedures that follow installation, to define and activate a system that meets specific requirements and performs well, are collectively known as *tuning*. Tuning has three stages:

1. Definition—supplying the correct information in the panels available with DPPX/SP. You can use these panels at installation time or at any time to redefine system resources. The panels let you define users and environments and customize software, hardware, and system data sets. You can customize a remote site system through the panels as well.

2. Measurement—monitoring the system's performance to determine whether it meets the objectives you had for it when you installed it. If all your objectives are met, no further work is required.

   Measuring system performance consists of:

   - Identifying which events are to be measured.
   - Determining how the measurements will be expressed. Two typical ways are response time (number of seconds to complete a request) and throughput (number of jobs processed in an hour).
   - Determining the time of day the measurements will be taken, such as during the busiest hour.
   - Making the measurements. One tool you can use to do this is the DPPX Performance Tool program product.

3. Analysis—interpreting the results of measurement to determine why the system does not meet your objectives. Analysis provides information that lets you isolate the specific areas that do not meet your objectives and decide what customization changes you need to make.

Analyzing the measurements means comparing them with your objectives. As a result, you might want to make changes in the following areas:

- The execution priorities of system and application programs.
- Which system and application programs permanently reside in processor storage.
- Parameters for communication sessions (for example, the size of communication message units).
- System parameter settings (for example, the amount of storage set aside for I/O buffers).
- The way the system's hardware is arranged (the configuration). You can use the DPPX/SP customization panels to modify your system's configuration after it is installed.

- The program products used in the system.
- The way your application programs are designed and how they work with system facilities.

You may need to repeat this process more than once until the system's performance meets your objectives.

If you have several 8100s and have tuned some systems at one location, you can transfer tuning information from that location to other 8100 locations in command lists using the Distributed Systems Executive program product.

Administration

Administering DPPX/SP actually begins when the customization choices described above are first considered. The selection of the configuration definition on the panels and the plans for further customization are administrative tasks. The topics below describe the DPPX/SP facilities for administration. The facilities are used both immediately after installation and at any time later when changes are necessary.

User Definitions

Often, you may want to define a structure so that certain people at a site have access to all system facilities, others have limited access, and possibly others have no access at all. Each person with access (called a **user**) is defined to DPPX/SP by a data area called a *profile*. You can display, define, change, or delete user profiles through the DPPX/SP panels once installation is done. (You can also use the interactive editor in DPPX/SP to edit profile data sets, but you may find the panels faster.)

Assigning a profile (including a password) to a user lets the user log on to one of a number of programs, such as those provided with DPPX/SP. The user then has access to those DPPX/SP functions for which the user is authorized. Once logged on, the user can benefit from certain other supportive environments, such as the batch queue processing environment and the Printer Sharing environment.

When a user logs on, the supplied ID and password are checked for validity. Users without valid IDs and passwords are not allowed to log on. The user profile contains other authorization data that controls the commands, data sets, printers, and programs that the user can access.

The assignment of user IDs, passwords, and other authorization level definitions can be controlled at individual 8100 locations or at a central site. The DPPX/SP panels can be used to make the necessary data changes.

Central control can be achieved by one of these methods:

- **From a host computer.** A system administrator can send user profiles from a host location to any 8100 location attached by a data link. Or, the administrator can use the installation information from one site and send it to a remote site. The data sets can originate from any of the 8100s or from the host.

- **Through a host computer**. Using the Host Command Facility program product, a user with proper authorization can log on at a host to the Host Command Facility and make any necessary changes to user profiles at any 8100 system attached to the host.
- **From an adjacent 8100**. A user with proper authorization can log on to an 8100 and send or make changes to user profiles at an adjacent 8100 connected by a data link.
- **Using diskette or tape**. User definitions can be copied to tape or diskette and can be physically sent from one 8100 to another.

Disk Management

DPPX/SP lets you define storage volumes and manage the space on these volumes. A system administrator can, for example, initialize such data areas as the volume table of contents, data set profiles, and the volume label for disks, diskettes, and tape.

When a volume has been defined and activated, the space on the volume is available for use. Control information about the data sets defined on the volume and mapping information about the space on the volume are stored in special data sets called *catalogs*. The names assigned to the system's catalogs provide a naming mechanism for user data sets.

Besides grouping data sets, the catalog structure is useful when diskettes must be exchanged between two 8100s. By using similar menus or commands at two 8100 locations, data sets can be stored on a diskette at one location and, when the diskette is inserted at the other location, the data sets can be accessed as if they were defined there.

Among other uses, a catalog defines the level of security required to access each data set.

Configuration Management

Configurations come in many sizes and involve varying levels of complexity. An 8100 Information System is characterized by the configuration of the hardware units, by the terminal devices attached to the units, by application programs, and by telecommunication lines to other processors or subsystems in the network.

When a business wishes to expand or delete from its configuration, it is not necessary to redefine the entire network. Configuration-defining commands allow for incremental changes. DPPX/SP panels also allow for configuration changes. Terminals can be added to a configuration to meet the needs of increasing numbers of terminal users. Specific commands to define the terminal and the connection to the terminal need be issued only once. Immediately after they are defined and activated, the devices can be used. Additional steps such as initial program load (IPL), system generation, and redefinition of the terminal to individual DPPX program products are not necessary.

This flexibility in the configuration extends to application programs. In particular, newly coded application programs can be installed and invoked without needing an IPL or a redefinition of the configuration.

Chapter 10. Application Program Development

DPPX/SP and associated program products allow development of application programs interactively.

For the most part, programmers log on to DPPX/SP at a terminal and use DPPX/SP dialogs or commands. This gives the programmer access to development tools—an interactive editor, an interactive programming tool, programming languages, and so on.

The first step in this process is the logon procedure.

The Terminal User Interface

To establish contact with DPPX/SP, a terminal user enters a logon command. The typical application developer logs on to DPPX/SP Command Facility, supplying a user ID and a password. The system locates the developer's *user profile*, a data area defined by the system administrator or some other authorized user. The user profile determines the operating characteristics of the interactive environment started for the user when the logon command is processed.

A user can also be automatically logged on. A session is started between the user at a terminal display and the system. When the user turns on the terminal display, a logo appears on the screen automatically. The user has passed over the usual logon procedure.

An *environment* in DPPX/SP is a collection of resources used to accomplish a unit of work. To the programmer using an interactive environment, this means that certain catalogs, data sets, devices, and areas of storage are available for use. The programmer's authority to use certain programs, printers, data sets, and commands is established.

Other users log on to or use other, more specialized environments such as installation-defined environments.

After logon, the user responds to menu prompts or by entering commands. The menu method is generally easier to use. In some cases, the system can log the user on at power-on so the user will see the menu first.

When commands are used, they are entered as one- or two-word requests. Almost all commands are accepted in either an abbreviated or a spelled-out form. For example, to define a data set, a user can enter the command DEFINE.DATASET or the abbreviated form DEFDS. Commands used in their long form are self-documenting. This is especially useful to new users and in command lists.

The DISPLAY.COMMAND command lets users interactively see the syntax of many DPPX/SP commands. A user can use it to review a specified command. DISPLAY.COMMAND also lets users issue the requested command directly from its display.

While working at a keyboard display, a user can request a printed copy of the screen image. One or more screen-image printouts provide a record of a terminal session. This ability to quickly obtain a record of work being done is convenient. The record can be carried away from the terminal, updated if necessary, and used during subsequent terminal sessions.

The PRINT key on keyboard-display terminals that have it can be used to request such a printout. For example, pressing the PRINT key on an 8775 Display terminal prints the screen image on an associated 3287 Printer. In certain configurations, a program function key on a 3277 can be assigned to invoke this screen-image copying function.

A user can access Distributed Processing Control Executive (DPCX) and its Distributed Office Support Facility (DOSF) through DPPX/SP.

Interactive Editor

An application programmer can enter source-program statements from the terminal using the DPPX/SP interactive editor.

The programmer enters a DPPX/SP command to name the source-program data set and to prepare the editor for subsequent subcommands. The programmer then begins to enter source statements, using the editor subcommands as an aid. The editor subcommands can be used from either a keyboard display or a keyboard printer. A person familiar with these subcommands needs only to relearn keyboard characteristics to be able to work from different terminals.

Editor subcommands make it easy to add, locate, move, change, copy, and number the lines of a source-program data set. The editor allows the person entering a source program to define a format for the data being entered. For example, a person could define a format to coincide with the format of DPPX COBOL source statements.

When the command to begin an editing session is issued, a backup data set can be named as an added protection against data loss.

When entering source programs, a user can keep sets of editor subcommands in *command lists* that can be called during edit sessions. Users can keep different command lists for differing format requirements. For example, one command list can be called to enter lines of code, and a different command list to define the format for comment lines. Predefined data can be inserted within the lines of a data set being edited using the INCLUDE subcommand.

During editing, the interaction between terminal and user is enhanced by editor responses to user input. Users editing line by line control the data lines they type. When the system fails to execute a subcommand or executes it with the possibility of an unexpected result, the editor issues warning or error responses.

A full-screen capability adds convenience for keyboard-display users. An additional set of subcommands is available for local line-editing tasks, including inserting, deleting, and repeating lines.

With the full-screen editor, the screen image can be "split." The user can view two separate sets of lines of the data set simultaneously. Editing can take place on either half.

The full-screen editor lets program function keys be used to represent predefined, frequently used subcommands. This can save time during text entry. In addition to character data, the interactive editor can be used to enter hexadecimal data directly.

Users can design and enter application programs at a terminal through interactions with the Cross System Product/Application Development for DPPX/SP program product. (You need the Cross System Product/Application Execution for DPPX/SP program product on a system to run Cross System Product/Application Development–developed programs.)

Programming Language Support

Once a source program is entered, the programmer at a terminal can request that the appropriate compiler or assembler translate it. The program products available to translate source programs under DPPX/SP are for COBOL, PL/I, FORTRAN, and Assembler language programs.

A sequence of commands triggers program translation. The commands allocate output and work data sets and name the options to be used. Lists of commands have been defined by IBM for general-purpose invocations. Programmers may alter these command lists as necessary. The programmer starts the sequential execution of the commands in a command list by issuing a single command from a terminal. The programmer can view the results of compilation at the terminal or direct them to a system printer.

Using any of the languages, a programmer can code an application program that can be shared among multiple users.

Linkage Editor

The DPPX COBOL, DPPX PL/I, DPPX FORTRAN, and DPPX Assembler program products let a programmer divide a large program into parts, each part containing one or more procedures. After the procedures are assembled or compiled, the programmer requests the linkage editor to combine the procedures into an executable program.

IBM has defined in command lists the DPPX/SP commands for requesting linkage-editor execution. Users can alter the command lists to specify certain linkage-editor options, such as to request an output report or to produce an overlay structure.

As part of its processing, the linkage editor creates an information data area called a *program profile*. The profile contains execution characteristics such as the program's size, the authority required to request its execution, and the program's priority with respect to other programs competing for execution time.

Debug Facilities

Once a program is free of compilation errors, execution testing can begin. Initial testing of an application program is typically done in the same interactive environment used for program definition and entry. No special environment need be set up.

A programmer can enter debug commands to control and monitor program execution. In particular, a programmer defines breakpoints to stop program

execution. When a breakpoint is reached, various conditions can be checked, and the contents of storage, registers, or system status areas can be displayed and modified. If necessary, the programmer can request a dump of user storage.

The DPPX COBOL Compiler, DPPX PL/I Compiler, and the DPPX FORTRAN Compiler program products include statements that can be inserted into a program solely for the purpose of diagnosing the program.

The DPPX PL/I Library program product contains an Interactive Execution Debug Facility that provides panels from which the user can monitor program execution and display and change program data. Symbolic variable names can be used to access data.

DPPX/SP includes the services of an execution debug monitor used with its map formatting services. Applications using the DPPX/SP IMD program product when handling input/output (I/O), perhaps from a COBOL or PL/I program under DPPX/SP, can use these debugging facilities. The debug monitor:

- Directs formatting error messages to the system operator terminal.
- Lets programmers stop execution before or after I/O events and display data and control information. It can be used as an I/O facility.

A help facility explains how to use this debug monitor.

Further Development Aids

DPPX/SP provides other aids to assist the development process:

- DPPX/SP contains data management facilities to ease data-set organization, definition, and manipulation.
- The DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product helps you develop programs that use formatted displays to communicate with an end user. For example, inventory control and other applications that service commercial transactions often involve formatted displays that convey or solicit information.

Data Set Management Facilities for Application Programmers

Two major uses of DPPX/SP data sets are:

- To store source programs created during application development
- To store the data sets processed by user applications

DPPX/SP data set management facilities are accessible by commands or through the menu interfaces to these services. From a terminal, a programmer can create data sets and data bases and change certain of their characteristics.

Control information about DPPX/SP data sets or data bases is stored in special data sets called **catalogs**. In addition, catalogs contain mapping information to control the space on direct-access storage volumes.

The names assigned to the system's catalogs provide naming conventions for the data sets. The authorized user who determines the catalog structure is responsible for assigning catalogs to a site's application programmers. Each application programmer must then follow a convention for naming data sets. This convention is determined by the name of the catalog the programmer is using.

All DPPX/SP data sets contain fixed-length records and can be accessed in three ways: directly, sequentially, and by index (in an order determined by user-defined keys or directly by key).

Display Formatting

Most interactive transactions involve communication with keyboard-display operators, who fill out fields on formatted screens. Properly formatted screen layouts can increase operators' productivity. The DPPX/SP Interactive Map Definition (IMD) program product simplifies the creation of screen layouts. The application programmer need not be familiar with the intricacies of the keyboard display used by the operator. By focusing on application program logic and presentation separately, a programmer can produce more usable applications. Both the coding and updating of the logic and presentation aspects of the application are simplified.

Normally, a programmer begins designing the layout of data on the screen before coding the corresponding application program. A prototype of each display is entered on a keyboard display using IMD.

In addition to positioning fields on a screen, the programmer also names the fields so that they can be referred to by an application program. Because the data produced by IMD maps the correspondence between names and positions, it is known as a *map*.

Data entered on the screen using IMD is held in a source form that can be edited and changed until a satisfactory layout is produced.

Once satisfied with the layouts, the programmer starts an IMD process to produce a generated application data structure and a generated version of the map.

The application data structure contains the names given to display fields. The names are held in a format that can be copied into a source application program to define an input/output area. The generated version of the map contains formatting and mapping information. This generated version is used during execution to format the screen in the required manner and to map the data passed between the application data structure and the screen.

When the map and the application data structure have been generated, writing the application program is considerably simplified. An input/output area is already generated, with names that correspond to fields on the screen.

Input/output statements are not complicated by the intrusion of complex formatting instructions. The programmer is left free to concentrate on the logic of the program and its interactions with the end user.

Invoking Application Programs

Applications vary widely in their requirements for such system resources as processing time and main storage and in their degree of interaction with the end user. The different processing modes of DPPX/SP are designed to use the system resources for the various types of applications most efficiently.

The sections that follow describe how DPPX/SP uses various processing modes for application program execution.

Batch Execution

Certain applications require data to be collected (or batched) over a period of time and processed at intervals. Such application programs can be submitted for batch execution (see Figure 9). A monitoring program within DPPX/SP accepts batch requests and schedules their execution. Up to sixteen batch requests can be processed concurrently with DPPX/SP 2.

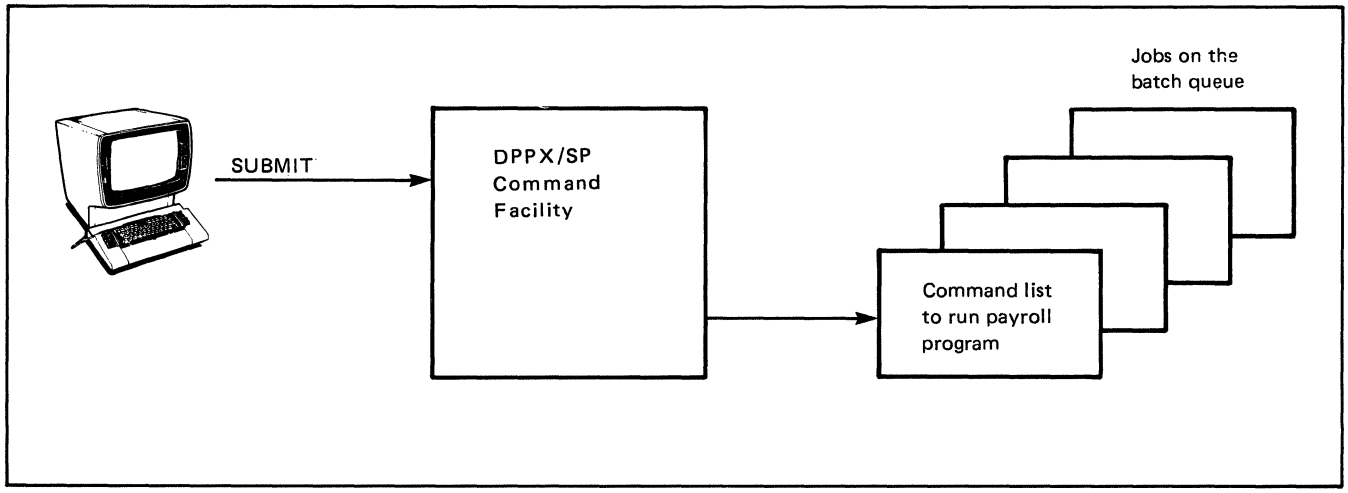


Figure 9. Payroll Program Submitted as Batched Job

Command Facility Execution

People wishing to have an application program execute immediately can log on to the DPPX/SP Command Facility and choose among a number of methods of execution.

One choice involves the CALL.PROGRAM command. This command requests program execution and passes parameters to the program to be executed. Normally, CALL.PROGRAM is one of a number of commands in a command list. Other commands that must be issued to prepare for the program's execution are included.

A command list can contain special commands that make the same list applicable in many situations. When the command list is submitted for execution, values can be specified for insertion into the list.

For example, a command list used to copy data set UCAT.A into data set UCAT.B can be made more general by defining UCAT.A and UCAT.B as default names. The option to substitute other data set names is then open.

A command list can execute commands conditionally. Making a command's execution conditional is done by preceding it with a check of the code returned from the preceding command. Whether a command list is submitted for immediate or for delayed (batch) execution, all command list capability is available.

If the application program is to execute often, a user can define a command for requesting program execution. DPPX/SP provides a number of easy and varying options for extending the DPPX/SP-defined command set. Such installation-defined commands provide a simplified method for users to request application programs to execute.

Figure 10 is an illustration of a simple application dealing with order entry. A terminal user enters an installation-defined command named ORDER, followed by variable data to be used by ORDERA. This data is passed to ORDERA as a parameter. The ORDERA program can return an acknowledgment message to the terminal when it completes. Before the message is returned, a conversation between the terminal user and the application program can take place.

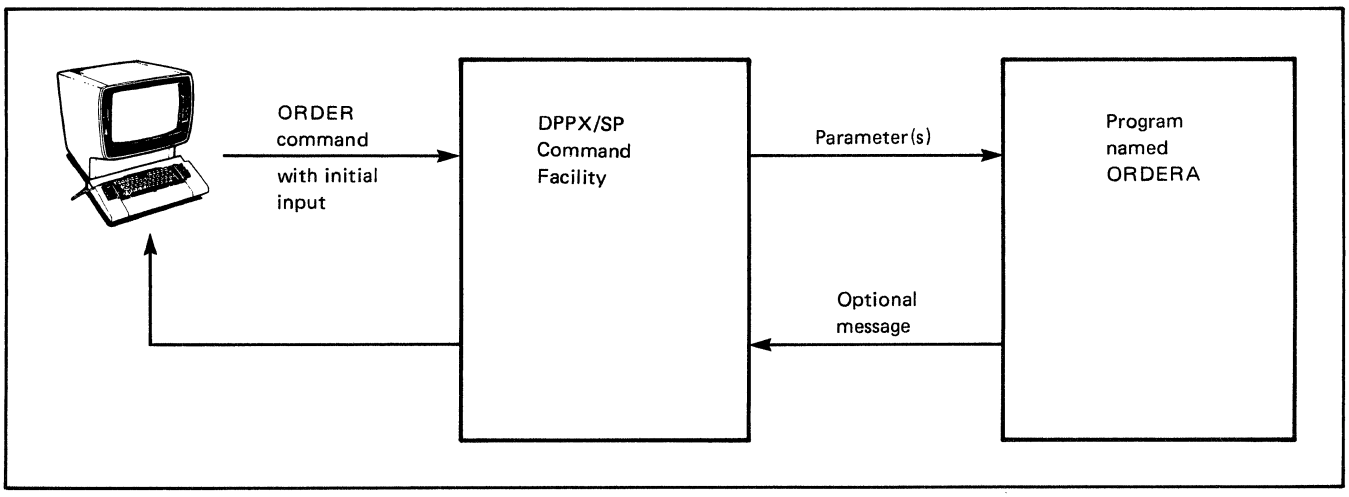


Figure 10. An Order-Entry Program

Data may be entered in full-screen mode or line by line. The DPPX/SP IMD program product supports the full-screen mode, letting preformatted screens be generated and produced by application programs.

Additional Support for Transaction Applications

Certain application programs can benefit from two special aspects of DPPX/SP: its handling of transactions and its management of data.

DPPX/SP matches terminal users and their transactions. Application programs can be designed as if a single transaction were taking place. Dynamically scheduling the many transactions that actually take place is controlled by DPPX/SP services.

The order-entry program shown in Figure 10 could as well have been coded as a DPPX/SP transaction. The transaction could then benefit from DPPX/SP services, which include:

- The ability to remove all data changes from any transaction that is canceled or, for some other reason, does not complete normally
- A greater degree of control over multiple users updating the same records
- Commands to re-create data that has become unusable

The following examples of application programs written to handle order-entry transactions show two ways of designing programs to take advantage of this DPPX/SP support.

Figure 11 shows a program named ORDERB accepting a customer name and address as initial input and, following a conversation between the terminal operator and the application program, sending the operator an end-of-transaction indication.

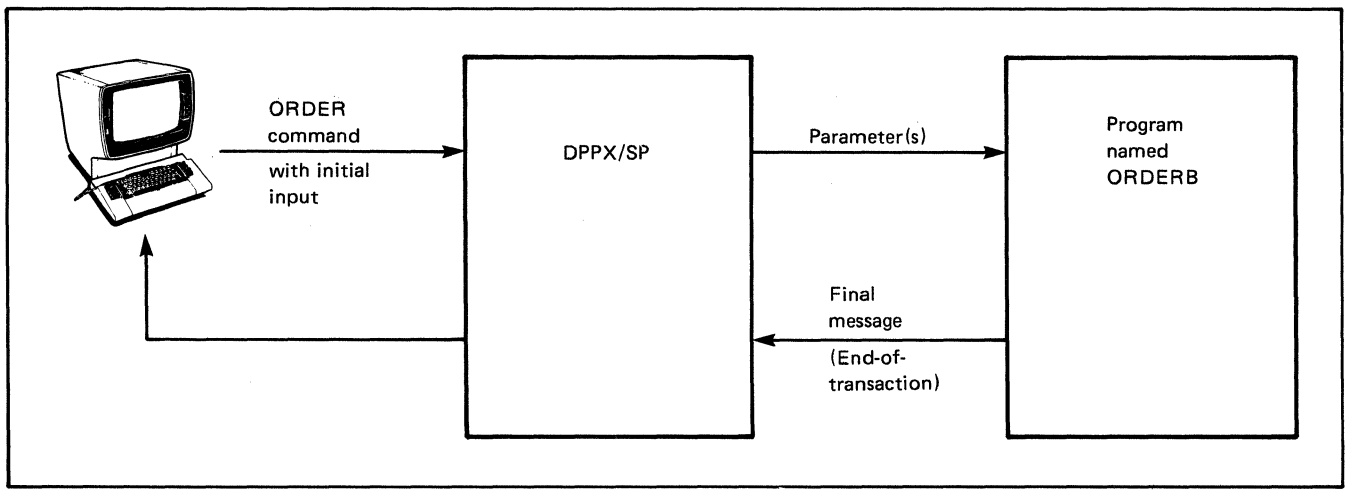


Figure 11. Order Entry Handled by One Transaction Program

There are trade-offs involved with implementing such a "conversational" program. From the moment of initial data input until the remaining data is entered and processed, storage and other resources supporting the transaction are unavailable for other uses.

The application is easy to write and maintain. Also, since the resources that support the transaction are available throughout the transaction, the time that would have been spent releasing and then reallocating resources is saved.

In Figure 12 a programmer has coded three programs that support an order-entry transaction. The three programs correspond to three items of information to be entered at separate times by a customer.

Between the execution of programs ORDERX, ORDERY, and ORDERZ, DPPX/SP lets other programs use certain resources. Yet it retains enough information to continue the transaction with the next program. When the final message from program ORDERX prompts the user for the information that becomes input to program ORDERY, the time that the user spends thinking about and entering a response is not spent tying up system resources. Other transactions are using computer resources during that time.

When the input to program ORDERY is entered, DPPX/SP makes sure the resources necessary to support the transaction are again made available. If, during the processing of any of the programs, the customer decides to cancel the order, DPPX/SP can reset all data changes that have been made.

DPPX/SP also ensures that programs supporting separate transactions do not make simultaneous (and possibly conflicting) updates or checks of the same data.

Application Program Independence from Resource Characteristics

Application programs use symbolic names to refer to the data sets, devices, and application programs that they access. Before an application program executes, an association must be made between the symbolic names used in the program and the actual data set, device, or application program being accessed. The ASSOCIATE commands accomplish this.

When an ASSOCIATE is issued before an application program executes, DPPX/SP associates the symbolic name with the data set, device, or application. The DPPX/SP ASSOCIATE commands can be compared with DD statements in certain host operating systems (OS/VS1 and MVS). A DD statement relates a symbolic name to an actual data set on auxiliary storage. In DPPX/SP, the ASSOCIATE commands relate a symbolic name to an actual data set on auxiliary (disk, diskette, or tape) storage, and to devices and other programs as well.

When an application program requests a connection to a resource, a symbolic name identifies the resource. In many instances, programs can remain unaware of the nature of the actual resource to which they are connected because of the way DPPX/SP implements connections and associations. Of course, programs requiring a particular type of resource may be specific.

But for actions described below, application programs can be independent of the resource type. A single program can, for example, operate on either terminal or data set input. This ability is particularly useful during the testing of a program. A program developed to accept data from a terminal can, during an interactive debug session, accept "simulated" terminal input from a data set. The change is handled outside the program.

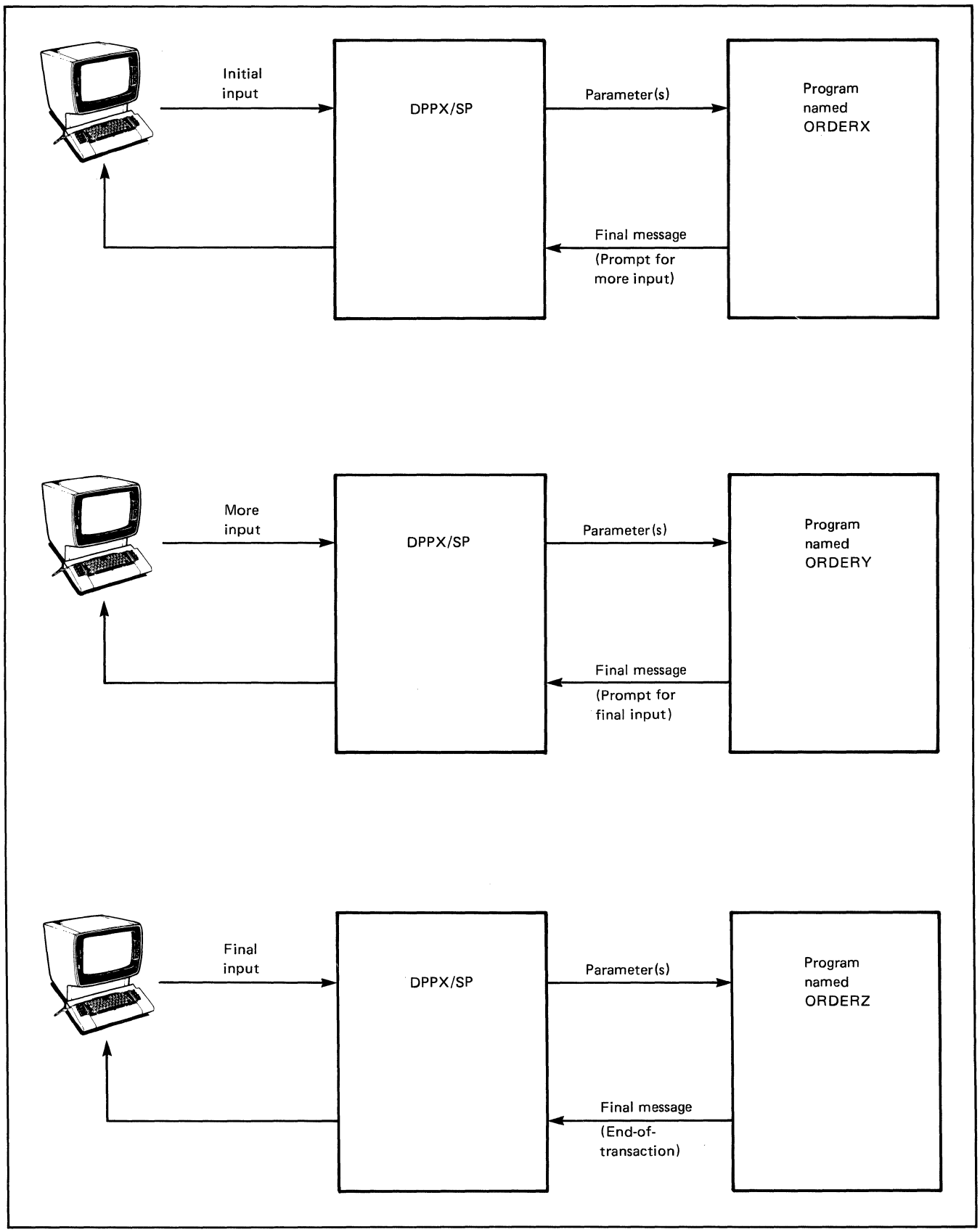


Figure 12. Order Entry by Multiple Programs

These three programming actions typify the input and output choices available to an application programmer:

- Sequential input of data: For data read sequentially, the source can change among card reader, terminal, and data sets (relative sequential data sets and indexed data sets). When a relative sequential data set is read sequentially, the source can change among disk, diskette, and tape. The same is true of indexed data sets from disk or diskette.

- Sequential output of data: For output data written sequentially, the destination can change among card punches, printers, keyboard printers, keyboard displays, and data sets (relative sequential data sets and indexed data sets). When a relative sequential data set is written sequentially, the destination can change among disk, diskette, and tape. When an indexed data set is written sequentially, the destination can change among disk and diskette.

- Interactive transfer of data: An application program can send data to and receive data from a keyboard printer, keyboard display, or certain other application programs. The interaction can be tested by using input and output relative sequential data sets stored on disk or diskette.

For each of the above actions, the selected source or destination of program data can be determined by an ASSOCIATE command issued before the program executes. No change to the application program is required.

Printer Sharing

Programs that write directly to a printer depend on the printer's availability. To prevent the bottleneck that could occur if many application programs contend for the same printer, a Printer Sharing facility permits print requests to be issued independently of the actual printing.

When an application program generates data to be printed, the data can be stored on a disk. The disk data set can then be scheduled for later printing. An authorized user can then issue a command to begin processing the print requests, with the option of specifying classes to be printed.

The system administrator or an authorized user can control which users use which printers. The administrator can divide printers and printer queues into resource groups and then decide which users access which group.

Chapter 11. Problem Handling

Should a problem occur during the operation of DPPX/SP, there are ways to lessen the effect of the problem, find its source, and apply corrective code. The following topics describe how problems are handled.

Lessening a Problem's Effect

DPPX/SP gives each user an isolated subset of system resources called an *environment*. Because of this, a problem that causes one user's failure does *not* affect other users operating with different resources. The resources associated with a failing user can be made available for new uses.

Programs executing under DPPX/SP can have associated with them a special recovery program called a *specified process check exit (SPCE)* program. IBM defines one SPCE program for general usage. For each application, a programmer has the options of using the IBM-defined SPCE program for recovery or coding a specialized SPCE program.

When an error occurs during user processing, the SPCE program for the failing program tries to recover from the error. If it cannot, the failing user environment is terminated. Resources in the environment are returned to DPPX/SP. For system failures, an authorized SPCE program can contain an IPL command to reinitialize the system. The network can be reestablished without system operator intervention.

A save/restore IPL combination can save you time if the system must be reinitialized. The operator performs a save IPL. When another IPL is necessary, the operator performs a restore IPL. The system is automatically restored to the point it was at when the save IPL was issued. This makes the IPL run faster.

DPPX/SP data base recovery ability also includes these abilities:

- Data changes that a user makes are recorded and can be removed up until the changes have been accepted by the user program and recorded.
- An audit file of data changes is kept and can be used for recovery if data becomes unreadable.

Problem Determination Aids

Error Codes

Certain conventions and abilities to provide diagnostic information when necessary are built into DPPX/SP processing. The basis for much of DPPX/SP's problem determination ability lies in the module-level convention of generating, checking, and passing return codes to define the success or failure of an operation.

A part of every error code identifies the module that generated the code. If DPPX/SP cannot resolve the error, the code appears in a message that explains the error. This identification provides a good starting point to begin looking at (and perhaps immediately solving) the problem.

Error Recovery

If a process check or machine check occurs during system operation, there is a sequence of recovery events. An error recovery manager controls these events. The recovery events include the invocation of a specified process check exit (SPCE) program. Its dual purposes are:

- Recovering from the error
- Recording data about the error

Error Log

The SPCE program can request that an error log manager write error data to an error log. The error log can be interactively displayed at a terminal or printed at a terminal or printer. The error log may also be summarized in a report. This can be done from the 8100 system or from a host computer with the aid of the Host Command Facility program product. In addition, the report may be saved in a history file on DPPX/SP disk storage. The history files can then be analyzed during problem determination to detect trends or to track individual problems.

Program-Controlled Dump

The SPCE program also lets the user request that a program dump be written to a dump data set. The user can display or print selected parts of the dump. The display or print can be formatted or unformatted.

Stand-Alone Dump

If DPPX/SP becomes disabled, a user can use a stand-alone dump program, available with DPPX/SP, to produce data necessary for diagnosis. A user loads the dump program from a prepared diskette and executes it. The resulting program dump of main storage is written to the dump data set on the diskette from which the program was loaded. The user will be prompted to insert another diskette if required. The dump data set can be printed after a re-IPL of DPPX/SP.

Dump data sets produced by the stand-alone dump program can be formatted, and the formatted dump can be displayed or printed. In addition, portions of the dump data sets (registers, control blocks, and parts of the address space are examples) can be selected or left out as needed.

System Trace

A user can obtain data for diagnosing system problems without taking a dump. An authorized user can invoke a *system trace* to record diagnostic data during I/O operations, program exceptions, system checks, and/or system-assisted program linkages.

Usual system activities—interactive and batch operations—can continue while the trace data is being recorded, although there will be some effect on performance. When desired, the user stops the trace and prints the recorded data. The traced records appear in the chronological order in which they were recorded.

Environment Tests

DPPX/SP also provides distributed system environment tests, which are started by DPPX/SP commands. These tests support problem determination by testing the system hardware and programs in a user environment. If errors occur as a result of running these short tests, the problem appears as if it were discovered by an application program. Other tools may be used to isolate the specific cause of the failure.

Another DPPX/SP command verifies that data can be transmitted between a DPPX/SP command-processing program and a device connected to the 8100. When a user issues the command, an IBM- or user-defined pattern is sent from the terminal to the program and back. The user controls the number of times that this test is done.

Execution Debug Monitor

DPPX/SP includes an *execution debug monitor*. This lets users intercept requests sent to a terminal or printer and display data and control information about that request. The following requests can be monitored: send, receive, connect, disconnect, or user exit requests.

Host Alert of 8100 Errors

When 8100s are connected to a host with an SDLC link, DPPX/SP uses the problem determination aid to notify the network operator at the host about 8100 hardware and DPPX/SP software errors.

Notice of these errors is routed automatically to the Network Problem Determination Application (NPDA) program product in the host as the errors occur. NPDA records the errors and can display them.

Link Problem Determination

The link problem determination application of DPPX/SP lets an operator test 386x modems downstream of the 8100. The operator can test a modem by issuing one of three test commands: TEST.LINK_MODEM, TEST.REMOTE_MODEM, and TEST.INTERFACE_MODEM. Results of the test are displayed at the operator's terminal and are also sent to the error log. The operator must be authorized to issue these commands.

If certain types of errors occur, DPPX/SP will automatically issue the appropriate command to check the modems and put the test results into the error log. The operator can display or print out a copy of the test results from the error log. The operator can also request that the test results be summarized in an error log report. The automatic test results are analyzed for conditions that must cause an alert.

Problem Determination by a Specialist at a Host

Problem determination can involve only the individual terminal user. Appropriate product publications contain problem-determination procedures for their audiences. If users follow the procedures described in their manuals, they can handle many of the procedural problems without reporting them to problem-determination specialists. When a specialist does become involved, the error log and dumps taken as a result of the problem become valuable sources of diagnostic information.

A specialist at a central site can gain access to such information generated at an 8100 site. The Host Command Facility program product installed in a host allows a person to log on at certain keyboard displays and use any of the DPPX/SP commands that help with problem determination. The specialist can display and analyze the error log and dumps that are at remote 8100 locations.

With the *logical connection verification (LCV)* part of Host Command Facility, a specialist can make sure that data is being transmitted correctly over the data link connecting an 8100 and a host. If a default or user-specified pattern is successfully transmitted over the link, the link can be used for communication between the host program and the DPPX/SP program.

IBM Assistance

Customers who have problems they believe are caused by defects in DPPX/SP or associated program products can contact IBM for help. The first contact point is the IBM Support Center. The Support Center is the interface to the various programming support groups in IBM and is available to basic license users.

Other DPPX/SP manuals contain information about how to diagnose and report system problems. For previously reported problems, the manuals describe how to apply a temporary fix to solve the problem. For previously unreported problems, the manuals describe how to submit an *Authorized Program Analysis Report (APAR).*

Code Updating

Program temporary fixes (PTFs) developed by IBM will be made available to basic licensed users. A PTF contains code that corrects defects in a program product.

When fixes are made, they are distributed on diskettes in groups called *fix packages*. A command transfers the necessary fixes from the diskette on which they are shipped to disk storage controlled by a special system catalog. During this transfer, day-to-day operations continue as before.

DPPX/SP is serviced on a single schedule. Since it combines several products' functions, users only have to handle *one* set of fix package diskettes.

During subsequent initial program loads (IPLs), the operator can choose to have application programs execute against the new level of the system. If application programs execute properly with the new system code, the fixes can be made a permanent part of the system. Otherwise, the fixes can be removed.

Occasionally, the latest level of a program product's code, in the form of a *service level update (SLU)*, will be made available. An SLU incorporates all available fixes for a given program product. Customers who wish to keep DPPX/SP current should apply all fixes as they become available. This eliminates the need to install new service levels. This is the preferred method of remaining current. The alternate method is to install new service levels.

Either method can be used to update DPPX/SP to the same service level.

As mentioned, fixes can be applied at the DPPX/SP location. An alternate approach is to make the change at one DPPX/SP location, test the change there, and distribute the tested code to other locations, using the Distributed Systems Executive program product from the host.

Advantages of centrally controlling code updates include:

- Fixes can be applied concurrently or staged throughout the network.
- Diskette handling is reduced.
- A record of the level of installed code at each DPPX/SP location is kept at the central location.

Chapter 12. Network Management

Using DPPX/SP can involve distribution of computing power and data throughout a geographically dispersed network. With this distribution there comes a concern about managing such a widely scattered system.

Experience with large networks of terminals has shown the effectiveness of systems management from a single point of control. With a network of distributed processors:

- You can apply the same concept of central control.
- It is possible to operate the network without highly skilled data processing personnel at each distributed processing site.

Overview of Network Management Programs

At the control of an IBM network is the network operator using a display terminal to access the Network Communications Control Facility (NCCF) program product (see Figure 13). DPPX/SP and the program products described in this chapter provide, through the window of this NCCF terminal, complete and timely information on the status of 8100 distributed processors in the network.

The following program products operate in the host system:

- Host Command Facility (HCF) Version 2
- Network Communications Control Facility (NCCF) Release 2, with the Terminal Access Facility feature
- Network Problem Determination Application (NPDA) Version 2
- Distributed Systems Executive (DSX) Version 2

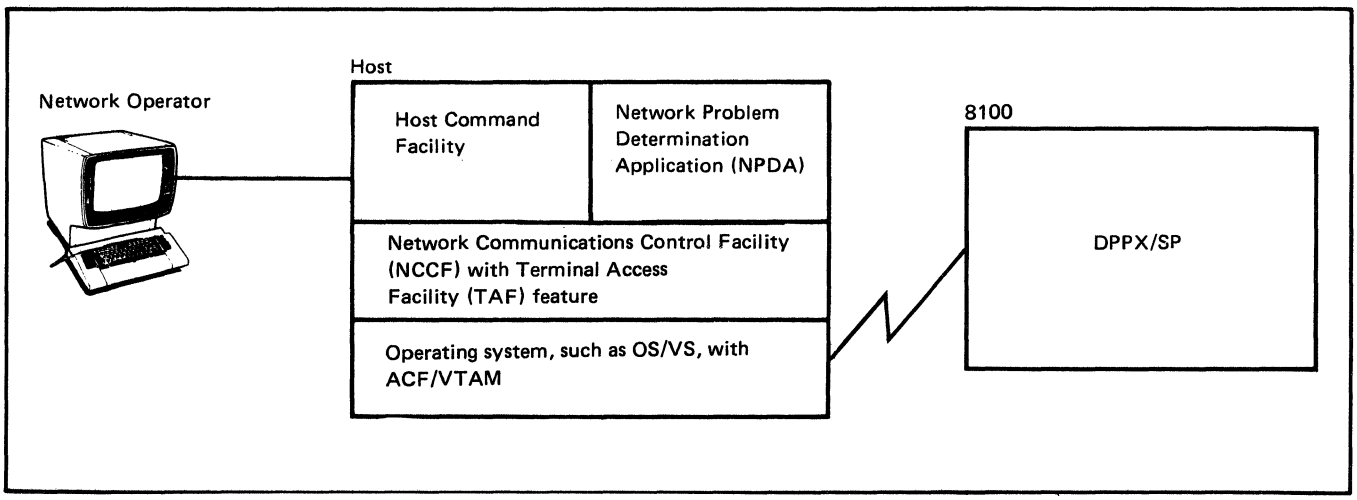


Figure 13. Program Products for Managing an 8100-DPPX/SP Network

DPPX/SP (in the 8100) provides these services:

- Provides for a *programmed operator*, performing many system operator functions that do not require physical activity
- Alerts the host of many 8100 problems
- Allows operators at host terminals to log on to remote DPPX/SP systems

The network operator manages the network through the use of NCCF. In a multisystem network, the operator might share that management with other NCCF operators in the same or other host systems. Each may have a defined span of control, which limits that operator's control to a subset of the network's resources.

The network operator has direct access through NCCF to the problem determination facilities of Network Problem Determination Application (NPDA). In addition, the Terminal Access Facility feature of NCCF offers concurrent access to the Host Command Facility (HCF), Customer Information Control System/Virtual Storage (CICS/VS), and Information Management System/Virtual Storage (IMS/VS) program products. DPPX/SP gathers information on 8100 system operation and routes it to the host if certain errors occur or if information is requested.

Through DPPX/SP's services, the network operator can effectively manage the network, detect problems, and supply information to technical support personnel doing problem determination.

The following sections discuss how each of the DPPX/SP services or associated program products contributes to the network operator's ability to act as a single point of control. We will start with the DPPX/SP services in the 8100.

Network Management DPPX/SP Services in the 8100

Starting DPPX/SP

The most basic task in operating DPPX/SP is *initial program load (IPL)*. To start DPPX/SP, the operator can press the IPL/RESET key on the operator panel. To restart DPPX/SP after it is running, an authorized user can also issue an IPL command.

Another form of IPL, *save/restore IPL*, can save you time on subsequent IPLs. An operator performs a save IPL. If the operator has to reIPL the system, a restore IPL is used. This system is restored to the state it was in when the save IPL was issued.

After IPL, the operator can start environments. These environments identify the system resources available to users. For example, one user can log on to an interactive environment and use resources that are independent of those available to another terminal user.

The different types of DPPX/SP environments give an installation a large amount of control. Environments that can be started include:

- A Printer Sharing environment, for controlling operation of the system's printers and punches.
- A batch environment, for running jobs that users have submitted for delayed execution
- An environment for running transaction programs

The DPPX/SP Programmed Operator

DPPX/SP provides for a "programmed operator." In this way, DPPX/SP minimizes the amount of work required by a human operator at the 8100 site. Using this DPPX/SP service requires planning and tailoring on your part, but it also offers some significant benefits.

First, it offers a method for standardization of message and return code handling. (A *return code* is an operator response that provides information about the failure of some requested function.) This standardization can eliminate many of the inconsistencies normally involved in managing a network of distributed processors.

Second, it lets each installation establish its own criteria for determining which messages require routing to a human operator and whether that operator should be at the 8100 or at the host location.

Finally, this DPPX/SP service lets users define a set of actions to be performed at specific times of the day or at the end of specific time intervals.

IBM supplies default responses for all messages and return codes. Users can change these responses to meet their requirements. The responses are kept as action statements in a disk data set. DPPX/SP references them upon the arrival of messages, return codes, and timer-driven events.

Each message or return code received by DPPX/SP is logged in the system operator session log data set. A search is then performed against the action statement data set to locate the appropriate user-defined action, and the action is performed.

Six actions are possible:

- A reply, consisting of user-provided text, can be returned to DPPX/SP.
- The message or return code can be ignored if logging it was sufficient.
- A predefined command list can be started.
- A program can be started.
- The original message or return code text, or user-provided substitute text, can be routed to one or more terminal users. Routing can be to a terminal user at the 8100, a terminal user at the host, or to all users logged on to DPPX/SP.

- Text can be routed to an 8100 terminal user or to a host terminal user with a reply requested. An example might be the request of a GO reply when forms mounting on an 8100-attached printer has been completed. The user's reply is returned to the programmed operator service of DPPX/SP, and it will reply to the system if necessary.

Message action statements define not only the type of action to be performed but also whether the action applies to a specific message or to a message category.

Categories can be based on message type—information, action, or problem determination. For example, all problem determination messages, no matter what the source, could be routed to the network operator.

Categories can also be based on message type plus DPPX/SP facility identification. Some examples: all information messages from the Printer Sharing component could be routed to a terminal operator at the host; all Printer Sharing messages that require operator action could be routed to the 8100 operator.

A general default action can be specified for all messages not otherwise defined by action statements.

Action statements may also be defined for categories of return codes and timer-driven events. DPPX/SP has several sets of commands available for program operator definition. Some commands let the user define, alter, and display both action statements and timer events. They can be executed either singly or as part of a predefined command list. They can be used at any 8100 that has DPPX/SP installed. They will most commonly be used at an 8100 application development site where action-statement data sets and timer-event data sets are being developed for distribution to other 8100s.

A monitor command allows an authorized terminal user to view all messages and return codes sent to DPPX/SP as well as the commands that DPPX/SP issues. This monitor function could be used for validating work done in developing action statement data sets or for observing the action of the DPPX/SP programmed operator service at a specific 8100 system.

Change commands are available to request rerouting of DPPX/SP output. Either an operator at the host system or an operator at the affected 8100 may issue these commands. The availability of this set of commands makes tailoring of DPPX/SP both interactive and dynamic.

Problem Determination Aid

DPPX/SP aids in problem determination. It alerts the network operator (at the host site) of significant hardware or software events so that problem determination and corrective action can begin immediately. No user tailoring or 8100 operator action is required to use these DPPX/SP services.

Three things are needed to route alert messages to the host from an 8100 and to process the alerts at the host:

- DPPX/SP must be installed and activated in the 8100.

- The 8100 must be linked either directly or indirectly to the host system via Synchronous Data Link Communication (SDLC) facilities. SDLC linkage is required because DPPX/SP shares the link with other network transmissions.
- The Network Problem Determination Application (NPDA) program product must be installed in the host to process the alerts.

DPPX/SP receives a copy of all events recorded by the DPPX/SP error log manager. It then determines whether an alert should be generated (for routing to a host) based on the following criteria:

- Is the event an unrecoverable device or adapter error?
- Is the event the failure of an IBM program product or a user-written application program?
- Is the event the result of an alert command issued by a user application program?

DPPX/SP also logs alerts passed through the network from other 8100 systems or from downstream alert-generating devices or systems. Alerts are routed immediately to the host and are recorded there by NPDA. The operator at the host can display the alerts.

Although we might tend to think of all alerts as problem warnings, the ability of a user-written application program to issue alert commands offers other uses. For example, when the last input transaction of the day is received by an application program, that program could send an alert to the network operator saying to schedule a Distributed Systems Executive session. That session could be used to transfer the transaction data from the 8100 to the host.

Network Management Programs in the Host

So far we have reviewed the techniques used in DPPX/SP at the 8100 for routing important information to the host. Now let us see what network management programs are available at the host to review and act on this information.

Terminal Access Facility Feature of Network Communications Control Facility (NCCF)

An important host element is the Terminal Access Facility, a feature of the Network Communications Control Facility (NCCF) Release 2 program product. The Terminal Access Facility is a set of command processors that allow the network operator to establish sessions with CICS/VS, IMS/VS, and Host Command Facility (HCF), and to communicate concurrently through these sessions to host applications and to multiple 8100 systems.

CICS/VS, IMS/VS, and HCF may be in the same processor as the Terminal Access Facility or may be in a different processor connected through SNA networking facilities. In fact, the Terminal Access Facility can communicate with multiple copies of each of these systems to multiple processors.

The network operator can interleave input to these various systems using Terminal Access Facility commands. Output from the systems is also interleaved on the network operator's terminal. An identification code is displayed with each message to define its source.

Figure 14 presents a simple network of 8100 systems to show the use of multiple sessions between the Terminal Access Facility and HCF. The Terminal Access Facility operator has started four sessions with HCF. Assume that the three lower sessions are all keyboard-printer sessions.

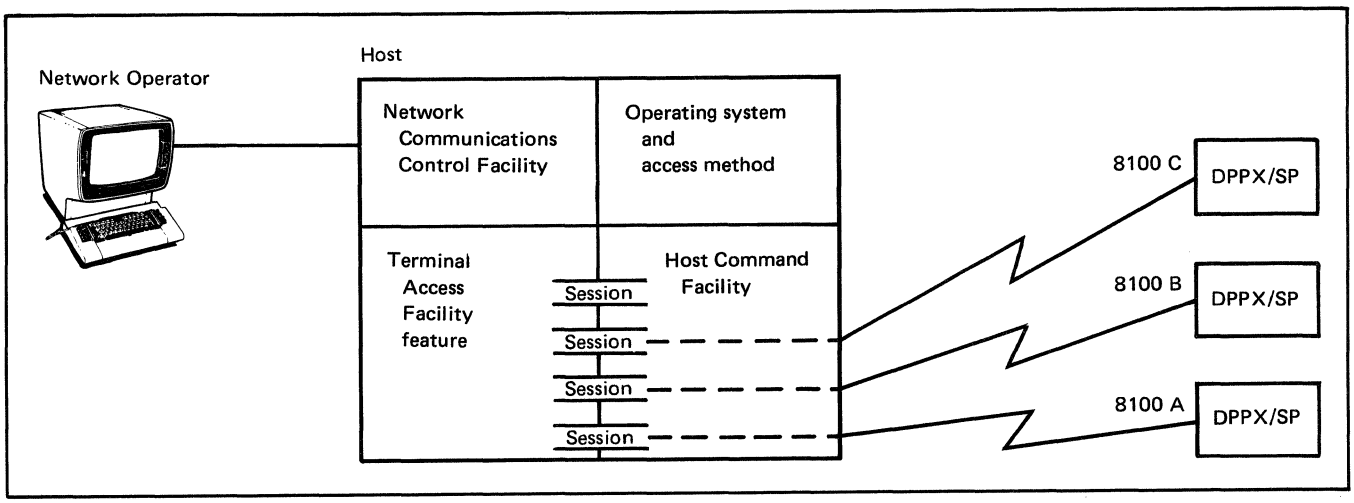


Figure 14. Network Operator Access to Multiple 8100 Systems

Messages and return codes routed over these sessions by DPPX/SP will appear as interleaved output on the network operator's terminal. The network operator, in turn, can send commands to any of the 8100s at any time over these same sessions.

The top session in the figure is assumed to be a full-screen session between the Terminal Access Facility and HCF. Whenever a specific 8100 requires full-screen network operator services, this session can be used.

Figure 15 is a representative sequence of events occurring at the network operator terminal of our simple 8100 network.

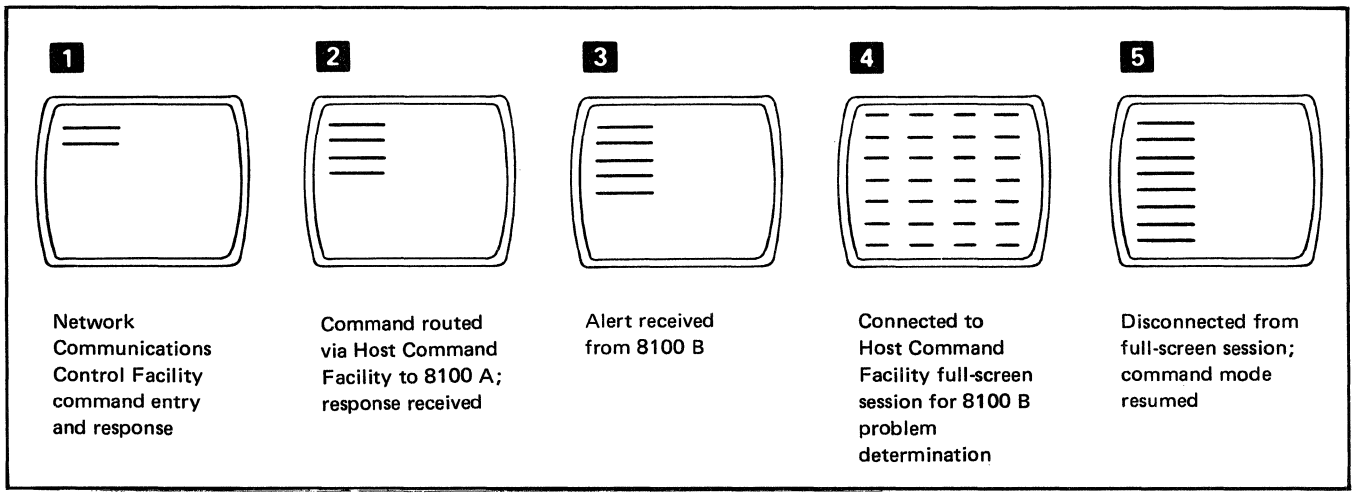


Figure 15. Example of Events at the Network Operator Terminal

Screen number 1 shows the operator using the NCCF operator control format. The operator has entered a network command, such as a request for status display of a data link, and has received a response.

Later, as shown on screen number 2, the operator has routed a command via HCF to 8100 A and received a response. This command might, for example, be a VERIFY command to check the logical connection between the host and 8100 A.

Still later, as shown on screen number 3, an alert arrives from 8100 B. Assume that the alert indicates the failure of an adapter at 8100 B. If the network operator wishes to concentrate temporarily on further problem determination for 8100 B, the operator can connect to the full-screen HCF session. The operator can then issue an ACQUIRE command to HCF for 8100 B and begin to use the DPPX/SP error log analysis facility at 8100 B, as shown on screen number 4.

At any time, the network operator can disconnect from the full-screen session and return to normal NCCF operator control, as on screen number 5. While in this mode the operator will continue to receive input and issue commands on the keyboard-printer sessions with each of the 8100s. If desired, the operator can reconnect to the full-screen session for further diagnostic activity with 8100 B. That session will remain active with 8100 B until the network operator issues a DROP command to HCF.

This brief example shows the importance of the Terminal Access Facility feature to the implementation of a single point of control.

Host Command Facility (HCF)

The Host Command Facility (HCF) program product lets the user of a host-attached display terminal log on to an 8100 system and to perform most of the functions that could be performed by a terminal operator at the 8100. These functions can include:

- Acting as the 8100 operator
- Performing 8100 administrative and control functions such as modifying catalogs and adding or deleting user profiles
- Performing problem determination functions such as displaying a program dump or tracing program execution

When the HCF program product is used with the Terminal Access Facility feature of NCCF, it enhances the value of HCF to DPPX/SP networks in two important ways:

- It allows the monitoring of multiple 8100 systems from a single terminal.
- It allows that terminal to be the network operator's terminal.

That single operator now has access to the combined facilities of the NCCF and Network Problem Determination Application (NPDA) program products in the host and all DPPX/SP problem determination functions in the 8100.

Network Problem Determination Application (NPDA)

The Network Problem Determination Application (NPDA) program product provides for the collection, recording, display, and analysis of problem-related data retrieved from a number of network devices. These include the 3705, telecommunication lines, 386x modems, and 3270 control units.

NPDA runs in the host computer, not in the 8100, and it accepts alerts sent from 8100 (and 3600) systems and routes them to the network operator. These alerts are also recorded in the NPDA data base for later reference.

NPDA also offers a filtering function controlled dynamically by commands entered by the network operator. Through the use of these commands, the network operator can define, over various time periods, the categories of alerts NPDA should record or display.

Filtering of alerts may be based on any of these criteria:

- The name of the specific resource (such as an 8100 system) sending the alert
- The type of resource (such as all 8100s) sending the alert
- The particular type of alert (such as alerts generated by commands issued within 8100 application programs)
- The time interval during which the alerts are received

This filtering capability can help the network operator to concentrate on a particular category of network problems or on problem isolation for one particular network resource, such as a selected 8100 system.

NPDA also provides an export capability that can be used when the Information/Management feature of the Information/System program product (5735-OZS) is active in a host. This export capability offers an automated method of initiating problem records when alerts are received by NPDA. Adding information to these problem records and tracking problem status can be done under the control of Information/Management.

Distributed Systems Executive (DSX)

The Distributed Systems Executive (DSX) Version 2 program product allows a user at a host-connected terminal to retrieve data sets from 8100s in a network and store, manage, and distribute them wherever they are needed in the network.

Using DSX, a central-site operator can send DPPX/SP programs, maps, command lists, messages, and other data sets to each 8100 in the distributed system. The central-site operator can also retrieve these items from an 8100 and put them into central libraries in the host.

Using DSX, a customer can assign application program development to one or more DPPX/SP locations and control updates or new releases of the program.

Chapter 13. Communication

A company's data processing resources include people, hardware, and programs. These resources can be distributed across different geographic locations. Choice of location is influenced by many factors. Some factors may be the scope of the applications the enterprise is considering, the size and organization of the enterprise, and economics.

The implementation of a data processing organization can be simple, involving only one or more stand-alone processors used to develop and run application programs. The implementation can also be more diverse, involving network communication that supports application program development and increases the scope of application program execution.

Resource Distribution and Methods

By describing the resources that can be distributed, this section shows how DPPX/SP supports communication in both simple and diverse data processing organizations.

Application-to-Application Communication

Application programs that communicate data between 8100 systems use statements that cause DPPX/SP to handle communication facilities internally. READ and WRITE statements in DPPX COBOL programs, GET and PUT statements in DPPX PL/I programs, the VREAD and VSET functions in DPPX APL programs, and SEND and RECEIVE macros in DPPX Assembler language programs can be used to transfer logical records from one application program to another.

Figure 16 shows an example of program distribution. In the example, an enterprise has inventories at two warehouse locations: A and B. An application program at each location could handle order entry. The programs could be designed to check their local inventory data for an ordered part. If the part is not found locally, the programs check if the part is in the other warehouse. Appropriate data updates are made when parts are found, and shipping instructions are generated if necessary.

Application programs can communicate with host Customer Information Control System/Virtual Storage (CICS/VS) and Information Management System/Virtual Storage (IMS/VS) application programs. DPPX/SP handles transmission protocols. Application programs written in DPPX COBOL, DPPX PL/I, DPPX Assembler language and using DPPX/SP services can use a CALL statement to request that host transactions take place. The host transactions could, for example, update a host data base. DPPX/SP provides transmission integrity for host transactions.

Figure 17 shows one of many possible applications that can benefit from this method of communication between DPPX/SP and host application programs. This inventory example involves an enterprise with a number of warehouses throughout a country, and a factory and warehouse in a headquarters location.

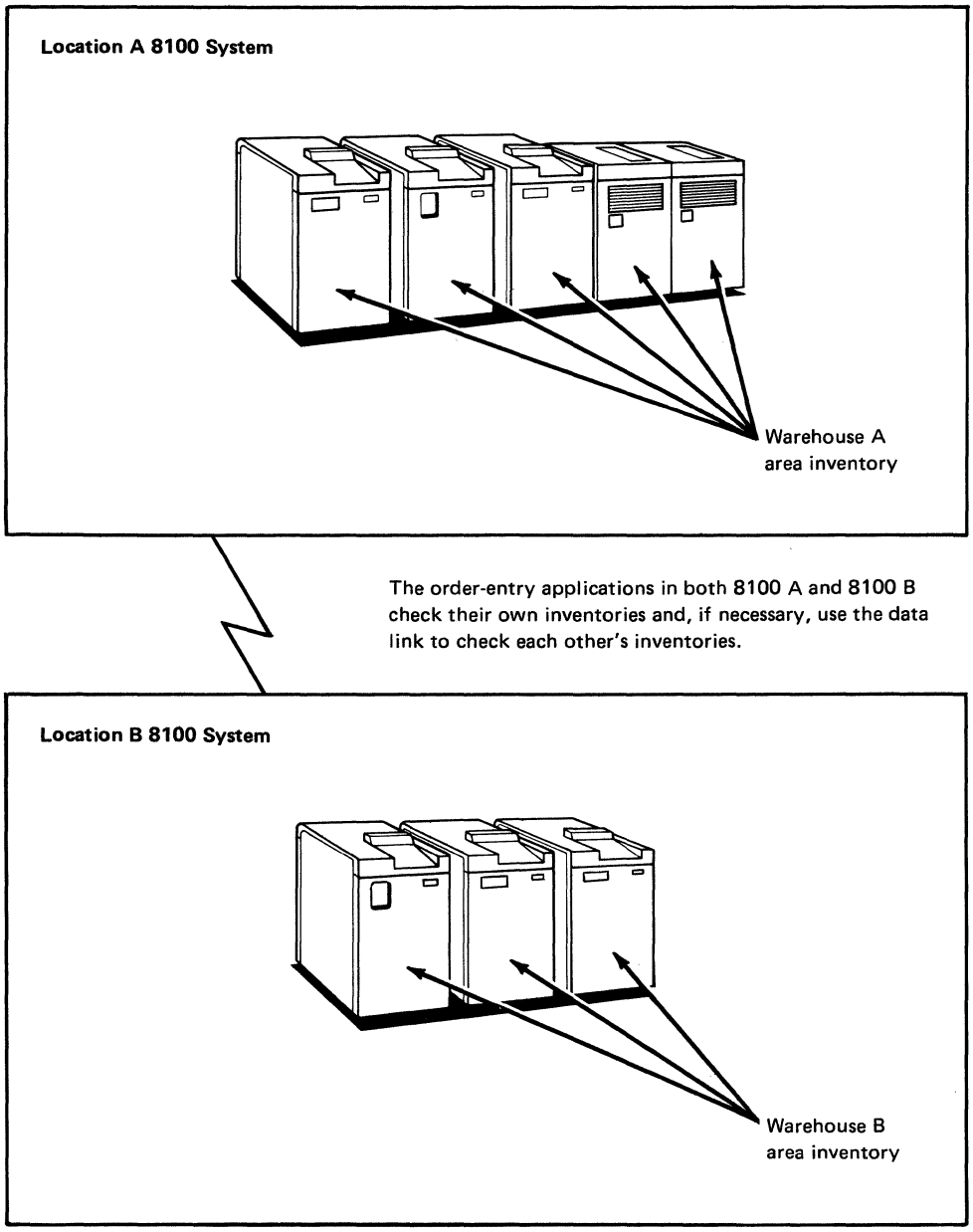


Figure 16. Application Programs Checking Order Entry

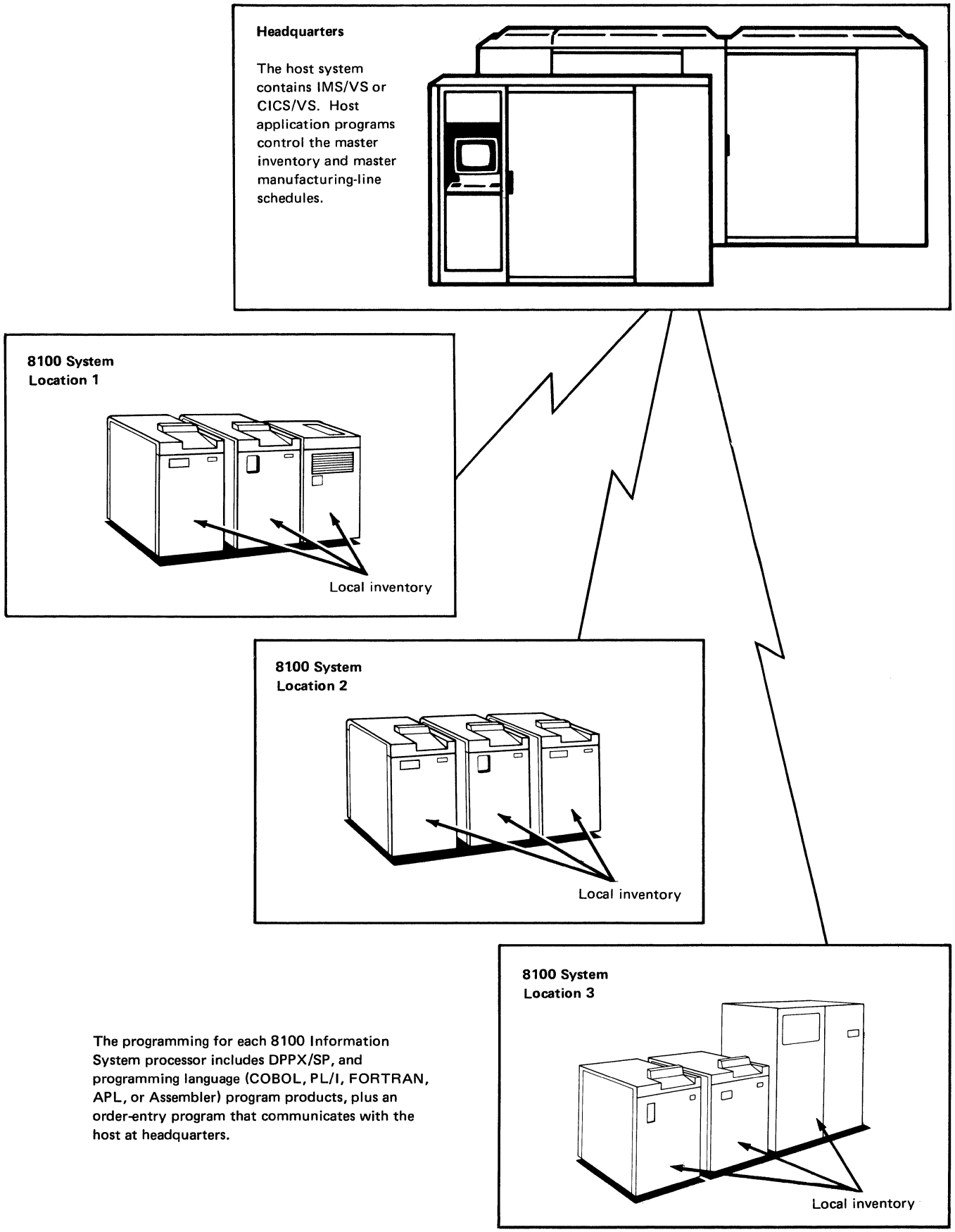


Figure 17. Central Control of Remote 8100 Information System Inventory

The headquarters application program controls the local factory's inventory data and manufacturing-line schedules. Given a local order, the order-entry application programs at each of the attached 8100 Information Systems check their local inventory.

If the part is not available locally, a request is sent to the host application program. That program checks the master inventory to see if the part is available at any of the other warehouse locations. Part shipments are scheduled, appropriate data changes are made when parts are shipped between any two locations, and manufacturing-line schedules are updated.

Data Distribution

DPPX/SP services allow data transfer between:

- Host and 8100 systems
- Two 8100 systems (including an 8100 with the Distributed Processing Control Executive [DPCX])
- Certain subsystems, such as the Personal Computer

The following sections highlight those DPPX/SP services and associated program products that enhance DPPX/SP's ability for distributing data.

Sending Data from a Host to DPPX/SP

Many business applications require data to be transferred between central and subsidiary locations. These transfers typically involve the central site sending data at intervals established by business needs.

When necessary, an application program can communicate with a host application program for retrieval and update of host data, as shown previously in Figure 17 on page 145

DPPX/SP facilities also allow batch exchange of data sets between host and 8100 locations. **Distributed Systems Executive (DSX)** is a host program product that can be used to transfer data sets from a host location to 8100 locations. With DSX, existing host facilities can be used to send data sets to 8100 sites.

In the host, DSX maintains a library of DPPX/SP data sets. Any DPPX/SP data set—including load modules, user data sets, executable command lists, and problem-determination data—can be stored and transferred using DSX. DSX schedules transmissions to and from various DPPX/SP locations in the network.

Figure 18 shows an example of how DSX is used. Assume that an enterprise has a 3081 host at location A and an 8100 at location B. The enterprise takes orders in the A area. Its B location is notified each morning of orders for parts that must be manufactured within one week of the order. Host utilities can be used to keep records of customers' orders. Once every morning, DSX is used to send the summary of these orders from location A to location B.

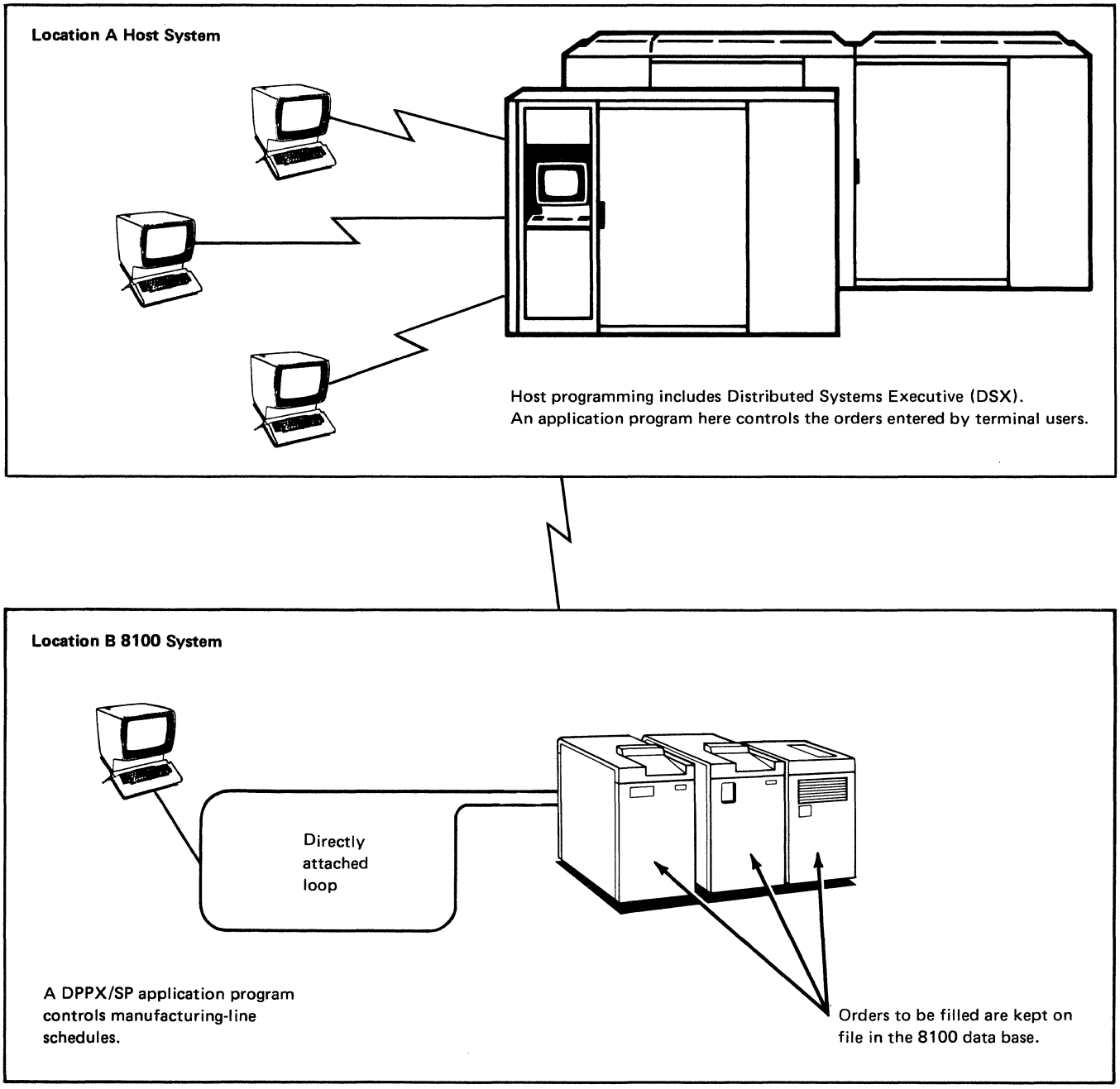


Figure 18. Host-to-8100 Data Transmission

DPPX/SP includes programming for host data transmissions. This programming is involved when:

- Data sets are sent from an 8100 to a host and from a host to an 8100
- Administrative messages are sent from a host to a DPPX/SP operator
- Command lists are sent from a host to an 8100 for automatic execution

Data transfer between DPPX/SP and a host application program takes place according to a defined systems network architecture (SNA) interface.

Sending Data from DPPX/SP to a Host

Figure 17 and Figure 18 show examples of enterprises that require data to be sent from an 8100 location to a host location.

The applications in Figure 17 continually contact the host application program, sending and receiving data about orders received and updates to be made.

The example in Figure 18 shows one of many possible business requirements for batched data set transfer from a central site to one (or more) business locations. If other aspects of the enterprise are handled centrally, such as billing, the ability to send data in the other direction—from individual business locations to the central site—becomes a requirement.

The business could add a billing application program to the 3081 processor in location A, for example, where the B location must notify the A location whenever finished goods are shipped. Using DSX, a person with proper authorization in location A can schedule automatic batch transfer of summary data sets with information about the shipments that take place in location B. The billing application program receives the data sets from DSX using existing host facilities.

Magnetic tapes may also be shipped to handle summary data set transfers.

DPPX/SP can send batch jobs and accompanying data in card-image form to a host for processing.

Sending Data between DPPX/SP Locations

Transfer of data from one 8100 location to another can take place with or without the intermediate assistance of a host.

When two or more 8100s are attached by data links to the same host, a person at the host location can use the DSX to schedule the transmission of data sets. An enterprise could receive data sets from one 8100 location and send the same data sets (or data sets that have some relationship to the received data) to one or more other 8100 locations.

When two 8100 systems are connected by a point-to-point telecommunication line, data can be sent back and forth between user-written application programs at one processor and terminals at the other. If a multipoint telecommunication line is used to connect more than two 8100s, an application program at the

"controlling" 8100 can receive data from and send data to terminals at any of the other processors; the other processors cannot transfer data among themselves, however.

Diskettes and magnetic tape can also be used to transfer data between 8100 systems.

Network Access Facilities

8100 systems can be added to existing networks to serve many purposes. For example, an 8100 allows off-loading of work from a host. Selected work formerly done as part of the host can be shifted to an 8100 either in the same room with the host or in a remote location.

When adding 8100 systems for such purposes, the number of terminals and lines added to support the additional hardware units can be reduced by using DPPX/SP and certain associated program products. The examples in Figure 19 explain how they enhance the processing capability of added 8100 systems.

In the first example, a terminal is remotely connected by a data link to a 3081 host. This host contains IMS/VS, CICS/VS, or TSO applications used by the terminal operator.

In example 2, an 8100 Information System has been added. Using DPPX/SP, the terminal can be shared between the 3081 and 8100 processors for application access. Line costs are reduced due to the sharing of the data link between the processors.

A terminal connected to a remote 8100 system can also access host facilities. Jobs to be processed by a host can be submitted in one of two ways:

- A terminal user can create a data set on disk, diskette, or magnetic tape and fill it using the DPPX/SP interactive editor. The user can then issue a command to place this job on the DPPX/SP job queue. When the Remote Job Entry component of DPPX/SP is started, the jobs are submitted to the host.

- Card decks containing jobs can be read by a card reader attached to the 8100. These jobs are transmitted directly to the host.

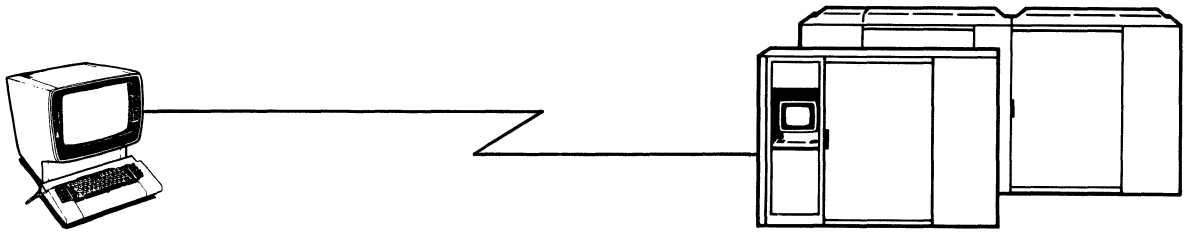
DPPX/SP prepares the data stream for transmission to the host. When the jobs have been processed, and the host program sends the output to the originating 8100 system, DPPX/SP receives the job output and sends it to the proper output device.

Data and Program Control

Different enterprises have different strategies for developing their application programs. The cost of duplicating effort is a determining factor in the selection of a strategy. It is possible to control development costs by assigning a few locations to specialized aspects of application programs' development. DPPX/SP facilities allow variations of data and program control. This permits various cost-effective development strategies.

The enterprise shown in Figure 20 has decided to control application program development at its 3081 location.

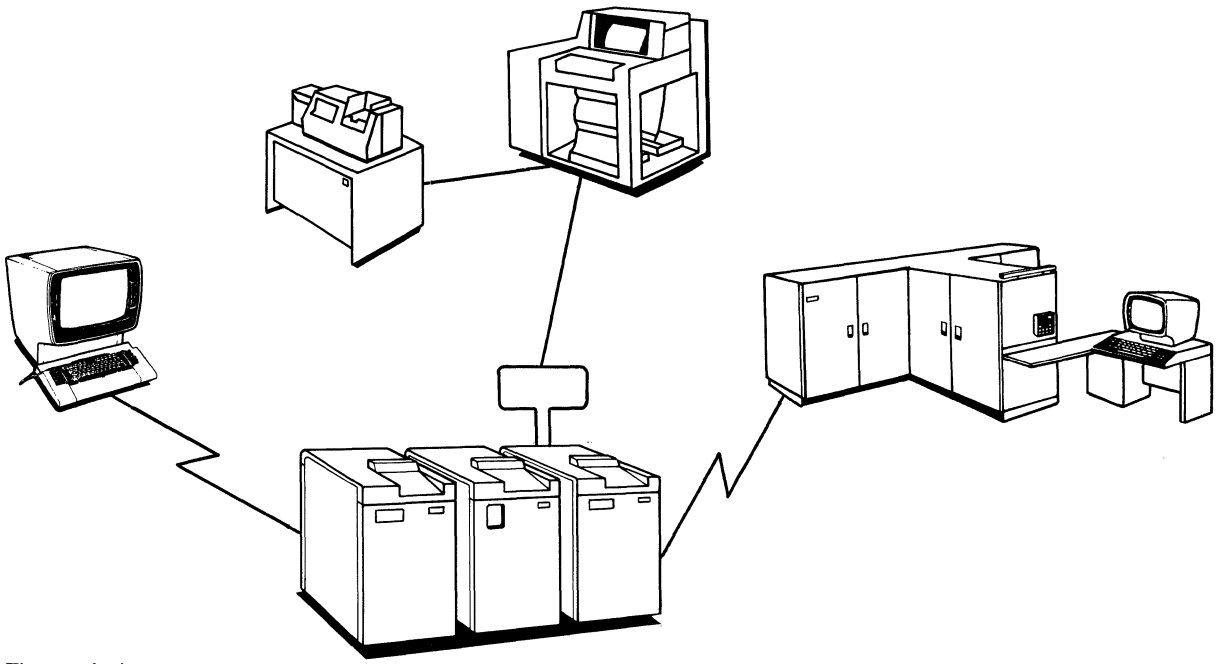
Example 1



The terminal operator can access IMS/VS, CICS/VS, or TSO applications.

The host contains IMS/VS, CICS/VS, or TSO applications.

Example 2



The terminal operator can access DPPX/SP applications and host (IMS/VS, CICS/VS, TSO) applications. Also, jobs to be processed at the host location can be initiated from the terminal.

The host contains JES2 and unchanged IMS/VS, CICS/VS, or TSO applications..

Figure 19. Network Access Examples Showing DPPX/SP

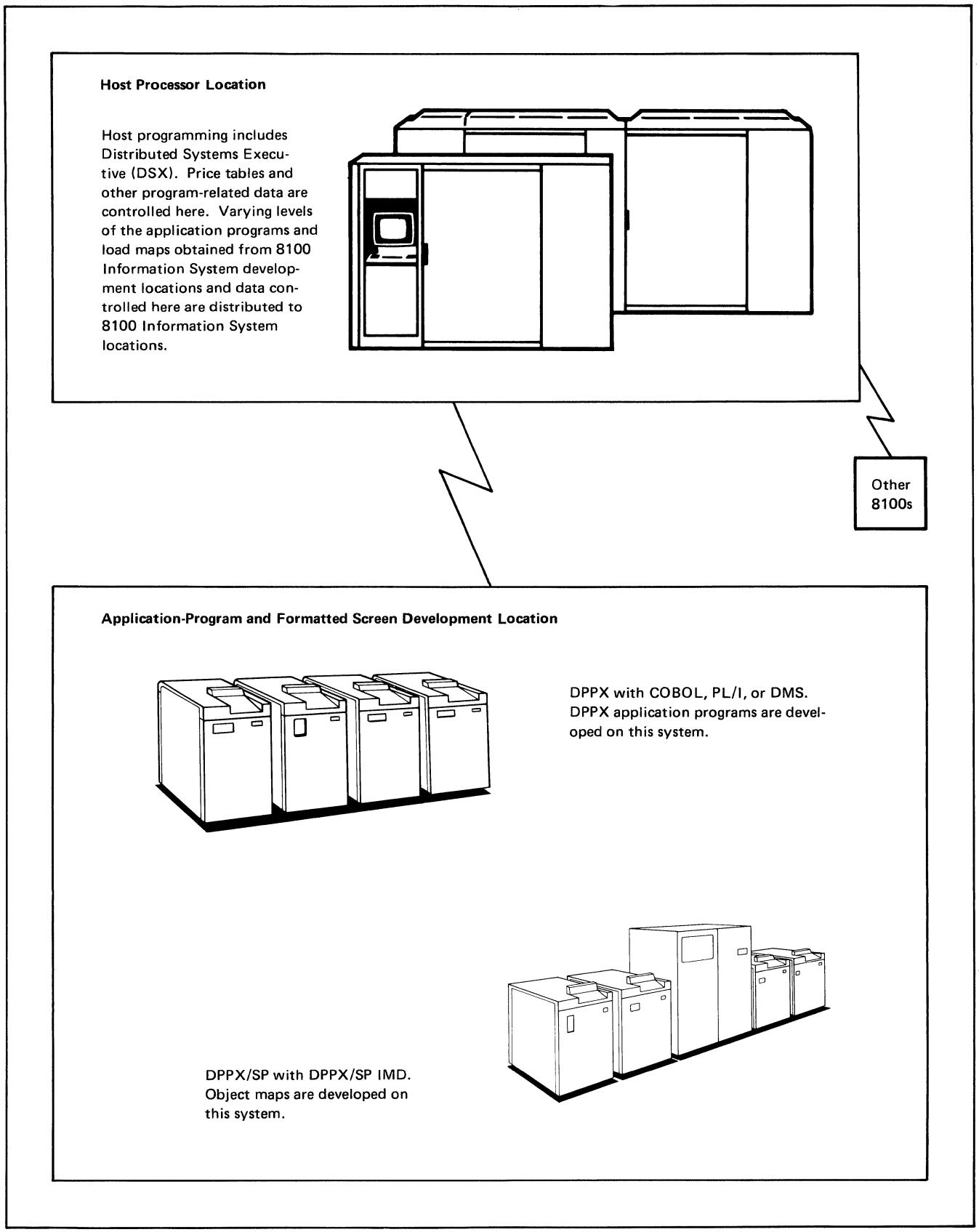


Figure 20. Distributed Application Development

Another location with two 8100 Information Systems has related application programming responsibilities. One system develops all of the application programs for the enterprise. The other 8100 system, which includes the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product, uses DPPX/SP IMD to develop presentation formats for keyboard-display operators.

DSX at the 3081 location obtains the data sets containing programs and presentation formats from the 8100 location. Someone at the 3081 location is responsible for managing program-related data (price tables, for example) and any command lists used to aid application program execution.

Different user-specified levels of the same application program or program data can be sent to different locations as necessary. Data sets can be sent from the 3081 host to selected 8100 systems on predetermined days or at certain times every day.

An Orderly Approach through Systems Network Architecture (SNA)

By using the Synchronous Data Link Control (SDLC) line control discipline, and by following other rules of systems network architecture (SNA) for the format, protocol, and operation sequences for transmitting information units through a communication system, DPPX/SP provides an orderly approach to data communication. This orderly approach:

- Provides a consistent and comprehensive structure for communication system growth
- Minimizes the effects of system changes
- Distributes network functions away from the host computer
- Allows sharing of network resources
- Supports many different kinds of devices
- Extends system functions conveniently and effectively to the user
- Minimizes the user's involvement in the details of system operation

Summary of Communication Support

Host Communication

Support for communication between an 8100 and a host computer consists of:

- Distributed Systems Executive (DSX), a program product that provides facilities to catalog, store, and distribute programs, maps, and other data sets using a host computer. It also provides the ability to receive diagnostic data at the host, permitting central control of a network of 8100 systems. This support is SDLC only.
- Host Command Facility (HCF), a program product that provides connection of a host terminal to an 8100-DPPX/SP system in SDLC mode. This allows access to remote 8100 systems from terminals at the central site.

DPPX/SP also provides host communication support:

- Host Transaction Facility (HTF), a component of DPPX/SP, lets DPPX/SP transaction applications communicate with host applications running under IMS/VS or CICS/VS. The HTF Direct facility of DPPX/SP 2 decreases the path length required to process messages, bypasses certain DTMS services and processes individual transmission requests than standard HTF processing.

- DPPX/SP lets certain displays, printers, controllers, and processors attached to an 8100 operate as if directly attached to a host.

- DPPX/SP provides remote job entry support for BSC (Binary Synchronous Communication) and SDLC links, with multileaving capability in BSC mode and multiple logical unit support in SDLC mode.

DPPX/SP-to-DPPX/SP Communication

DPPX/SP supports user application-to-application communication. This lets an application program in one 8100 exchange data with an application program in a peer 8100.

DPPX/SP-to-DPCX Communication

DPPX/SP supports communication with the Distributed Processing Control Executive (DPCX) and its Distributed Office Support Facility (DOSF).

DPCX can be attached upstream of DPPX/SP. Terminals and printers attached to DPPX/SP can access DPCX applications, including DOSF and text entry and edit under DOSF.

DPPX/SP can be attached upstream from DPCX. Terminals and printers attached to DPCX can access DPPX/SP applications. Data can be exchanged through a DPPX/SP application and an application in DPCX when DPCX is downstream from DPPX/SP. The DPCX data exchange application may be the Document Transmission Facility (DTF) or a user-written program.

Device Support

DPPX/SP provides a common interface to a wide variety of devices. Programming support for most devices is at a high level, such that the programmer deals with logical records without concern for physical device characteristics.

This support exists for the high-level languages, for DPPX Assembler languages, and for the Cross System Product/Application Development for DPPX/SP program product.

Line-Control Disciplines

DPPX/SP supports these line control disciplines for data communication:

- Synchronous Data Link Control (SDLC)—for communication with a host, another 8100, and most terminal types
- Binary Synchronous Communication (BSC)—for communication with a host using DPPX/SP components that support BSC host communication and for communication with certain terminals
- Start/stop—for communication with certain terminals

Telecommunication-Line Configurations and Types of Connection

SDLC communication between an 8100 and a host can be:

- Nonswitched, either point-to-point or multipoint.
- Switched point-to-point via:
  - EIA RS-232-C or CCITT V.24/V.28 interface. Only autoanswer is supported.
  - Data networks with the CCITT X.21 interface. The following X.21 capabilities are supported:
    - Address calling and autoanswer
    - Abbreviated address call
    - Direct call
    - Closed user group
    - Calls barred
    - Automatic registration

All other SDLC communication is nonswitched.

BSC communication is point-to-point nonswitched.

Start/stop communication is point-to-point nonswitched.

Ways of Attaching Devices to an 8100 with DPPX/SP

Devices can be attached to an 8100 with DPPX/SP in these ways (not all devices can attach in all ways):

- By loop, with the loop either directly attached or data-link attached
- By data link
- Directly

For More Information

For information about specific communication support provided by DPPX/SP, see "Communication Support" on page 53.

Part 3. Additional DPPX and Host-Related Program Products

Part 3 summarizes the important functions of each program product and each host-related program product that works with DPPX/SP. The host can be a 308x, 303x, 4300, or System/370 computer.

It is a reference section and not intended to be read from beginning to end.

Chapter 14. DPPX/SP Interactive Map Definition (DPPX/SP IMD)

DPPX/SP Interactive Map Definition (DPPX/SP IMD) is a program product (5660-282) that helps programmers specify how an application program should format data on display terminals and printers.

Advantages of Using DPPX/SP IMD

DPPX/SP IMD prompts the programmer to enter information about how the data should appear on the screen or printer. The programmer can either enter new information or accept the defaults that DPPX/SP IMD supplies. This interactive ability helps the programmer format data quickly and easily.

With DPPX/SP and DPPX/SP IMD, an application program's logic is separated from data formatting information, making coding simpler.

For example, it is irrelevant to the logic of an application program whether a particular piece of information appears at the top, bottom, or middle of a screen.

Formatting information is kept separately in maps.

Creating Maps

Before compiling the application programs, programmers create maps that contain information about the format in which data should appear on a screen or printer. These source maps are stored in a data set called the *map specification library.*

DPPX/SP IMD presents a series of questionnaire panels that the programmer uses to enter such information as size and position of fields, field length, field names (so that they may be referred to in the program), and data attributes. The programmer can see the screen layout as he or she creates the map. The programmer can make changes until the map is satisfactory. An online *tutorial* is available to give backup information at any stage in the process. The programmer can also read the tutorial sequentially, as an introduction to DPPX/SP IMD.

For a device such as the IBM 8775 Display Terminal, where the screen can be partitioned for use as a number of viewports, the process of map creation includes definition of the partition sizes and positions.

Before DPPX/SP IMD can use a map, the programmer must generate the map. Generation also produces DPPX COBOL, DPPX PL/I, or DPPX Assembler language declarations, called *application data structures.* These describe the layout of the data sent by the application program. They can be retrieved from the appropriate source statement library when the application program is compiled. Examples of statements to do this are the COPY statement in DPPX COBOL and the %INCLUDE statement in DPPX PL/I.

These statements are also used to include a control structure for passing control information between DPPX/SP and the application program during execution.

The IMD functions are also available through Cross System Product/Application Development for DPPX/SP. The user may use Cross System Product (not IMD) facilities to define maps, and the created maps are compatible with Cross System Products and IMD.

Maps Can Be Distributed

Maps created under DPPX/SP IMD at one 8100 can be sent for execution by DPPX/SP to other 8100s where DPPX/SP IMD is not installed. The maps can be sent using the Distributed Systems Executive (DSX) program product.

Device-Independent Coding

DPPX/SP IMD allows device-independent coding. Layouts that correspond to the same DPPX COBOL, DPPX PL/I, or DPPX Assembler language declaration can be prepared for any DPPX/SP terminal, and the required layout chosen and automatically applied at execution time. Exceptions arise when application programs are to be run as floating maps on multiple screen sizes.

Format-Independent Coding

DPPX/SP IMD allows format-independent coding. Layouts that correspond to a DPPX COBOL, DPPX PL/I, or DPPX Assembler language declaration can be changed without changing the application data structure itself and, consequently, without altering or recompiling the application program.

For example, to introduce color, a user could edit and regenerate the map. As long as no changes are made to the corresponding application data structure, no changes need be made to the application program, and it does not need to be recompiled.

Another example is that heading data in the screen layout can be changed to suit different purposes. End users who speak a different language can use the same program, with alterations to headings being made by using a different copy of the map at different locations.

Screen Functions Supported

DPPX/SP IMD, with DPPX/SP, lets programmers write programs that use these screen functions (on terminals that have them):

Full screen
Protected and unprotected fields
Keyboard freeing
Constant and variable fields in panel
Modified data tag
Field selector tag
Program access and attention keys
Program function keys
Alarm
Multiple partitions
Extended highlighting
Programmed symbols
Extended color

Scrolling
Intensity
Field validation
Light pen selection
Cursor positioning

Outboard Formatting

DPPX/SP IMD, with DPPX/SP, supports distributed data processing through outboard formatting for applications on systems network architecture (SNA) connections. This can reduce line traffic and improve response time for applications using the Customer Information Control System/Virtual Storage (CICS/VS) program in the host. For more information, see "Outboard Formatting and DPPX/SP Data Formatting" on page 97.

DPPX DPS Users Migrating to DPPX/SP IMD

The DPPX/SP IMD program product provides all the existing functions and application interfaces of the IMD feature of DPS, plus the enhancements listed below:

- The ability to update objects in one map specification library and to list and make references to objects in up to five other map specification libraries.
- Improved error checking and reporting (through lists of DPPX/SP IMD messages) in the steps of the mapgroup generator.
- The ability to display the maps in a mapgroup so that the programmer can see how they will appear on the display device. This is done with the Mapgroup Test step.
- Reduced disk storage requirements.
- Support of the new DPPX/SP mapping functions described under "DPPX Users Migrating to DPPX/SP" on page 98.

Compatibility between DPPX/SP IMD and the IMD Feature in DPS

Application programs that run under any version of DPS will run under DPPX/SP without modification. DPPX/SP IMD accepts source maps created by DPS Versions 1 and 2. DPS Version 1 source maps are converted to DPPX/SP IMD format before they are edited by DPPX/SP IMD. DPS Version 2 maps can be used without conversion, but you should convert them if you want to use all the functions and get the best results. You can convert source maps back to DPS Version 2 format but not to Version 1 format. Object maps generated by DPPX/SP IMD require a DPPX/SP system for execution.

Chapter 15. Cross System Product Set for DPPX/SP

The Cross System Product Set for DPPX/SP forms a productivity tool for developing and running application programs without using a programming language. It guides users online in writing application programs by requesting responses to prompts at display terminals. It also executes the application programs developed.

There are two Cross System Products for DPPX/SP:

- Cross System Product/Application Development for DPPX/SP (5660-284), for developing, testing, and generating Cross System Product programs
- Cross System Product/Application Execution for DPPX/SP (5660-285), for running Cross System Product programs in a production environment

Portability and compatibility exist for Cross System Product applications between systems. Any Cross System Product—defined application, regardless of the system environment, can be sent and executed in any other supported environment. Care must be taken in certain cases, as in DPPX to DPCX, to use only Cross System Product functions.

The Cross System Product Set for DPPX/SP uses the format management services of DPPX/SP and the services of the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product to create maps. The user only sees Cross System Product prompts, but the Cross System Product takes the user's input and creates a compatible IMD map.

Using the Cross System Product Set for DPPX/SP is a convenient and simple way to interactively define data structures, maps (screen formats), and processing statements; test these definitions; and run them in production. The Cross System Product Set offers some important advantages over programming languages:

- It requires less programming experience and skill.
- Cross System Product/Application Development for DPPX/SP takes the initiative by prompting the user for information that it can turn into a program.
- Cross System Product/Application Development does not require the user to know technical system interfaces.

Situations where Cross System Product Set might not be advantageous are:

- You have existing host programs (such as OS/VS COBOL programs) that you want to recompile and run on the 8100.
- You need access to more system services the Cross System Product Set for DPPX/SP allows.

Easy Program Development

A user who needs help from anywhere in Cross System Product/Application Development can get it by pressing PF 1 or PF 13. This calls up a display that further explains the current application development function or message.

Application program development is done interactively. (Application program definition with Cross System Product/Application Development is much like a data dictionary approach to program definition.) Once the definitions are complete, the user can execute the application program in test mode by using the test facility. The user can use trace options to help debug the application.

A user can make corrections to a program after running the test facility. Because changes are made interactively, a user can quickly make modifications and retest the program with the changes.

When the application program is ready for production, the final definitions are generated into an image of that application. An application can be generated to execute under DPPX/SP as conversational, segmented conversational, or both.

Member Specification Library

Data definitions, map definitions, and application definitions are stored in a user's *Member Specification Library (MSL).* A Cross System Product/Application Development user references only one MSL at a time and cannot share that library with others while using it. However, a definition to be used by several application programs can be copied to separate MSLs to avoid redefinition.

The format of the member specification library is compatible with that of the DPPX/SP IMD program product.

Application Program Development Facilities

Cross System Product/Application Development for DPPX/SP provides five interactive facilities for developing, testing, and maintaining application programs:

List processor: Lets users list all members in the member specification library and perform multiple utility functions against the members. These functions are copy, rename, delete, print, change, where used, edit, view, and export.

Definition: Allows users to define data structures, maps, and application logic. In the data definition function, single elements of data and their characteristics are defined as data items. Each data item name is unique within the member specification library. Data items are components of a record (which is associated with a file), table, or working storage. After a data item has been defined once, you may use it in other records, tables, working storage, and maps by entering just the data item name. Its characteristics need not be reentered, since they already exist in the member specification library.

When a user defines a data item, Cross System Product/Application Development for DPPX/SP provides default characteristics. For example, numeric fields with decimal positions default to right justification. Standard editing includes data type verification (that is, numeric, character, binary, and packed), length, and number of occurrences in record.

In the map definition functions, users may define maps for terminal displays and printers. Each map is given a unique name. The variable fields on the map are named and characteristics are defined for each variable field. If a new name is used to define a variable field, the user may enter the characteristics. If the user enters none, Cross System Product/Application Development for DPPX/SP provides default characteristics.

In the application definition function, users may define processing to be performed by the application program. Processing statements can be used to define such functions as arithmetic, movement of data, use of tables, and access to map fields and record data items. Processing flow may be conditional or unconditional.

Sample definitions are shipped with the product. Users can use IBM-provided models as is or customize them to fit specific requirements.

Test and generation: Users can verify the syntax and logic design of any Cross System Product/Application Development–defined application program. Users can also review the logical sequence of map displays as they will be seen in the production environment. If this review is done without tracing, the application program will perform exactly as it would in production, depending on the data in the test data sets and the input from the tester at the terminal. If the review is done with tracing, the tester is allowed to select trace options.

Generation allows the user to generate an application program to run under the Cross System Product/Application Execution for DPPX/SP program product. This generation consists of an application image and load modules. The user can also generate command lists for startup and termination, and a reference file.

Utilities and file maintenance: Users can copy, delete, print, rename, and move (from one member specification library to another) map definitions, data definitions, and application definitions. File maintenance allows users to display and change the contents of any file.

Tutorial: With the full-screen tutorial users can learn about Cross System Product/Application Development for DPPX/SP by reading it as if it were a manual or by referencing it to review certain sections. It can be accessed at any time by pressing the "help" program function key.

Execution

Cross System Product/Application Execution for DPPX/SP provides:

- Execution of all Cross System Product/Application Development–developed definition statements
- The ability to execute both batch applications and interactive applications under DPPX/SP
- The ability to access host applications and data through the DPPX/SP program product
- Concurrent execution of Cross System Product/Application Development–developed applications with other DPPX/SP programs

- The ability to call user-written Cross System Product/Application Development, DPPX COBOL, DPPX PL/I, and DPPX Assembler language subroutines
- Access to DPPX/SP data base management and transaction processing services
- Display of the original Cross System Product/Application Development source statement if an error should occur
- Selectable performance, allowing the user to indicate either conversational or segmented conversational execution

Testing and Production Environments for the Cross System Product Set

The table below shows the differences in function provided by the test environment of Cross System Product/Application Development and the production environment of Cross System Product/Application Execution for DPPX/SP. (Access to host application programs or data is provided by the Host Transaction Facility component of DPPX/SP.)

Functions or Characteristics	Test	Production
Host communication	No	Yes
Run-time environment		
Batch	No	Yes
Interactive	Yes	Yes
Run-time characteristics		
Conversational	Yes	Yes
Segmented conversational	No	Yes
Data base access	Yes	Yes

Benefits of the Cross System Product Set for DPPX/SP

The Cross System Product Set for DPPX/SP:

- Allows portability and compatibility of Cross System Product/Application Development applications to other supported environments. The supported systems are 8100, 4300, System/370 with CICS/VS, SSX/VSE, VM/SP CMS, or OS/VS 2 TSO. Also, Cross System Product/Application Development may be run on DMS/DPCX, but care must be taken to ensure only Cross System Product/Application Development definitions are used.
- Allows fast application development. Cross System Product/Application Development for DPPX/SP is self-teaching and menu-driven. Users can become proficient quickly and with minimal training.
- Lets non–data processing personnel do some program development and helps data processing professionals be more productive.
- Reduces the level of manual preparation of documentation, because the system is self-documenting.
- Lets those with limited resources justify interactive systems.

- Lets changes in physical configuration be made with minimum user effort and software impact.
- Has well-documented messages. Numerical error codes and single messages for multiple error conditions are avoided.
- Has short, simple commands with consistent syntax, designed to be easily understood.

Chapter 16. DPPX COBOL Compiler

COBOL is a programming language suited to developing business applications. It is especially efficient in the description and manipulation of files of formatted data. The name COBOL stands for COmmon Business-Oriented Language.

DPPX COBOL is a level-1 implementation of 1974 American National Standard (ANS) COBOL (ANSI X3.23-1974).

It also has most of the features in the December 1975 Federal Information Processing Standard (FIPS), PUB 21-1, low-intermediate level.

In addition, DPPX/COBOL provides language extensions to writing structured programs, an option to create reentrant code, and CALL statements (HCEPOPEN, HCEPCLSE, HCEPWRIT, HCEPREAD) to use the maps created by the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product.

The DPPX COBOL Compiler (5760-CB1) program product checks syntax and generates object code. DPPX/SP contains reentrant routines that perform arithmetic, logic, data conversion, and input/output operations.

The DPPX COBOL Compiler program product does not have to be installed in every 8100 that has DPPX/SP. A development 8100 could have DPPX/SP and the COBOL Compiler to compile and test application programs. A production 8100 could have just DPPX/SP to run programs that were compiled and tested on the development 8100.

Using DPPX COBOL

Developing a DPPX COBOL program generally follows these steps:

1. Designing the program.

2. Coding the program.

3. Entering the source program into a data set using the DPPX/SP interactive editor.

4. Compiling the program, initiated interactively or in batch mode. IBM provides command lists to do this. Users can specify options to control the Compiler's operation.

   The Compiler produces two data sets: one contains an object module and the other a listing. The listing can contain:

   - The source program input
   - Errors the Compiler discovered
   - The generated code, with mnemonic operation codes and operands
   - The relative address of each statement with respect to the start of the program
   - The compiler options in effect
   - Information on data items, such as location and attributes
   - Alphabetized cross-references of user-defined names

5. Link-editing the program. IBM provides command lists to do this. Link-editing forms an executable load module from the object module input. The linkage editor can also combine subprograms of a multiprogram run unit, edit subprograms out of a load module, and set up an overlay structure.

   Instead of using the linkage editor to include subprograms of a multiprogram run unit in the same load module, users can leave the subprograms to be automatically loaded at run time as they are needed. All the library routines are handled in this manner.

6. Executing the program.

7. Debugging the program. Debugging aids are discussed below.

Dialogs are available to help in creating source code.

Language Interface to DPPX/SP

DPPX COBOL provides native language statements for using all DPPX/SP data base and transaction services.

COBOL input/output language is used for access to data bases.

Extensions are provided when ANS COBOL has no language to use a DPPX/SP function. The COMMIT statement is provided for committing changes to a data base when processing completes successfully. And the ROLLBACK statement is provided for resetting data to prior values when an abnormal condition occurs.

(See "DPPX/SP Manages Data Bases and Processes Transactions" on page 61 for information about services DPPX/SP provides.)

CALL Interface

COBOL programs can use all DPPX/SP format management services and DPPX/SP IMD program product services via a CALL interface. (See "DPPX/SP Presents Data according to User-Developed Maps" on page 96 for information about DPPX/SP data formatting services. See "Chapter 14. DPPX/SP Interactive Map Definition (DPPX/SP IMD)" on page 157 for information on DPPX/SP IMD.)

Structured Programming

Structured programming makes programs easier to debug and modify. Typically, a structured program is a hierarchy of modules, each of which has a single entry point and a single exit point. Control is passed downward through the structure without branches to higher levels of the structure.

DPPX COBOL includes language to aid in structured programming:

- Explicit scope terminators provide an orderly notation for nesting conditional statements. For example, the statements IF...END-IF can be specified anywhere an imperative statement is allowed in the COBOL language.

- New PERFORM statement formats allow the programmer to code loops inline, that is, without having to define paragraphs located elsewhere to contain the statements to be performed.

- A new statement, EVALUATE, causes multiple conditions to be tested in order to make a choice among multiple resulting actions (a "case" construct).

Optimized and Reentrant Code

The programmer can direct the DPPX COBOL Compiler to optimize the generated code in the Procedure Division of a program. Optimized code will occupy less space and may run faster.

The programmer can direct the Compiler to create reentrant object modules. Reentrancy saves main storage when a program is to be run concurrently by more than one user. A single reentrant copy of the program, rather than a copy for each user, resides in main storage during program execution.

A user can specify both capabilities by the Compiler option.

Application-to-Application Communication

Two user-written DPPX COBOL application programs can communicate with each other. The programs can be in the same or different 8100 processors. After certain configuration commands are issued, READ and WRITE statements within the COBOL programs control the sending and receiving of data between the programs.

Command Invocation from within a Program

Through a call to a DPPX/SP library routine, DPPX/SP or user-written commands can be issued from a DPPX COBOL program.

Calls to Non-COBOL Programs

A COBOL program cannot directly call a program coded in another high-level language. But a COBOL program can call an Assembler language program, which in turn can call a program coded in another high-level language.

Shared Storage

DPPX/SP library routines and user programs can be made resident. They can then be shared by all COBOL programs running concurrently. This improves performance.

Debugging Aids

To help programmers locate errors in their programs, these COBOL debugging statements can be put into the source program:

- Debugging lines—any DPPX COBOL statement added to the program and identified by a D in column 7. Including the WITH DEBUGGING MODE clause in the SOURCE-COMPUTER paragraph tells the Compiler to include the debugging lines in the object code.

- USE FOR DEBUGGING declaratives, which let programmers create their own procedures to examine or alter the internal states of programs during execution. The declaratives can be activated or deactivated by command prior to execution.

- The TRACE statement, an IBM extension, which allows programmers to display or print all or parts of a program's logic flow.

- The EXHIBIT statement, an IBM extension, which allows programmers to display or print the values of data items at particular points during the execution of a program.

ANS COBOL Comparison

The two COBOL sections are designed according to the level-1 specifications of American National Standard (ANS) COBOL, ANSI X3.23-1974, as understood and interpreted by IBM as of September 1978.

The ANS COBOL language is organized into functional modules, and each language element is classified as being at level-1 or level 2. Level-2 language contains all the elements in level-1, plus additional elements.

The modules in DPPX COBOL are as follows. (The first digit represents the level of the modules in DPPX/SP. The second digit represents the lowest level in the American National Standard. [0 implies that the module may be completely missing from some standard compilers.] The third digit represents the highest level of the American National Standard.)

1 NUC 1,2 (Nucleus): Includes the basic language elements for program structure and data descriptions; statements for arithmetic, comparison, data manipulation, and procedure branching; and miscellaneous language elements. DPPX COBOL supports all elements of ANS COBOL level 1, plus the following:

- Nested conditional statements
- MOVE CORRESPONDING
- Qualification (up to 50 levels)
- COMP-3 (packed data) (IBM extension)
- COMP-4 (binary data) (IBM extension)
- Conditional-names (level 88 data items)
- Sign condition (POSITIVE, NEGATIVE, ZERO)
- Complex conditions (AND, OR, NOT, combined and abbreviated combined)
- Series form of arithmetic statements
- Arithmetic expressions and COMPUTE statement (with some restrictions)
- STRING and UNSTRING statements (with some restrictions)
- PERFORM statement, all formats
- Structured programming (IBM extensions—Scope terminators to nest statements, inline PERFORM statement, and EVALUATE statement)
- SET condition-name (IBM extension)

1 TBL 1,2 (Table Handling): Allows the definition of tables of up to three dimensions and reference to these tables by both subscripting and indexing. DPPX COBOL supports all elements of ANS COBOL level 1, plus the following:

- SEARCH statement, format 1

1 SEQ 1,2 (Sequential I-O): Allows the definition of sequential files and serial access to the records in them. DPPX COBOL supports all elements of ANS COBOL level 1 except:

- RERUN clause
- STANDARD-1 alphabet-name referred to in the CODE-SET clause
- The optional words REEL and UNIT in the CLOSE statement

DPPX COBOL also includes the following:

- SAME RECORD AREA clause
- LINAGE clause (with some restrictions)
- File and record-locking options (IBM extension)

1 REL 0,2 (Relative I-O): Allows the definition of direct-access files and access to records in them either directly (by relative record number) or sequentially. DPPX COBOL supports all levels of ANS COBOL level 1 except:

- RERUN clause
- The data restriction for Relative and Indexed I/O that an application program must not write a record with hexadecimal FFFF in the first two characters. DPPX/SP uses that convention to indicate a deleted record.

DPPX COBOL also includes the following:

- SAME RECORD AREA clause
- File and record-locking options (IBM extension)
- DYNAMIC access mode

1 INX 0,2 (Indexed I/O): Allows the definition of direct-access files in which each record is identified by the value of a key. Access is by means of an index, and can be either direct (by key) or sequential. DPPX COBOL supports all elements of ANS COBOL level 1 except:

- RERUN clause

DPPX COBOL also includes the following:

- SAME RECORD AREA clause
- File and record locking options (IBM extensions)
- Data base support through DPPX/SP, including COMMIT and ROLLBACK statement (IBM extensions)
- DYNAMIC access mode
- Alternate indexes, via separate File Descriptions (IBM extension)
- Duplicate keys allowed on the prime index (IBM extension)

1 SEG 0,2 (Segmentation): Allows the overlaying of sections of a program at execution time in order to use main storage more efficiently. DPPX COBOL supports all elements of ANS COBOL level 1.

1 LIB 0,2 (Library): Allows the insertion of previously written COBOL statements from a COBOL source-statement library into a source program at compile time. DPPX COBOL supports all elements of ANS COBOL level 1.

1 DEB 0,2 (Debug): Allows the monitoring of execution of a COBOL program to locate errors, under conditions specified by the programmer. DPPX COBOL supports all elements of ANS COBOL level 1, plus the following:

- EXHIBIT statement (IBM extension)
- TRACE statement (IBM extension)

1 IPC 0,2 (Interprogram Communication): Allows one COBOL program to invoke another. Parameters can be passed, enabling both programs to have access to the same data items. DPPX COBOL supports all elements of ANS COBOL level 1, plus the following:

- RETURN-CODE special register (IBM extension)
- CALL USING parameters may be identifiers or nonnumeric literals (IBM extension)

Sort: Not represented in DPPX COBOL.

Report Writer: Not represented in DPPX COBOL.

Communication: Not represented in DPPX COBOL.

OS/VS COBOL Comparison

Most differences between DPPX COBOL and OS/VS COBOL have to do with the language definition. The major differences are:

- DPPX COBOL permits data base I/O with native language statements, rather than requiring an intermediate Assembler language interface.
- DPPX COBOL contains terms for structured programming.
- DPPX COBOL offers "call-by-content" as well as "call-by-reference."
- No dual language level support in DPPX COBOL (LANGLVL option); where DPPX COBOL conforms to the ANS standard (as it mostly does), it invariably conforms to the 1974 version—not to 1968.
- No Report Writer feature in DPPX COBOL.
- No Communication Feature as such in DPPX COBOL. However, terminal I/O support is available through DPPX/SP and the DPPX/SP IMD program product.
- No sort/merge language in DPPX COBOL. But DPPX/SP lets you sort, merge, and copy records.
- No RERUN (Checkpoint/Restart facility) in DPPX COBOL.

The major differences other than language content are:

- DPPX COBOL will generate reentrant code. All its library subroutines are reentrant.
- No BASIS support in DPPX COBOL (only COPY). There is also no "replacing" facility on its COPY.
- No Lister feature in DPPX COBOL.
- No FIPS flagging in DPPX COBOL.
- No batch compilation of two or more programs in DPPX COBOL.
- In DPPX COBOL, no compile-time control over whether user-written subprograms will be linked or dynamically loaded at execution time. The user makes the choice at link-edit time.
- In DPPX COBOL, all the library subroutines are handled dynamically (being either in shared storage or loaded during program initialization). Users cannot choose to link them with their load modules.
- No execution-time symbolic dump facility (SYMDMP option) in DPPX COBOL.
- No comprehensive error message printing facility (ERRMSG) in DPPX COBOL.
- Error severity levels and message numbers and texts differ.
- Options for compile time and run time (execution time) differ.
- Arithmetic results in DPPX COBOL will not always be the same as those in OS/VS COBOL. Reasons are:
  - If diagnostic messages are issued during compilation, arithmetic results may not be the same.
  - For a divide operation (not the DIVIDE verb), DPPX COBOL may carry more precision than OS/VS COBOL.
  - Truncation, when necessary, affects the rightmost digits for DPPX COBOL.
- In general, DPPX COBOL offers fewer options than OS/VS. However, its options usually give more function and selectivity.

Using DPPX COBOL Programs with Data Bases

DPPX COBOL programs that use data sets do not require source language changes to access the data sets as data bases. The user simply issues commands to redefine the data sets as DPPX/SP-managed data bases.

However, one of the main assets of DPPX/SP is its recovery capability. The capability, requires COMMIT and ROLLBACK statements. Adding these statements to a program would require recompilation.

Chapter 17. DPPX PL/I

PL/I is a general-purpose programming language. It is designed for business, scientific, and system programming applications that involve the manipulation of numerical and nonnumerical data. PL/I is the most versatile of the major high-level programming languages.

DPPX PL/I is a subset of American National Standard PL/I (ANSI X3.53-1976). Its design closely follows that of the American National Standard PL/I General-Purpose Subset (ANSI X3.74-1981). DPPX PL/I therefore contains the more important features of ANSI PL/I yet is easier to learn and understand. Programs written in DPPX PL/I are less likely to contain programming errors than programs written in the full language, because DPPX PL/I is both simpler and less permissive than the full language.

In addition, the DPPX PL/I programmer has access to DPPX/SP data base management and transaction processing services by using CALL statements, and to DPPX/SP data formatting services, by using built-in subroutines.

DPPX PL/I consists of two program products:

- The DPPX PL/I Compiler (5760-PL1), which checks syntax, corrects errors where possible, and generates object code
- The DPPX PL/I Library (5760-LM2), which contains routines to set up and manage the execution environment, perform such often-used functions as opening a file or calculating a square root, and provide access to DPPX/SP

The DPPX PL/I Compiler and DPPX PL/I Library do not have to be installed in the same 8100. A development 8100 could have both the compiler and the library to compile and test application programs. A production 8100 could have just the library to run programs that were compiled and tested on the development 8100.

Using DPPX PL/I

Developing a DPPX PL/I program generally follows these steps:

1. Designing the program.
2. Coding the program.
3. Entering the source program into a data set using the DPPX/SP interactive editor.
4. Compiling the program, initiated interactively or in batch mode. IBM provides command lists to do this. Users can specify options to control the Compiler's operation, the listing printed by the Compiler, and the object module produced by the Compiler.

   The Compiler produces two data sets: one contains an object module and the other a listing. The listing can contain:

   - The source program input with statement numbers and an indication of the level of nesting

- Alphabetical cross-references of the identifiers used and their attributes
- A storage map for data items
- The relative address of each statement with respect to the beginning of the program
- A listing of the generated code, with mnemonic operation codes and operands
- Description of errors down to a specified severity level
- The Compiler options in effect

5. Link-editing the program. IBM provides command lists to do this. Link-editing forms an executable load module from the object module input. The linkage editor can also combine subprograms to form a multiprogram run unit, edit subprograms out of a load module, and set up an overlay structure.

   Instead of using the linkage editor to include subprograms of a multiprogram run unit in the same load module, users can leave the subprograms to be automatically loaded at execution time as they are needed.

6. Executing the program. The program would typically be invoked in one of these ways:

   - As a transaction by DPPX/SP
   - By the CALL.PROGRAM command, to execute the program with the DPPX/SP Command Facility
   - By the PLI.CALL command, which differs from CALL.PROGRAM in that execution options can be specified as parameters in this command. (Execution options are listed under "Run-Time Options" on page 187.)
   - As a command processor, where a command can be defined for the program, allowing the arguments to be provided by way of meaningful keywords and in a form suitable for processing by the program (for instance, FIXED BINARY)
   - As an exit routine from either the format management or sorting and merging services of DPPX/SP.

7. Debugging the program. Debugging aids are discussed under "Debugging Aids" on page 178.

Using DPPX/SP Data Base Management and Transaction Processing Services

DPPX PL/I programs can use all DPPX/SP data base management and transaction processing services. PL/I record-oriented data transmission is used for accessing data bases by means of READ, WRITE, REWRITE, and DELETE statements. CALL statements are used in the PL/I program to access other DPPX/SP services.

Using DPPX/SP Data Formatting Services

Built-in subroutines let DPPX PL/I programs use all the data formatting services of DPPX/SP.

Interface to Distributed Presentation Services

Built-in subroutines allow DPPX PL/I programs to use all data formatting services of DPPX/SP Format Management (FM).

Reentrant Code

The DPPX PL/I Compiler creates reentrant object modules. Reentrancy saves main storage when a program is to be run concurrently by more than one user. A single reentrant copy of the program, rather than a copy for each user, resides in main storage during program execution.

Application-to-Application Communication

Two user-written DPPX PL/I application programs can communicate with each other. The programs can reside in the same or different 8100 processors. After certain configuration commands are issued, READ and WRITE statements within the PL/I programs control the sending and receiving of data between the programs.

Command Invocation from within a Program

DPPX/SP or user-defined commands can be issued from a DPPX PL/I program by means of a built-in subroutine.

Multiple Compilations

The DPPX PL/I Compiler allows compilation of more than one PL/I external procedure with one invocation of the Compiler. The individual external procedures are separated by *PROCESS statements. This reduces job stream and Compiler processing overhead.

Shared Storage

Library routines and user programs can be loaded into shared storage. They may then be shared by all DPPX PL/I programs running concurrently. This can improve performance by reducing storage requirements.

Real or Simulated Floating-Point

Floating-point arithmetic is often used in PL/I programs. It is always used by the mathematical built-in functions. The programmer can direct the DPPX PL/I Compiler either to produce floating-point machine instructions or to simulate floating-point operations.

Floating-point machine instructions can be used only if the hardware floating-point feature has been installed on the processor. They are generally far more efficient than simulated operations. Simulated floating-point operations can be used on all processors, whether or not they have the floating-point feature.

Debugging Aids

DPPX PL/I provides aids to reduce the time and effort needed to write and check a program. Some aids may be used while the program is being compiled. Others may be used while the program is being run.

Compilation Debugging Aids

Diagnostic messages describe errors found while a program is being compiled. They indicate:

- The number of the statement where an error was detected (if applicable)
- The source item within the statement involved (if applicable)
- The cause of the error
- Any assumptions made or actions taken by the Compiler

Messages are classified in five levels of severity. The programmer may optionally specify the level at and above which messages are to be printed.

The Compiler provides information on the program and its usage of storage. This information helps the programmer correct and improve the program.

Run-Time Debugging Aids — Batch Processing

Diagnostic messages describe errors found while a program is being run. They indicate:

- The name of the external procedure where the error was detected
- The position of the instruction in storage, relative to the start of the external procedure, where the error was detected
- The number of the source statement where the error was detected, if the GOSTMT option was specified when compiling the program
- The cause of the error
- The condition name, if applicable
- The condition code (ONCODE value)
- The name of the file (ONFILE value) and the access key of the record (ONKEY value), if applicable
- The DPPX/SP return code, if applicable.

DPPX PL/I supports PL/I error handling features that let the programmer detect and react to errors during execution. The major features are:

- On-units, which let the programmer specify what action is to be taken when a particular condition occurs
- The condition built-in functions, which give the programmer more information about the source of the error

The program can access various return codes, such as that from the last invoked external procedure, built-in routine, or DPPX/SP function. Return codes may be accessed by means of the external variable PLIRC.

The FLOW option on the PUT statement prints a trace table of the most recent points in the program where control has been transferred.

The SNAP option on the PUT statement prints the active procedures, begin-blocks, and on-units.

With the CALL PLIDUMP statement, the user can print a diagnostic dump. This dump gives more information on PL/I control blocks and storage contents.

The REPORT option on the PLI.CALL or PLI.SETOPT command gives such information as the storage used and the library routines called. (See "Run-Time Options" on page 187.)

Run-Time Debugging Aids — Interactive Processing

The programmer can test a program and check and change the values of variables interactively with the DPPX PL/I Execution Debug Facility. This facility is invoked by the DEBUG option, which the programmer usually specifies on the PLI.CALL statement or sets (and resets) in the load module by means of the PLI.SETOPT command. The Execution Debug Facility can be used with all kinds of program invocation, including invocation as a transaction program. No change is made to the program, so it need not be recompiled.

The DPPX PL/I Execution Debug Facility presents a panel through which the programmer can monitor the execution of the program. In this panel the programmer may specify either points where processing is to pause or whether a trace is to be written to a separate data set, or both. When one of these pause points is reached, the Execution Debug Facility displays program-related information on the panel. This information includes:

- The name of the currently active block
- The current statement number
- The offset address of the current statement
- A label that identifies the current point of control
- The condition that caused the program to pause

The programmer may then enter commands to:

- Display more information, such as the current values of variables
- Change the values of variables
- Specify new points where processing is to pause
- End processing

Variables can be accessed by means of their symbolic name.

The DPPX PL/I Execution Debug Facility can also help improve the performance of a program. The programmer may use the trace information to find where and how frequently library routines are called, for example, and then decide to modify parts of the program and/or tailor the shared nucleus library accordingly.

Industry Standards

DPPX PL/I is designed according to the specifications of these industry standards as understood and interpreted by IBM as of January 1981:

- American National Standard (ANS) PL/I, X3.53-1976, which is technically identical to ISO 6160-1979 (International Organization for Standardization) and ECMA-50 (1976) (European Computer Manufacturers Association)
- American National Standard PL/I General-Purpose Subset, ANSI X3.74-1981

For the language features supported, see the following sections.

OS PL/I Comparison

A large common set of PL/I language elements is accepted and processed consistently by the DPPX PL/I Compiler and the OS PL/I compilers. Source programs written for the OS PL/I compilers will in most cases need some revision to compile and execute successfully on DPPX PL/I. The following are the main areas of difference between OS PL/I and DPPX PL/I:

- Some common language elements are implemented differently. DPPX PL/I follows the rules of ANSI PL/I, which differ in some areas from OS PL/I. For example, in DPPX PL/I:
  - All variables must have at least one type attribute declared. The default attributes are BINARY and FIXED and do not depend on the initial character of the name.
  - In edit-directed input, an all-blanks field yields zero, as does an all-blanks string in character-to-arithmetic conversion. OS PL/I raises the CONVERSION condition in both cases.
  - In passing an array as an argument with adjustable bounds, the corresponding bounds of the parameter must be defined by asterisks. OS PL/I allows also any fixed bounds to be declared for the parameter.
- DPPX PL/I does not contain all the language elements available in the OS PL/I Optimizing Compiler, for instance:
  - Tasking
  - List- and data-directed input/output
  - COMPLEX data type
  - CONTROLLED and DEFINED storage classes.

- Differences exist in implementation-defined limits. For example:
  - The maximum length of an external identifier is 8 (OS PL/I Optimizing Compiler: 7).
  - The maximum number of arguments in a CALL statement is 32 (OS PL/I Optimizing Compiler: 64).
- DPPX PL/I provides access to DPPX/SP, associated program products, and to certain DPPX/SP facilities in a system-dependent manner.
- Differences exist in the ANS implementation-defined items (for instance, ENVIRONMENT option).

Data mapping of all data types supported by DPPX PL/I is identical to that in OS/VS PL/I and DOS/VS PL/I.

The DPPX PL/I Language

The main features of the DPPX PL/I language include the following:

- DPPX PL/I offers a large selection of American National Standard (ANSI) PL/I language elements.
- Structured programming is supported by DO loops, DO groups, IF-THEN-ELSE clauses, block structure, and function and subroutine procedures.
- Program modularity is provided by the PL/I block structure and scope rules.
- Extensive input/output capabilities are provided.
- STATIC, AUTOMATIC, and BASED storage class attributes are supported, giving the programmer control over storage usage.
- Easy data manipulation is provided by a comprehensive set of operators and built-in functions.
- Program checkout facilities are available, including on-units and condition built-in functions.
- Separately written PL/I code can be included from a PL/I source statement library into the source program at compile time (%INCLUDE statement).
- Non-PL/I external subroutines can be invoked from DPPX PL/I.

The following sections discuss data types, data organization, storage control, data manipulation, program organization and flow of control, input/output capabilities, and DPPX PL/I extensions.

Data Types

The types of data that may be used in a DPPX PL/I program fall into these categories:

- Problem data, which is used to represent values to be processed by computational operations. It consists of:
  - Arithmetic data, which can be represented in either binary or decimal base and can be either fixed or floating-point. Arithmetic data can also be numeric character data, declared with the PICTURE attribute.
  - String data, which can be either bit or character strings.
- Program control data, which the programmer uses to control the execution of the program. DPPX PL/I supports four types of program control data: entry, file, label, and pointer.

DPPX PL/I also lets programmers build and manipulate chained data lists, using the POINTER data type and the BASED storage class.

Data Organization

Data items may be single data elements, or they may be grouped together to form data aggregates called arrays and structures. Any type of problem data or program control data supported by DPPX PL/I can be collected into arrays or structures. An array element can also be a structure.

Storage Control

All variables, problem data, and program control data require storage. Each variable has associated with it a storage class, declared explicitly or by context. DPPX PL/I supports these storage classes:

- Static—where storage is allocated at the start of an external procedure and remains allocated until the program terminates.
- Automatic—where storage is allocated upon entry to the block in which the automatic variable has been declared. The storage is released on exit from the block that allocated it. It can be reallocated many times during execution of a program. The extents of arrays and the length of strings can be defined dynamically.
- Based—which is fundamentally different from the static and automatic classes. A based variable describes the storage of another variable, whose location is identified by a pointer. The based variable has no storage of its own.

Data Manipulation

DPPX PL/I supports all major PL/I data types, statements, and operators. Of particular interest are:

- Expressions can contain all PL/I infix and prefix operators and the pointer qualifier.

- Expressions can be in parentheses to specify the order of evaluation and may contain function references and fully or partly qualified and/or subscripted variables.

- All common conversions within expressions and all conversions on scalar assignment are allowed.

- Many types of built-in functions are provided: arithmetic, mathematical, string-handling, array-handling, condition-handling, storage control, input/output, date, and time.

  UNSPEC and SUBSTR can also be used as pseudovariables.

- Aggregates can be assigned if they match in size, shape (structuring or array dimensions), and component data types. Both aggregates must have connected storage.

Program Organization and Flow of Control

A PL/I program consists of one or more external procedures, each of which may contain other procedures and/or begin blocks. Each procedure begins with a PROCEDURE statement preceded by a label, and ends with an END statement.

Procedures can be invoked by a CALL statement or a function reference. Function references can be used wherever an expression is possible, and allow a value to be computed and returned to the point of invocation. Control can be returned to the point of invocation by a RETURN statement.

A begin block must be contained in a procedure and need not have a label. It is headed by a BEGIN statement and ends with an END statement. Begin blocks are executed in the normal sequence of control governing statement execution.

The STOP statement terminates execution of a PL/I program.

DPPX PL/I supports the IF-THEN-ELSE clause, the DO, iterative DO, and DO WHILE statements, and the GOTO statement.

The ON statement allows condition-handling and supports the ERROR condition and the I/O conditions UNDEFINEDFILE, ENDFILE, KEY, and ENDPAGE.

The condition-handling built-in functions ONFILE, ONKEY, and ONCODE will help further to determine the nature of the condition.

The SIGNAL statement may be used to raise a condition, such as simulating interrupts for debugging purposes.

Input/Output Capabilities

DPPX PL/I supports both stream- and record-oriented data transmission. DPPX/SP data bases are supported by record-oriented data transmission. DPPX/SP supports only fixed-length records.

Stream-Oriented Data Transmission: Stream-oriented data transmission statements get and put data with a minimum of programming effort, because formatting and conversion are done automatically by the Compiler and Library.

Stream-oriented data transmission is edit-directed. It provides these format items, which make it more versatile than record-oriented data transmission in its formatting of data:

- A for character string data
- B, B1, and B4 for bit string data
- E for floating-point data
- F for fixed-point data
- X for spacing
- COLUMN, LINE, PAGE, and SKIP control format items

Record-Oriented Data Transmission: Record-oriented data transmission does not do any conversion. It is therefore faster than stream-oriented data transmission and more versatile in the ways in which data can be processed and in the types of data sets it processes:

- SEQUENTIAL I/O—this facility is available through the READ, WRITE, and REWRITE statements.
- DIRECT I/O—this facility is available through the READ, WRITE, DELETE, and REWRITE statements with the KEY or KEYFROM option.

Locate-mode input is available through the READ statement with the SET option.

The READ statement with a search for KEY-GREATER-OR-EQUAL is supported.

DPPX PL/I Extensions

These extensions to DPPX PL/I support DPPX/SP facilities and program products:

- Ability to call subroutines that are not written in PL/I, such as DPPX/SP data base management and transaction processing services
- Built-in subroutines to support data formatting and to issue DPPX/SP commands
- Testing and setting of return codes

DPPX PL/I Language

All PL/I keywords are in uppercase. Lowercase designates features supported without specific reference to keywords. DPPX PL/I support of a keyword or function does not imply support of all facilities associated with that keyword or function in ANSI PL/I.

The PL/I language includes:

- Statements and statement options
- Attributes and attribute options
- Built-in functions and pseudovariables
- Built-in subroutines
- Conditions
- Format items
- Compiler control statements

Statements and Statement Options

Assignment
 scalar
 aggregate
 array=element
BEGIN
CALL
CLOSE
 FILE
DECLARE
DELETE
DO
DO *iterate*
 BY
 TO
 WHILE
END
GET
 EDIT
  data list items
  format items
 FILE
 SKIP
GO TO *or* GOTO
IF
 THEN/ELSE
null statement
ON
OPEN
 FILE
 INPUT
 LINESIZE
 OUTPUT
 PAGESIZE
 TITLE
 UPDATE
PROCEDURE
 parameters
  scalar
  array
  structure
 OPTIONS
 RETURNS
PUT
 EDIT
  data list items
   expressions
  format items
 FILE
 FLOW
 LINE
 PAGE
 SKIP
 SNAP

Attributes and Attribute Options

ALIGNED
 BIT
AUTOMATIC
BASED
BINARY
BIT
BUILTIN
CHARACTER
constant
 arithmetic
 string
 ENTRY
 FILE
 LABEL
DECIMAL
dimension
DIRECT
ENTRY
 OPTIONS
ENVIRONMENT
 REGIONAL
 INDEXED
 RECSIZE
 KEYLENGTH
 KEYDISP
 EXCL/NEXCL
 SHARED/NSHARED
 CTLASA
EXTERNAL
FILE
FIXED
FLOAT
INITIAL
INPUT
INTERNAL
KEYED
LABEL
OUTPUT
parameter
PICTURE
POINTER
precision
PRINT
real
RECORD
RETURNS
SEQUENTIAL
STATIC
STREAM

Built-in Functions and Pseudovariables

Mathematical
 ACOS
 ASIN
 ATAN
 ATAND
 COS
 COSD
 EXP
 LOG
 LOG2
 LOG10
 SIN
 SIND
 SQRT
 TAN
 TAND
Arithmetic
 ABS
 BINARY
 DECIMAL
 DIVIDE
 FIXED
 FLOAT
 MAX
 MIN
 MOD
 ROUND
 SIGN
 TRUNC
String Handling
 BIT
 CHARACTER
 COPY
 INDEX
 LENGTH
 SUBSTR
 SUBSTR *pseudovariable*
 TRANSLATE
 UNSPEC
 UNSPEC *pseudovariable*
 VERIFY
Array Handling
 DIMENSION
 HBOUND
 LBOUND
Condition Handling
 ONCODE
 ONFILE
 ONKEY
Storage Control
 ADDR
 NULL

Built-in Subroutines

PLICMD
PLIDUMP
PLIPOPEN
PLIPCLSE
PLIPWRIT
PLIPREAD

Conditions

These conditions are enabled and may be specified in ON or SIGNAL statements:

ENDFILE
ENDPAGE
ERROR
KEY
UNDEFINEDFILE

These conditions are also enabled, but cannot be specified in ON or SIGNAL statements:

all other input/output conditions
conversion
fixedoverflow
overflow
record
stringsize
transmit
underflow
zerodivide

Format Items

A
B
B1
B4
COLUMN
E
F
PAGE
LINE
SKIP
X

Compiler Control Statements

*PROCESS
%INCLUDE
%PAGE
%SKIP

Statements and Statement Options	Attributes and Attribute Options	Built-in Functions and Pseudovariables
READ	*structure*	*Input/Output*
INTO	UNALIGNED	LINENO
FILE	FLOAT BINARY	SAMEKEY
SET	FIXED BINARY	*Other*
KEY	FLOAT DECIMAL	DATE
KEYTO	UPDATE	TIME
RETURN	VARIABLE	PLIHEX
REWRITE		
FROM		
FILE		
KEY		
SIGNAL		
STOP		
WRITE		
FROM		
FILE		
KEYFROM		

DPPX PL/I Library Summary

The DPPX PL/I library contains these functions and subroutines to support an executing PL/I program:

- Program management routines invoked by the compiled code to set up and terminate the execution environment, handle entry to and exit from PL/I blocks, allocate storage, and handle interrupts.
- Routines that are invoked to perform some of the more complicated PL/I functions, such as certain data conversions and input/output operations. Examples of these are conversion from character to decimal representation, opening files, and reading and writing data.
- Built-in functions, comprising functions for mathematical and arithmetic operations, storage control, array handling, string handling, condition handling, and the DATE, TIME, and PLIHEX functions. The PLIHEX built-in function converts character data to hexadecimal.
- Built-in subroutines to invoke DPPX/SP commands or Format Management services, or to produce a dump.
- PL/I Execution Debug Facility to test a program while it is running. (See "Run-Time Debugging Aids — Interactive Processing" on page 179 for details.)

The library routines are not link-edited with the user program. This reduces the load module size. Library routines that are used often by a program or shared by two or more programs will be link-edited together into a nucleus. When any of these routines are called, they can be accessed as quickly as possible. This nucleus would normally be placed in shared storage to allow shared use.

Any library routines not in this nucleus will be loaded if required. An IBM-supplied link-edit command list lets users tailor the nucleus to their needs. This tailoring can be based on tuning information obtained from the REPORT run-time option and the Execution Debug Facility.

Run-Time Options

Run-time options can be specified:

- In the OPTIONS attribute of the external procedure
- As keywords of the PLI.SETOPT command. Any execution option specified in this command permanently changes the corresponding option already established for the load module.
- As keywords of the PLI.CALL command when invoking the program.

The execution options that may be specified in a PROCEDURE statement, PLI.SETOPT command, or PLI.CALL command are:

- ISASIZE—specifies the amount of storage to be allocated for the initial storage area (ISA)
- STACKINCR—specifies the amount of storage to be allocated for follow-on segments
- LIBLIMIT—specifies a limit in bytes, which may not be exceeded by the total size of library modules loaded for this program
- REPORT or NOREPORT—specifies whether a report showing the usage of library modules and main storage is to be displayed when the program terminates
- FLOW or NOFLOW—specifies whether to obtain flow information for a program that was compiled with the FLOW compiler option

The execution options that may be specified in only a PLI.SETOPT or PLI.CALL command are:

- DEBUG or NODEBUG—specifies whether the DPPX PL/I Execution Debug Facility is to be used
- INFO or NOINFO—specifies whether messages are to be displayed whenever PL/I detects a situation that could affect the execution of the program
- SUPPRESS or NOSUPPRESS—specifies whether certain messages are to be suppressed

The run-time options that may be specified in only a PLI.CALL command are:

- INIT or NOINIT—specifies whether the run unit set up should be initialized
- CALL or NOCALL—specifies whether the initial procedure should be given control
- TERM or NOTERM—specifies whether the run unit should be terminated

Compiler Options

There are three types of Compiler options:

- Options that control the Compiler operation are: COMPILE, NOCOMPILE, DUMP, NODUMP, MARGINS, SIZE.

- Options that control the listing printed by the Compiler are: AGGR, NOAGGR, ATTR, NOATTR, FLAG, NOFLAG, LINECOUNT, LIST, NOLIST, MAP, NOMAP, OFFSET, NOOFFSET, OPTIONS, NOOPTIONS, SOURCE, NOSOURCE, TERMINAL, NOTERMINAL, XREF, NOXREF.

- Options that control the object module produced by the Compiler are: FLOW, NOFLOW, GOSTMT, NOGOSTMT, NAME, OBJECT, NOOBJECT, PROCUNIT(FLOAT), PROCUNIT(NOFLOAT).

Chapter 18. DPPX FORTRAN

FORTRAN is a mathematically oriented programming language, best suited to manipulating numerical data. It can help solve research and analytical problems in science, engineering, and business and is simple enough to be used by the nonprofessional programmer. It is the oldest high-level programming language, first developed in the mid-1950s. The name FORTRAN stands for FORmula TRANslator.

DPPX FORTRAN is a complete implementation of American National Standard (ANS) Basic FORTRAN (X3.10-1966), has most of the features of ANS FORTRAN (X3.9-1966), and has many features not contained in either standard.

DPPX FORTRAN is useful for coding general-purpose problem-solving programs.

DPPX FORTRAN consists of two program products:

- The DPPX FORTRAN Compiler (5760-FO1), which checks syntax and generates object code
- The DPPX FORTRAN Library (5760-LM1), which contains reentrant mathematical, data conversion, and bit manipulation functions; some miscellaneous service subroutines (test for divide check, obtain time of day, and so on); and routines for internal services (such as handling I/O)

The DPPX FORTRAN Compiler and DPPX FORTRAN Library do not have to be installed in the same 8100. A development 8100 would have both products to compile and test application programs. A production 8100 might have just the Library to run programs that were compiled and tested on the development 8100.

Using DPPX FORTRAN

Developing a DPPX FORTRAN program generally follows these steps:

1. Designing the program.
2. Coding the program.
3. Entering the source program into a data set using the DPPX/SP interactive editor.
4. Compiling the program, initiated interactively or in batch mode. IBM provides command lists to do this. Users can specify options to control the Compiler's operation. (The options are listed under "Debugging Aids" on page 191.)

   The Compiler produces two data sets: one contains an object module and the other a listing. The listing can contain:

   - The source program input
   - Errors the Compiler discovered

- Names and relative locations of common blocks, FORMAT statements, variables, arrays, equivalenced items, and called subprograms
- The relative address of each statement with respect to the start of the program
- The Compiler options in effect
- Alphabetized cross references of all variables and arrays

5. Link-editing the program. IBM provides command lists to do this. Link-editing forms an executable load module from the object module input. It brings into the program any Library routines required by the program, such as intrinsic function routines and routines to handle I/O. The linkage editor can also combine subprograms of a multiprogram run unit, edit subprograms out of a load module, and set up an overlay structure.

   Instead of using the linkage editor to include Library routines and subprograms of a multiprogram run unit in the same load module, users can leave them to be automatically loaded at execution time as they are needed.

6. Executing the program.

7. Debugging the program. Debugging aids are discussed below.

Reentrant Code

Programs can be made reentrant. The DPPX FORTRAN Compiler can produce reentrant object code, and all the routines in the DPPX FORTRAN Library are reentrant.

Reentrancy saves main storage when a program is to be run concurrently by more than one user. A single reentrant copy of the program, rather than a copy for each user, resides in main storage during program execution.

Real or Simulated Floating-Point

The programmer can direct the DPPX FORTRAN Compiler to produce either real or simulated floating-point machine instructions. Simulated floating-point instructions allow floating-point operations on processors that do not have the floating-point feature.

Application-to-Application Communication

Two user-written DPPX FORTRAN application programs can communicate with each other. The programs can reside in the same or different 8100 processors. After certain configuration commands are issued, READ and WRITE statements within the DPPX FORTRAN programs control the sending and receiving of data between the programs.

Command Invocation from within a Program

Through a call to a DPPX FORTRAN Library routine, DPPX/SP or user-written commands can be issued from within a DPPX FORTRAN program.

Debugging Aids

To help locate errors in their programs, programmers can put DPPX FORTRAN debug statements into their source programs to:

- Trace the order of statement execution
- Trace the order in which subprograms are called
- Display or print the values of variables, array elements, and arrays at specified points in a program's processing

Calls to Non-FORTRAN Programs

A DPPX FORTRAN program cannot directly call a program coded in another high-level language. But a DPPX FORTRAN program can call a DPPX Assembler language program, which in turn can call a program coded in another high-level language.

Multiple Compilations

The DPPX FORTRAN Compiler allows compilation of more than one FORTRAN source program with one invocation of the Compiler. This reduces job stream and Compiler processing overhead.

Comparison with ANS FORTRAN

DPPX FORTRAN is designed according to the specifications of American National Standard Basic FORTRAN, X3.10-1966, as understood and interpreted by IBM as of July 1978.

In addition, DPPX FORTRAN has most of the features in ANS FORTRAN, X3.9-1966. The features it does not have are:

- Complex data type
- Adjustable (object-time) dimensions
- Reading of formats at execution
- G edit descriptor in FORMAT statement
- Two levels of nesting in FORMAT statement
- Format specification in an array
- These intrinsic functions: AIMAG, CABS, CCOS, CLOG, CMPLX, CONJG, CSIN, CSQRT, CEXP

But DPPX FORTRAN has these features not in ANS FORTRAN, X3.9-1966:

- Direct access input/output statements—READ, WRITE, DEFINE FILE, FIND
- PROGRAM statement
- ENTRY statement
- IMPLICIT statement
- Debug statements—DEBUG, AT, TRACE ON, TRACE OFF
- Character constants (literal enclosed in apostrophes)
- Hollerith constants (literal preceded by H)
- Hexadecimal constants in DATA statement
- Mixed-type expressions
- List-directed input/output
- Arrays of up to seven dimensions

- Multiple exponentiation in an expression
- 2-byte integer intrinsic functions and arguments (in addition to 4-byte functions and arguments)
- Character constant in PAUSE statement
- Alternate return in RETURN statement
- Length specification in INTEGER (INTEGER*2, INTEGER*4), REAL (REAL*4, REAL*8), and LOGICAL (LOGICAL*4) statements
- Z (hexadecimal) edit descriptor in FORMAT statement
- ERR and END parameters in READ statement
- ERR parameter in WRITE statement
- Character constant as actual argument in CALL statement or function reference
- Apostrophe (') edit descriptor in FORMAT statement
- Initial data values in explicit specification statements
- Generalized subscripts
- Name can start with currency symbol ($),
- ' (apostrophe), and & (ampersand) characters
- These intrinsic functions: ACOS, ANINT, ASIN, BTEST, COSH, DACOS, DASIN, DCOSH, DDIM, DFLOAT, DINT, DNINT, DPROD, DSINH, DTAN, DTANH, IAND, IBCLR, IBSET, IDNINT, IEOR, IOR, ISHFT NINT, NOT, SINH, TAN
- These service subroutines: DATE, DVCHK, FETCH, OVERFL, SYSCMD, SYSRC, TIME

Comparison with System/370 FORTRAN

DPPX FORTRAN does not have the following features in System/370 FORTRAN (implemented by the FORTRAN IV [G1] and FORTRAN IV [H Extended] compilers):

- Complex data type
- GENERIC statement
- NAMELIST statement
- PRINT statement
- PUNCH statement
- Asynchronous I/O (not in G1)
- Extended list-directed I/O (not in G1)
- Adjustable (object-time) dimensions
- Extended precision and Q edit descriptor
- G edit descriptor
- Two levels of nesting in FORMAT statement
- Reading of formats at execution
- Length specification of 1 in LOGICAL statement (LOGICAL*1)
- Format specification in arrays
- Subprogram names that begin with & (ampersand)
- These intrinsic functions (primarily complex and extended-precision functions): AIMAG, ALGAMMA, ARCOS, ARSIN, CABS, CCOS, CDABS, CDCOS, CDEXP, CDLOG, CDSIN, CDSQRT, CEXP, CLOG, CMPLX, CONJG, COTAN, CQABS, CQCOS, CQESP, CQLOG, CQSIN, CQSQRT, CSIN, CSQRT, DARCOS, DARSIN, DBLEQ, DCMPLX, DCONJG, DCOTAN, DERF, DERFC, DGAMMA, DIMAG, DINT (not in G1), DLGAMMA, DREAL, ERF, ERFC, GAMMA, HFIX, IQINT (not in G1), LGAMMA, QABS, QARCOS, QARSIN, QATAN, QATAN2, QCMPLX, QCONJG, QCOS, QCOSH, QCOTAN, QDIM, QERF,

QERFC, QEXP, QEXT, QEXTD, QFLOAT, QIMAG, QINT, QLOG, QMAX1, QMIN1, QMOD, QREAL, QSIGN, QSIN, QSINH, QSQRT, QTAN, QTANH, SNGLQ
- Generic function selection for these intrinsic functions: ABS, ACOS, AINT, ASIN, ATAN, ATAN2, CMPLX, CONJG, COSH, COTAN, CSIN, DBLE, DIM, ERF, ERFC, EXP, GAMMA, IMAG, INT, LGAMMA, LOG, LOG10, MAX, MIN, MOD, QEXT, REAL, SIGN, SIN, SINH, SNGL, SQRT, TAN, TANH
- These service subroutines: DUMP ERRMON (not in G1), ERRSAV (not in G1), ERRSET (not in G1), ERRSTR (not in G1), ERRTRA (not in G1), EXIT, PDUMP, SLITE, SLITET

But DPPX FORTRAN has these features not in System/370 Fortran (implemented by the FORTRAN IV [G1] and FORTRAN IV [H Extended] compilers):

- PROGRAM statement
- 2-byte integer intrinsic functions and arguments (in addition to the support of 4-byte functions and arguments)
- ERR parameter in WRITE statement
- These intrinsic functions: ANINT, BTEST, DNINT, DPROD, IAND, IBCLR, IBSET, IDNINT, IEOR, IOR, ISHFT NINT, NOT
- These service subroutines: DATE, FETCH, TIME, SYSCMD, SYSRC

The major differences outside of language content are:

- DPPX FORTRAN does not have a symbolic interactive debugging tool.
- The DPPX FORTRAN Compiler can generate reentrant code. Also, all the routines in the DPPX FORTRAN Library are reentrant.
- In DPPX/SP, subprograms and library routines can be excluded from a link-edit so that they are dynamically loaded when needed during execution.
- In DPPX/SP, segments of an overlay program are not automatically "fetched" into main storage when needed; there is no overlay supervisor. Programmers must code CALL FETCH statements in the source program to fetch overlay segments during execution.
- The DPPX FORTRAN Compiler has fewer options.

DPPX FORTRAN Language Summary

The DPPX FORTRAN character set consists of:

- Letters—A to Z
- Digits—0 to 9
- Special characters—

  (space character)
  = (equals)
  + (plus)
  - minus or hyphen)
  * (asterisk)
  / (slash)
  ((left parenthesis)
) (right parenthesis)
  , (comma)

. (decimal point or period)
' (apostrophe)
& (ampersand)
$ (currency symbol; this character has the same significance as a letter)

The elements of the DPPX FORTRAN language are:

- Constants—integer, real (single- or double-precision), logical, character, Hollerith, and hexadecimal
- Variables—integer, real (single- or double-precision), and logical
- Arrays—integer, real (single- or double-precision), and logical
- Expressions—arithmetic, relational, and logical
- Statements (which follow)

The statements in the DPPX FORTRAN language are:

- Main program and subprogram statements

  PROGRAM
  FUNCTION
  SUBROUTINE
  BLOCK DATA
  ENTRY
  END

- Specification statements

  DIMENSION
  EQUIVALENCE
  COMMON
  EXTERNAL
  IMPLICIT
  Type:
    INTEGER
    REAL
    DOUBLE PRECISION
    LOGICAL

- DATA statement
- Statement function definition statement
- Assignment statements

  Arithmetic assignment
  Logical assignment
  ASSIGN

- Control statements

  GO TO (unconditional)
  GO TO (computed)
  GO TO (assigned)
  IF (arithmetic)
  IF (logical)
  DO
  CONTINUE
  STOP
  PAUSE
  CALL
  RETURN

- Input/output statements

  Sequential
  READ (formatted)
  READ (unformatted)
  READ (list-directed)
  WRITE (formatted)
  WRITE (unformatted)
  WRITE (list-directed)
  BACKSPACE
  REWIND
  ENDFILE
  Direct-access
  READ
  WRITE
  DEFINE FILE
  FIND
  Used with sequential and direct-access
  FORMAT

- Debug statements

  DEBUG
  AT
  TRACE ON
  TRACE OFF

DPPX FORTRAN Compiler Options

Compiler options are:

- Options that affect Compiler operation:

  – SIZE—specifies the amount of main storage to be made available to the Compiler for its processing

  – OBJECT/NOOBJECT—specifies whether the compiler should produce an object module after it checks syntax

- Options that affect the listing printed by the Compiler:
  - SOURCE/NOSOURCE—specifies whether the Compiler is to print the source program statements and their associated internal statement numbers
  - OFFSET/NOOFFSET—specifies whether the Compiler is to print the relative main storage address for each executable source language statement
  - MAP/NOMAP—specifies whether the compiler is to print the names and relative locations of all data items in the program
  - XREF/NOXREF—specifies whether the Compiler is to print all variables and statement labels and the internal statement number or numbers where each is defined and used
  - LINECOUNT—specifies the maximum number of lines to be printed on a Compiler output listing page
- Options that affect the object module produced by the Compiler:
  - FLOAT/NOFLOAT—specifies whether the Compiler is to generate floating-point machine instructions for arithmetic operations performed with real numbers, or instructions that simulate floating-point operations
  - RENT/NORENT—specifies whether the Compiler is to generate a reentrant object module
- Option for sending compilation messages to the terminal:
  - TERMINAL/NOTERMINAL—specifies whether error messages generated by the Compiler are to be displayed at the user's terminal

Summary of Intrinsic Functions

Figure 21 summarizes the intrinsic functions in the DPPX FORTRAN Library.

General Type of Function	Specific Function	Function Name
Logarithmic	Natural logarithm Common logarithm	ALOG, DLOG ALOG10, DLOG10
Trigonometric	Sine Cosine *Tangent* *Arc sine* *Arc cosine* Arc tangent	SIN, DSIN COS, DCOS *TAN, DTAN* *ASIN, DASIN* *ACOS, DACOS* ATAN, DATAN, ATAN2, DATAN2
Hyperbolic	*Sine* *Cosine* Tangent	*SINH, DSINH* *COSH, DCOSH* TANH, DTANH
Miscellaneous mathematical	Exponential Square root Truncation *Nearest whole number* *Nearest integer number* Absolute value Remaindering Transfer of sign Positive difference *Double precision product* Choosing largest value Choosing smallest value	EXP, DEXP SQRT, DSQRT AINT, *DINT* *ANINT, DNINT* *NINT, IDNINT* IABS, ABS, DABS MOD, AMOD, DMOD ISIGN, SIGN, DSIGN IDIM, DIM, *DDIM* *DPROD* MAX0, AMAX1, DMAX1, AMAX0, MAX1 MIN0, AMIN1, DMIN1, AMIN0, MIN1
Type conversion	Convert to integer Convert to real Convert to double	INT, IFIX, IDINT REAL, FLOAT, SNGL DBLE, *DFLOAT*
Bit manipulation	*Logical product* *Inclusive OR* *Exclusive OR* *Logical complement* *Shift* *Bit test* *Bit set* *Bit clear*	*IAND* *IOR* *IEOR* *NOT* *ISHFT* *BTEST* *IBSET* *IBCLR*

Figure 21. DPPX FORTRAN Library Functions. Functions Not in ANS FORTRAN, X3.9-1966, Are in Italic Type

Chapter 19. DPPX Assembler

DPPX Assembler is a program product (5760-AS1) that translates source programs written in DPPX Assembler language into 8100 machine language. It also allows users to code macro instructions. The DPPX Assembler translates these user-written macros, as well as IBM-provided macros, into machine language.

Coding conventions are similar to those of System/370 DOS/VS Assembler (although the instruction set is not compatible with that of any System/370 Assembler).

Useful for System Programming

DPPX Assembler language is best used for writing original system code, replacing portions of IBM code, and writing specialized interface routines such as to support devices not supported by IBM.

For writing application programs, the Cross System Product/Application Development for DPPX/SP program product or a high-level language (such as DPPX COBOL) is more convenient to use.

Large Instruction Set

DPPX Assembler language has three types of instructions: machine, assembler, and macro.

Machine instructions are requests to the hardware. There are 112 machine instructions that provide arithmetic, comparisons, branches and jumps, shifting, logic (AND, OR, test and set, and so forth), and general control (store, load, and so forth).

Assembler instructions are requests to the DPPX Assembler. There are 36 assembler instructions. CSECT, EQU, DC, and DROP are a few.

Macro instructions are predefined series of instructions. There are two categories of macro instructions: IBM-provided and user-written.

- The IBM-provided macros are part of this DPPX program product. There are about 50 macros to invoke input/output (SEND, RECEIVE, CONNECT), supervisor (GETMAIN, FREEMAIN, FETCH), and other services.
- User-written macros may have as many as 50 symbolic parameters on a macro type and as many as 31 character positions in a macro instruction operand.

Interactive or Batch Assembly

Users can invoke the DPPX Assembler for interactive or batch execution.

Input is a source program in a data set and the name of a user macro library.

Output is two data sets: one containing an object module to be link-edited and one containing a listing. The listing contains an external symbol dictionary, source program statements, relocation dictionary, symbol cross-reference listing, error messages, and summary statistics.

Interactive or Batch Macro Definition

Users can invoke the DPPX Assembler for interactive or batch definition of user-coded macro instructions.

Input is a source macro definition in a data set.

Output is two data sets: one a macro library containing the newly encoded macro and one containing a listing.

Macros can be decoded from a macro library back into source form.

Reentrancy

Programmers can code reentrant Assembler language programs. A reentrant program can be executed concurrently by two or more users, saving processor storage.

Chapter 20. DPPX APL

APL is a programming language suited to interactive problem solving and application development. It has an extremely compact way of expressing problems and solutions. The name APL stands for A Programming Language.

APL lets business people, scientists, engineers, educators, and other professional people use the computer for:

- Business planning
- Management decision making
- Information gathering and analysis
- Solving engineering and scientific problems
- Educational purposes

The APL language includes common arithmetic operations plus many additional functions for manipulating data. APL handles single items as well as information in lists and in tables. This lets people work with information in the way with which they are most comfortable.

DPPX APL is a program product (5760-XR2) that brings the APL language to the DPPX/SP user and supports many DPPX/SP services, including data base management services and format management services.

DPPX APL can be used by different types of people. Programmers can use it to develop and run DPPX/SP applications. And people who are not data processing professionals can use it to write programs for their own use.

The DPPX APL user can:

- Write and run APL programs from informal programs to full-size interactive application programs using data bases. (However, transaction programs cannot be written in DPPX APL.)
- Use DPPX/SP commands
- Access data stored in DPPX/SP data sets and data bases.
- Use all format management services of DPPX/SP and services of the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product.
- Send workspaces to VS APL and receive workspaces from VS APL.

The DPPX APL language is compatible with VS APL, with a few exceptions. DPPX APL has extensions for system interfaces, such as issuing DPPX/SP commands from within the APL program.

Even those who have never used a computer before will find the APL language easy to work with. Newcomers to APL can start by learning a small but powerful subset of the language, letting them write programs of surprising function.

The language includes the normal alphabet, the numbers 0–9, and a set of symbolic characters. There are symbols for the arithmetic operations (+, −, x, and .), for comparisons (>, <, and =), and for many other operations. There are over 40 of these *function symbols*. Users learn them as they work with APL. For simple calculations, they only need to know and use a few.

When writing programs in APL, users must enter the special APL symbols. The display terminal has a special keyboard, which includes the numerals and alphabetic characters in their standard positions as well as the APL symbols.

APL Is Conversational

To use APL users sit at display terminals and carry on conversations with APL. They enter APL statements at the terminal and receive replies from APL as they proceed.

Each APL statement is an instruction telling APL what it should do. A user can, for example:

- Enter data into the computer
- Indicate an arithmetic operation to perform
- Tell what output data is to be displayed on the screen

APL monitors each step a user takes, and the user sees the results of programming statements as they are performed by the computer. APL also lets users know if they make any mistakes in using the language.

Two Modes of Operation

Users can work in either of two modes:

- *Execution* (or *desk calculator*) *mode*, in which any statement a user enters is acted upon immediately and the answer displayed.
- *Definition mode*, in which a user defines a function consisting of a sequence of APL statements, to be performed later.

An example of using execution mode is when a user enters an arithmetic expression, and APL responds by displaying the result.

Definition mode is for entering APL statements to be saved and used later. For example, a user could type the statements needed to solve a specific part of a problem. A user groups these statements together into a *user-defined function* and gives it a name. To use this function later, the user types its name.

The APL Workspace

APL users have their own working area in the 8100. This working area, called a *workspace*, is a user's private section of the computer in which to experiment, calculate, and run programs.

A user can save a workspace for later use by storing it in a library: either a private library or a public one available to all users. (DPPX APL stores workspaces as data sets.)

A Language Users Can Grow With

APL is a good language to experiment with. Nothing a user does from the terminal can damage the 8100. The more users experiment, the more they will learn about APL.

First-time APL users will probably want to restrict their programming to simple tasks. They may enter cost and sales information to determine the profit margins for a group of products. Or, they might want to enter experimental data and perform simple calculations on it.

Simple tasks such as these can be easily performed using APL after as little as a few hours' practice.

As users become familiar with the language, they will probably start writing their own small user-defined functions to do more complicated work. They may also discuss their programs with experienced programmers, who may provide them with user-defined functions to help in their work.

Eventually, users can become able to define functions that will let them perform extremely complex work. They might analyze business data and make projections about the outcomes for possible future actions. Or they might perform advanced analyses of experimental data and present the results graphically.

An Example of APL Use

Some of the basic principles of APL are shown in the following example. You can see how APL performs simple calculations on data that you enter, including data in table form. The data in this example is sales data used to calculate sales results.

Subtraction

Suppose you type the sales per quarter for product A, calling it SALESA.

SALESA ←35 65 80 150

The values 35, 65, 80, and 150 are now stored under the name

SALESA.

The left arrow, ←, is used to name the data.

To see the contents of SALESA, you would type SALESA:

SALESA

APL responds by typing its contents:

```
35 65 80 150
```

Your input is indented, while APL responses begin in the first position.

Now you could enter the sales data for products B and C:

```
SALESB←70 165 220 480
SALESC←775 1060 1235 1540
```

To find the difference in sales between product C and product A, you would type SALESC-SALESA. APL responds immediately with the result:

```
SALESC-SALESA  740 995 1155 1390
```

You have now seen a simple APL function for *subtraction*.

Reduction

Another function, the *reduction* function, takes the symbol on its left (in this case, the plus function) and applies it to the data on its right. To find the total sales of product C, you could use plus reduction to add together all values of SALESC:

```
+/SALESC  4630
```

To perform comparisons, you can use the symbols > and < (greater-than and less-than). For example, you can compare sales quantities against the forecasted sales figures by using the greater-than symbol:

```
SALESC>1000 1100 1200 1500
0 0 1 1
```

The result is a 1 whenever SALESC is greater than the forecast, and a 0 otherwise. You can show how many quarters sales exceeded forecasts by adding the 1s with plus reduction +/.

```
    +/SALESC>1000 1100 1200 1500
2
```

To put all sales data in a single table, you would type:

```
SALES←3 4 ⍴ SALESA,SALESB,SALESC
```

This gives the table the name SALES and defines it to have 3 rows and 4 columns.

To display its contents, you would type the table's name:

```
      SALES
 35   65   80  150
 70  165  220  480
775 1060 1235 1540
```

All sales data is now together in a single table that can be referenced with one name. You can now apply APL functions to the entire table.

Tables

You use the same function symbols to work with tables in APL as with single numbers. A user can often process a row, a column, or an entire table with a single statement, so there is no need to repeat statements or loop as in other programming languages. Even groups of tables, such as sales figure tables for different years, can be handled in this simple way.

For example, to find the total sales for each product in the table, you would use plus reduction as you did for SALESC above:

```
+/SALES  330 935 4630
```

To find the total number of products sold per quarter, you would use plus reduction again, specifying the first dimension of the table:

```
+⌿SALES  880 1310 1535 2170
```

To find the total sales for the year, you would use plus reduction of all items in the table, indicated by ,SALES:

```
+/,SALES  5895
```

Now you have seen some of the basic principles of how APL works. You have seen how to:

- Store data in the 8100
- Perform calculations on lists of data
- Store data in a table
- Perform calculations on the entire table

The above examples showed some of the APL function symbols. APL function symbols provide many possibilities for manipulating data, making APL an extremely powerful programming language.

System Commands

DPPX APL has several commands that:

- Control workspaces
- Request reports on the contents of workspaces and libraries
- End an APL session

These *system commands* always start with a right parenthesis,).

For example, the system command)LOAD retrieves a saved workspace from a library. The following example shows how to retrieve a workspace named FINANCE from a private library:

```
)LOAD FINANCE  SAVED 11:15:27 06/24/83
```

The APL response shows when the user last saved the workspace. The workspace FINANCE is now loaded into the user's workspace and the user may start using its contents.

Support Functions

DPPX APL includes a number of support functions. These APL functions provide access to such DPPX/SP resources as data sets and terminals, and to such DPPX/SP services as data formatting.

The support functions are in public workspaces. These workspaces are listed under "Summary of Support Workspaces" on page 208.

Issuing DPPX/SP Commands

DPPX/SP commands may be issued within an APL program. To issue a DPPX/SP command, the programmer uses a support function. For example, to issue a DPPX/SP ASSOCIATE command you would specify:

```
RETCODE ←DPPXCR 'ASSOCIATE.RSDS ABC.DEF CNAME(INPUT)'
```

The return code returned by the DPPX/SP command is available in the variable RETCODE.

Accessing Data Sets and Data Bases

APL programs can work with data that has been created by other programs and stored in data sets or data bases. The programs that stored the data do not have to be APL programs.

APL programs can read in this data, use it, add to it, or change it.

APL programs can also create data and store it in data sets or data bases, where any program can access it. The DPPX APL programmer can access:

- All DPPX/SP resources, including data sets, terminals, and applications, using sequential access
- Data sets, using direct access
- Data sets containing data in internal APL format, using sequential or direct access
- Data sets and data bases, using indexed access

Since APL is an interpretive language, which does not produce executable load modules, APL programs cannot be used as transaction processors under DPPX/SP.

Using DPPX/SP for Display Terminal Communication

The DPPX APL programmer can use the format management services of DPPX/SP. This lets an APL program display information at any desired position on the terminal screen and pick up information entered by the terminal user at any predefined position on the screen. A program can, for example, display forms on the screen to be filled in by the user. This means of communicating with the user contrasts with the serial, line-by-line communication shown previously.

Auxiliary Processor and External Programs

APL provides a more general means of issuing DPPX/SP commands and performing input/output, in addition to the support functions described above. An *auxiliary processor* and DPPX APL external programs provide access.

The auxiliary processor passes shared variables from the APL program to designated external programs. This forms the communication link to programs outside the workspace.

An APL program may use shared variables to communicate with designated external programs, as well as with other DPPX APL sessions.

External non-APL programs are provided for:

- Issuing DPPX/SP commands
- Performing input/output operations to data bases and other DPPX/SP resources
- Working with the formatting services of DPPX/SP

Experienced programmers can write APL functions that communicate with external programs. Examples of such functions are the support functions.

If the provided programs do not meet your installation's needs, the system programmer can write DPPX/SP programs as required. All DPPX/SP programs that use standard DPPX/SP linkage conventions can be accessed.

VS APL Comparison

DPPX APL incorporates the language features available in VS APL Release 4.0. Features that are unique to DPPX APL are:

- The system variable $\square CMD$ lets the user execute APL system commands in user-defined functions. For example:

  $\square CMD \leftarrow$ *'COPY STAT A B C'*

  causes objects *A, B,* and *C* to be copied from workspace *STAT* into the active workspace.

- The workspace size is approximately 60,000 bytes. The user may write programs that use any number of workspaces. To use an additional workspace, the user program starts an additional APL session in the main

session, using a support function. The programs in the separate workspaces can communicate with each other using shared variables or support functions.

- An APL library of workspaces is created as a DPPX/SP data set. The user can move a library between 8100 systems, as a normal DPPX object, using DPPX/SP commands.

- APL objects or workspaces may be sent between DPPX APL and VS APL on the host.

  The user can transfer APL objects and workspaces between DPPX APL and VS APL using functions in workspaces provided with DPPX APL.

- APL support functions for accessing data and DPPX/SP resources are similar to VS APL functions in the VSAMDATA and APLDATA distributed workspaces. DPPX APL also provides APL support functions for:

  - Issuing DPPX/SP commands
  - Accessing formatting services

- The auxiliary processor and the external programs let APL programs perform DPPX/SP-dependent operations through shared variables. These operations include:

  - Input/output operations for access to data, terminals, applications, and DPPX/SP services
  - The ability to issue DPPX/SP commands
  - The ability to access applications written in languages other than APL

  The user communicates with the auxiliary programs through shared variables.

See the appropriate DPPX APL publications for details of the DPPX APL language.

Summary of Support Workspaces

The support workspaces and their uses are:

DPPXDATA	Contains support functions for accessing DPPX/SP data sets and the data bases.
APLDATA	Contains support functions for storing and retrieving APL data in DPPX/SP data sets.
DPS	Contains support functions for using the Format Management services of DPPX/SP.
DPPXCMD	Contains support functions for using DPPX/SP commands.
SUBSESS	Contains support functions for starting and communicating with APL subsessions.

TRANSFER	Contains support functions to format APL objects transferred between VS APL and DPPX APL.
UTILITY	Contains support functions for data conversion.
SYGTRF	Contains support functions to format APL objects transferred between DPPX APL and VS APL. This is a VSAPL workspace.

System Requirements

Your system must have DPPX/SP installed to use DPPX APL.

To use their particular functions, the system must have these licensed products installed:

- DPPX/SP IMD, program number 5660-282, to define maps.
- Distributed Systems Executive (DSX), program number 5748-XXG, to transfer workspaces. (DSX must be in the host.)

Devices Supported

These terminals can be used as APL terminals. Users can use them to enter or display APL statements with APL characters. The APL feature must be installed on the terminal.

- 8775 Models 1, 2, 11, 12 Display Terminal
- 3276 Models 11, 12, 13, 14 Control Unit Display Station
- 3278 Models 1, 2, 3, 4 Display Station
- 3279 Models 2B, 3B Color Display Station (color is not supported)
- 3290 Information Panel

The following printer can be used to print output with APL characters. The APL feature must be installed on the printer.

- 3287 Models 1, 2 Printer

Chapter 21. Data Capture and Management System (DCMS) for DPPX

Data Capture and Management System (DCMS) for DPPX is a program product (5760-XR6) that allows bulk, full-screen data entry.

DCMS can be used in any application or industry that requires the input of large amounts of repetitive data in a batch mode. It differs from interactive terminal applications in that selection of next screen or type of operation is not necessary. It is simply input of predetermined data in a predetermined format, such as entry of information from life insurance application forms in the insurance industry.

Using DCMS

Before data entry operators can enter data using DCMS, *administrators* define input display maps and output record formats interactively from a keyboard-display terminal. They can also specify syntax checks, validation edits, and logical tests that a field or record must pass to be accepted. Additionally, they can specify tables of constants, dates, check digits, and other values that are automatically inserted into records.

Once this setup is accomplished, *data entry operators* can enter data. DCMS checks the data for proper field format, as specified during setup. Operators can correct errors immediately or mark any record containing errors or unusual conditions for later examination.

All or selected portions of the entered data can be verified by reentering it; the reentered data that does not match the original version results in a conflict that must be resolved. Operators can scan records already entered at any time to make insertions, deletions, or modifications.

DCMS can accumulate keystroke rates and other operator statistics.

DCMS Is Self-Teaching

Because DCMS uses questionnaire prompts, the process of defining input display maps and output record formats is self-teaching. Thus, administrators can learn how to use DCMS with minimal training and self-study.

Five Data Entry Operations

Five modes of operation correspond to these activities a data entry operator can perform:

- **Enter data.** An operator who initially enters data enters it into batches. DCMS checks and edits the data according to user-specified edits as soon as the operator presses ENTER. The operator may then correct any errors. Any data in error that cannot be resolved can be marked for examination later. An operator can also flag records to indicate some unusual condition.

- **Verify data.** An operator can check the accuracy of data already entered by reentering it. If any newly entered data does not match the previously entered data, DCMS informs the operator and the operator must resolve the conflict. The operator chooses the level of verification: one field per record or more than one field per record.

- **Scan data.** An authorized operator can scroll through a batch and examine and change any record. The operator may insert, delete, mark, flag, or modify records.

- **Correct data.** An operator can display marked records one at a time. The operator may correct a record or mark it again. Any modified data is edited according to user specifications.

- **Browse data.** An operator can examine a batch file in read-only mode.

Operator Statistics

DCMS can collect data entry operator productivity statistics and print them through a local terminal and printer (DCMS cannot print DCMS-generated statistics through Printer Sharing). These statistics include keystroke rates for the various modes of operation, the number of batches completed and saved, and error rates.

Checks and Edits Improve Accuracy

DCMS checks whether data entered into a field corresponds to that field's format definition.

Administrators can specify a wide variety of syntax checks, validation edits, and logical tests that a field or record must pass. These checks are made while the source document is being entered, eliminating the need for much of the user editing in application processing.

If desired, programmers can write exit routines to perform additional checks and edits tailored to specific applications.

The check and edit capabilities provided by DCMS are:

- Required field
- Minimum length field
- Alphameric
- Alphabetic
- Character (any EBCDIC characters)
- Numeric (signed or unsigned, aligned or unaligned)
- Blank detection
- Date check
- Time and date insertion
- Justify and fill (left or right)
- Automatic field duplication
- Automatic field skip (entry, verification)
- Field verification required
- Self-check (mod 10 or mod 11, verify or generate)
- Table value substitution
- Table validation (range checks and lists)
- Batch total accumulation and balancing

Data Capture in Source Department

DCMS allows data to be entered by the source department. Such data is usually more timely and accurate than data transported to a central site for entry. The people most familiar with the data are more likely to enter it correctly and find errors quickly.

Processing on 8100 or Host

DCMS may be used whether application processing will be done on an 8100 or on a host computer. For the latter case, Distributed Systems Executive (DSX) or DPPX/SP can be used to forward batches to the host.

Ease-of-Use Features

These ease-of-use features are provided:

- Records and maps may be used in multiple jobs for different functions.
- Additions or changes to existing specifications are entered interactively.
- By permitting the user to define data entry output record formats so that they may be immediately acceptable to an application processing program, the need for batch reorganization should be reduced.

There is a *help* facility. During job and data definition, users can request displays of one or more panels that describe each field on the definition screen, and instructions for completing that definition. Users can also obtain explanations of commands and error messages.

Data Security Features

DCMS has these features to help you control who accesses and processes data:

- Only authorized users can log on to DCMS.
- Each job is assigned an access code to restrict its use to properly authorized operators.
- Each batch is similarly protected.
- The modes of operation an operator is authorized to use can be restricted.

Programmers can write exit routines to provide installation-specific audit trails.

DCMS can checkpoint the system at prespecified intervals.

Chapter 22. DPPX Presentation Services for 3640 Terminals (PS3640)

DPPX Presentation Services for 3640 Terminals (PS3640) is a program product (5660-267) that helps programmers develop and run transaction programs for certain 3640 terminals. It helps with transaction invocation, data formatting, data editing, message assembly, message buffering, time-of-day display, time stamping, transaction time out, and other transaction functions for the following 3640 terminals: 3641, 3642, 3644, 3646, and 3647. (The other 3640 terminals—the 3643 and 3645—have a similar capability provided by the DPPX/SP and DPPX/SP Interactive Map Definition [DPPX/SP IMD] program products.)

DPPX PS3640 consists of two components:

- The Interactive Transaction Generator (ITG) feature is the development-time component. It issues prompts to guide a programmer at a display terminal in creating run-time formats (maps), which include prompts, edits, light control, and I/O control. ITG requires the DPPX/SP IMD program product and can be used from a terminal that DPPX/SP IMD supports.

- The Execution Manager (EM) is the run-time component. It controls the transfer of data between the application program and the terminal, based on the maps defined using the Interactive Transaction Generator.

Separation of Logic from Data Formatting

DPPX PS3640 separates a program's logic from data formatting information, making coding simpler.

For example, it is irrelevant to the logic of a program where a particular piece of information is entered in a transaction, and when or where the editing and prompting is performed to ensure validity of the entered information.

Accordingly, DPPX PS3640 isolates the formatting, editing, and message assembly required for each transaction into tables called *maps*.

Interactive Transaction Generation

Maps define the input and output locations of data and the characteristics of the data for each transaction. Before compiling their programs, programmers use the Interactive Transaction Generator to create maps.

Specifically, maps are used to:

- Describe the input/output medium type: display, magnetics, card, punched badge, encode/print, DI/DO terminal, control

- Describe the characteristics of the 3640 terminal: display size, short buzzer, light control, time of day

- Define variable field edits: range check, table check, length check, alphameric check, field assembly (magnetic menu card), left and right justification, zero and blank fill

- Associate variables in the application data area with fields in the transaction
- Allow user exits, such as for additional validation

The Interactive Transaction Generator presents a series of questionnaire panels in which the programmer enters the transaction information described above. The programmer orders individual transaction step maps within a transaction group to produce the complete transaction as seen by the 3640 terminal user. The programmer can see the appearance of the transaction layouts as the process proceeds and can make changes until the transactions are satisfactory.

The Interactive Transaction Generator provides these features to help programmers specify maps:

- Full-screen context editing for entry of 3641 display and 3642 printer formats
- Menu-driven processes with a help facility and defaults
- Compatibility with DPPX/SP interactive editor commands
- Restart capability—changes can be automatically recovered
- Library management and print utilities

As a result of map definition, DPPX PS3640 generates COBOL or Assembler language declarations that correspond to the layout descriptions stored in the map. These declarations can be retrieved from the appropriate source statement library when the application program is compiled. In COBOL, this is done with the COPY statement. This statement would also be used to include a DPPX PS3640-provided control structure for passing control information between DPPX PS3640 and the application program during execution. See "Benefits of DPPX PS3640" below for structure of PL/I declarations.

Run-Time Use of Maps

When the application program is run, the Execution Manager component of DPPX PS3640 controls the transfer of data between the 3640 terminal and the user's application program.

The Execution Manager supports these run-time facilities:

- Transaction assembly—used for assembly of data entered in one or several transaction steps into one application data structure. This shortens pathlength by assembling data more quickly than user applications.
- Variable field editing and data transformation—used to validate and edit fields entered on the 3640 terminals. Editing includes range checking, table checking, length checking, alphameric checking, left and right justification, and zero and blank filling.
- Field assembly—used for assembly of single-digit numbers from a magnetic menu card into a single field.
- Time-of-day broadcast to 3641 and 3647 terminals along with synchronization of time between these two terminals.
- Message buffering.

- Transaction control, such as prompting and enabling of input devices for carrying out the terminal user/application program dialog.
- Transaction invocation.
- Passing of user-entered data to the application program in the appropriate application data structure format.
- Feedback information provided through the control area.
- Encoding/printer support for the 3642.

The Execution Manager provides an application program interface to DPPX COBOL, DPPX PL/I, or DPPX Assembler language. And it provides user mapping exits to allow user-written programs to perform such processing as additional validation or transformation of entered data.

Maps Can Be Distributed

Maps created under the Interactive Transaction Generator at one 8100 can be sent for execution by the Execution Manager to other 8100s where the Interactive Transaction Generator is not installed.

Terminals for Transaction Execution

The Execution Manager component of DPPX PS3640 supports the following devices:

- 3641 Reporting Terminal
- 3642 Encoder Printer
- 3644 Automatic Data Unit
- 3646 Scanner Control Unit
- 3647 Time and Attendance Terminal

Data Security

DPPX PS3640 makes it possible to require users to enter identification codes and/or input media (badges or magnetic stripes) before any transactions are completed.

Benefits of DPPX PS3640

DPPX PS3640 is helpful because:

- It provides a standardized interface between the application program and the various terminal types.
- It can generate DPPX COBOL, DPPX PL/I, or DPPX Assembler language maps for for application data. For DPPX PL/I, you first create a DPPX COBOL map and then edit the structure and make several small changes, as explained in *DPPX PL/I Application Programming: Guide*, SH19-6083.
- It removes device and format dependence from the application program into maps. This simplifies application program modification, whether it be to fix existing code or to add new code (such as for new devices).
- It shields the programmer from complex data-stream communication.
- Transaction formats and flows are created interactively and therefore can be adjusted until the most usable formats and flows are found.

Chapter 23. DPPX Parameter Table Generation Facility for the IBM 3644 Automatic Data

DPPX Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit (GEN3644) is a program product (5760-ED1) that helps programmers efficiently customize the IBM 3644 Automatic Data Unit. Customization consists of defining I/O channel parameters, selecting 3644 functions, and specifying initial values of stored data items. It results in a parameter table that works with the 3644 microcode provided by IBM.

The IBM 3644 Automatic Data Unit

The IBM 3644 Automatic Data Unit (Figure 22) is one of the 3640 plant communication terminals. It is a customizable (programmable) terminal that attaches via a loop to an 8100. A variety of sensors, actuators, production equipment, and laboratory equipment can attach to it.

The 3644 collects data from these attached devices on either a cyclic or interrupt-driven basis. It either forwards the data directly to the 8100 or examines the data and reports to the 8100 only when specific conditions exist. The 3644 can also generate and send responses to attached devices without interacting with the 8100.

Figure 22. IBM 3644 Automatic Data Unit

Generating a Parameter Table

A programmer fills in the blanks on preprinted worksheets. At a terminal, the programmer enters the source statements from the worksheets into a data set using the DPPX/SP interactive editor.

The programmer then invokes DPPX GEN3644 by command. DPPX GEN3644 works much like a compiler. It edits the source data and, if it finds errors, notes them in a listing (data set). Errors must be corrected and the source data resubmitted. When there are no errors, DPPX GEN3644 translates the source data into a parameter table that can be loaded into the 3644 at any time for execution.

Parameter Table Distribution

A parameter table may, of course, be used in a 3644 that is attached to the 8100 that contains DPPX GEN3644. But it may also be used in a 3644 attached to a different 8100 (that doesn't contain DPPX GEN3644). The user can use the Distributed Systems Executive (DSX) program product to transfer parameter tables from one 8100 to another.

Compatibility

The input source statements (from the worksheets) and the output records of DPPX GEN3644 are compatible with the input and output of:

- The 3644 Translation Services program, one of the 3630 Plant Communication System host services programs
- The GEN3644 program product that operates on the IBM 4331 processor

Worksheets

The worksheets are distributed as reproduction masters. There are 14 worksheets, they come in a single package, and they are 8 1/2 by 11 inches in size. Customers are responsible for making as many copies as they need.

Figure 23 shows a sample worksheet. It is called the *common data definition worksheet*. It lets the user specify initial values for data items (such as limit check variables), to assign symbolic names to the data items, and to control operator access to the data.

The worksheet in the figure has been completed to provide:

- Symbolic names for several data items
- The values to be used when the 3644 is first started
- The format to be used when data is displayed
- Operator access to be allowed for the two variables that are used as high and low limits

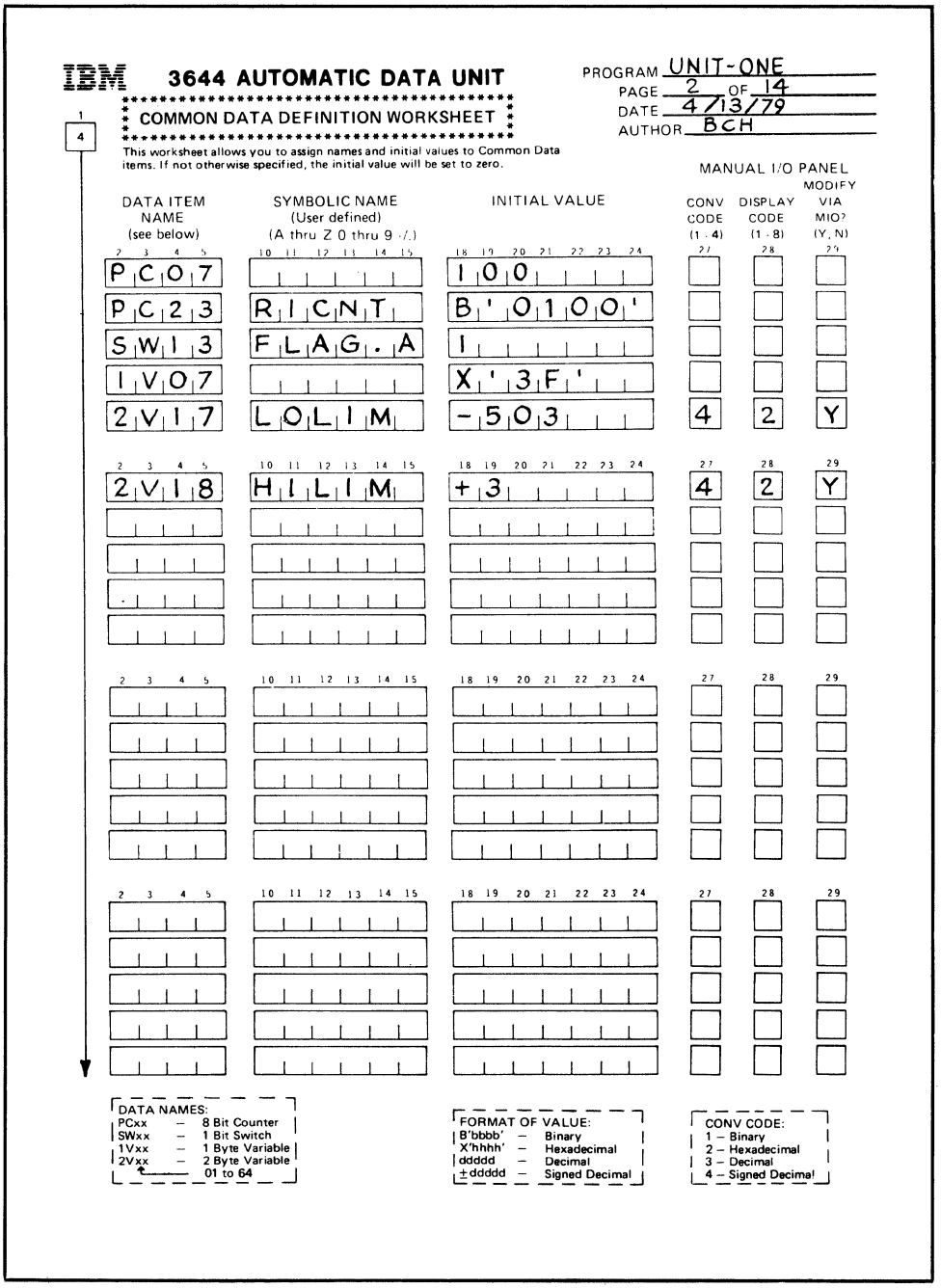


Figure 23. Sample DPPX GEN3644 Worksheet

Chapter 24. DPPX Performance Tool

DPPX Performance Tool is a program product (5760-XR5) that monitors and reports on activity of the 8100 system. It can be used to help tune the system both after initial installation and whenever the system changes.

DPPX Performance Tool has two components:

- The Monitor feature, which collects performance data
- The Reporter feature, which generates reports from the data collected by the Monitor feature

Types of Reports

DPPX Performance Tool can generate:

- Processor information:
  - Utilization by priority level
  - Distribution of dispatch queue by priority level
  - Utilization by environment
- Real storage information:
  - System buffer pool usage
  - Real storage allocation range
  - Transient module usage
  - Dynamic resident module fetches
  - Storage usage by environment
- Disk information:
  - Utilization by device
  - Distribution of request queue by device
  - Data set utilization
- DPPX/SP transaction information:
  - Number of transactions by type
  - Time under DPPX/SP control by type—time waiting for dispatch and time dispatched

Using DPPX Performance Tool

A user invokes both the Monitor and Reporter by command.

The user specifies which events to monitor. (The fewer the events, the better the Monitor's performance.) The user also specifies at what intervals the Monitor should collect data.

The Monitor writes output records to a data set at these intervals. This data set is input to the Reporter (when the Monitor has completed executing). The Reporter's output is also to a data set, which can be printed.

Data Collection Techniques

Data on DPPX/SP transient module usage, dynamic resident module fetches, and disk data set utilization are obtained by tracing. All other DPPX/SP measurements are obtained by sampling.

Statistics on the DPPX/SP data base management and transaction processing services are obtained by capturing transaction entry, dispatch, and exit events at appropriate points in DPPX/SP modules.

Chapter 25. Host Command Facility (HCF)

Host Command Facility (HCF) lets a user at a host-connected terminal access one or more 8100s. The user can issue any DPPX/SP command that does not require operator presence (such as inserting a diskette or mounting a tape would).

HCF helps manage a network. It lets a central-site user operate, control, and debug distributed 8100s. The major advantage is that it lets you form a central support group staffed by skilled data processing personnel at your central host location. This eliminates the need for skilled data processing personnel at each 8100 location.

HCF has two versions:

- HCF Version 1 (5735-XR1), which allows monitoring of one 8100 from a host-connected terminal
- HCF Version 2 (5668-985), which allows monitoring of more than one 8100 from a host-connected terminal when used with the Network Communications Control Facility (NCCF) program product's Terminal Access Facility feature

Unless otherwise specified, all features described in this chapter apply to both versions.

HCF runs in the host as a TCAM, VTAM, ACF/VTAM, ACF/TCAM, or ACF/VTAME application program under DOS/VS, VSE, OS/VS1, or OS/VS2 (MVS).

HCF makes it appear to users that they are connected directly to an 8100, when they are actually connected through the host.

Several HCF users can access the same DPPX/SP 8100 system at the same time.

HCF allows central:

- Administration
- Operation
- Problem determination
- Application development

HCF works with 8100s that have either the DPPX/SP or the Distributed Processing Control Executive (DPCX) operating system. This discussion is limited to 8100s that have DPPX/SP.

For a more complete description of HCF, see *Host Command Facility General Information*, GC27-0453.

Central Administration

An HCF user can define new devices to the system, tune the system after new devices have been added, test the new system, add or delete user IDs and passwords, and apply fixes to remote problems from the central site.

Central Operation

An HCF user can initiate work, terminate work, and perform all system control functions available to the DPPX/SP operator (except, as previously mentioned, functions that require physical presence, such as inserting a diskette or mounting a tape). All DPPX/SP operator messages can be routed to the HCF operator.

Central Problem Determination

An HCF user can access user and system data sets that might be required for troubleshooting. These include dump data sets, the error log, the system operator session log, and the trace data set.

HCF provides access to on-line diagnostic and problem-determination facilities, including logical connection verification, which verifies whether data can be transferred between the host and the 8100.

Central Application Program Development

The HCF user can develop, test, and debug application programs on any 8100 in the network. All DPPX/SP tools for interactive program writing, compiler invocation, and debugging are available.

However, you must add certain program products to your system to develop applications. Which program products you add depends on what languages your installation uses.

A Single Point of Network Control

With HCF Version 2, a single terminal can be used as the network operator's terminal.

From this terminal, the network operator can access all the facilities of the NCCF Release 2 program product in the host, the Network Problem Determination Application (NPDA) program product in the host, and all DPPX/SP problem determination functions in the 8100.

HCF Version 2 lets the network operator establish multiple concurrent sessions with 8100s from a single NCCF Release 2 terminal at the host. These sessions can be full-screen sessions or line-by-line sessions.

Full-screen sessions require that the control terminal be dedicated to one session at a time. Line-by-line sessions allow an interleaved display of messages from multiple 8100s. This allows operational monitoring of multiple 8100-DPPX/SP systems from a single terminal.

Some Considerations

For communication from a terminal to the host, HCF supports Binary Synchronous communication (BSC), Synchronous Data Link Control (SDLC), and local attachment.

For communication from the host to an 8100, HCF supports only SDLC links.

The HCF user must have the appropriate user profile and authorization for the selected DPPX/SP application. All 8100 authority checking is performed by DPPX/SP, not by HCF.

Differences between Version 1 and Version 2

HCF Version 2 has these features in addition to those of HCF Version 1:

- Provides concurrent control of multiple 8100s from a single terminal when used with the Terminal Access Facility feature of the NCCF Release 2 program product (5735-XX6).
- Supports the 3767 Communication Terminal at the host. The 3767 is a keyboard-printer terminal.

HCF Version 2 is compatible with HCF Version 1.

HCF Version 2 is designed to work closely with the Terminal Access Facility feature of NCCF Release 2. Without that feature, HCF Version 2 is functionally equivalent to HCF Version 1.

Chapter 26. Distributed Systems Executive (DSX)

Distributed Systems Executive (DSX) Version 2 is a program product (5668-986) that allows a user at a host-connected terminal to retrieve data sets from 8100s in a network and store, manage, and distribute them wherever they are needed in the network.

DSX runs in the host under VSE, OS/VS1, or OS/VS2 MVS. The host data transfer component of DPPX/SP in the 8100 supports DSX.

DSX provides library and transfer functions to help manage a network. With DSX at the host, it is possible to:

- Retrieve DPPX/SP programs, maps, command lists, and other data sets from an 8100 and put them into central libraries in the host.
- Send these same items from the host to each 8100 in the distributed system, thus adding, deleting, or replacing data sets in these systems.
- Retrieve program fixes from an 8100 and put them into the host central libraries.
- Send a command list to DPPX/SP for immediate execution or for batch execution. For example, you can send the command to install fixes after they are transmitted.
- Send a message to an 8100 user. For example, the message could tell the operator that fixes have been installed.
- Send print data to an 8100 and retrieve dumps produced at an 8100.
- Initiate a DPPX/SP session by releasing it from the DSX hold status. This is done with a DPPX/SP command issued by the operator at the 8100 site (Version 2 only).
- Display sessions by status, destination, or identification.
- Display and respond to certain messages issued by DSX.
- Enter a command to hold or delete data before it is sent (Version 2 only).

Using DSX, you can assign application program development to one or more DPPX/SP locations and control updates or new releases of the programs.

DSX works with 8100s that have either the DPPX/SP or the Distributed Processing Control Executive (DPCX) operating system, and the 3790 Communication System controller. This discussion is limited to 8100s that have DPPX/SP.

For a more complete description of DSX, see *Distributed Systems Executive General Information*, GH19-6229.

Using DSX

The key elements of DSX are central libraries and control files, the programs to build and use them, and a transmission control program that handles program and data flow between the host and clusters in the network.

All of the DSX programs are controlled by statements that allow the user to specify the functions to be performed, the data to be used, and other parameters qualifying the request.

When a DSX user establishes a communication session with a remote 8100, the content and order of the transmission are controlled by a transmission control file previously prepared by the user. During the session, programs and data are sent to and from remote clusters and stored in or retrieved from DSX libraries and files.

A file called the *cluster master file* keeps accurate records of the status and contents of each cluster and logical groups of clusters in the network.

DSX produces reports that allow the user to inspect the contents of libraries and files and to determine the results of DSX program execution.

Central Library Support

The central library support functions include:

- Library maintenance—The ability to create and update a library of programs, maps, and command lists.
- Cluster master file maintenance—The ability to create and update an inventory of programs, maps, command lists, data set IDs, and configuration data that belong to each 8100 in the network.
- Group table maintenance—Group tables exist so that the user can define logical groupings (for example, by application type or geographic location) of 8100s and resources. Functions may then be requested for one or more groups rather than for each individual cluster or resource.
- Reports—Facilities are available for displaying the status of the clusters in the network, the contents of the DSX libraries, and the status of scheduled sessions.
- Data formatting—DSX prepares print data and other user data for transmission from the host to an 8100. Data sets received from 8100s are stored on virtual storage access method (VSAM) files, but these may be converted into sequential data sets for processing at the host.
- Printing formatted abend dumps—Dumps retrieved from DPPX/SP clusters may be printed at the host by DSX if they are formatted when retrieved.
- Incident reporting and tracking—The ability to record and report incidents or errors by clusters. The incidents or errors may be opened, closed, or reopened. User-noted incidents are entered by the user at the host with control statements. In addition, error logs can be retrieved from 8100s.
- Security—The maintenance of passwords and logon IDs for sessions between DSX and DPPX/SP. Each central library support function may also be individually protected by password verification.

DSX Files

There are five types of DSX files. All but one are VSAM files and stay at the host location. The files are:

- Libraries—Contain a complete, current set of the network's resources: programs, maps, and command lists. The libraries contain tables of groups of these resources defined by the user, and also any security passwords defined by the user.

- Holding files—Provide temporary storage for data to be sent to the clusters (messages, print data, file updates, and data sets) and data received from the clusters (transactions, messages, data sets, and storage dumps). In addition, one file stores records of problem incidents entered by the user.

- Master file—Includes the cluster master file, which contains central information about the clusters in the network, and a group table, which lists clusters assigned to geographic or functional groups by the user.

- Transmission control file—Contains current schedules for transmission sessions, as defined by the user.

- Statistics file—Contains a history of sessions scheduled and what happened when the sessions were run. It is updated automatically after every transmission run.

DSX Reports

DSX offers a variety of reports to keep users informed of network status and the progress of transmission sessions.

Some reports are automatic:

- There is an automatic statistics report at the end of every transmission run.

- Every DSX function provides a printed log of what occurred during execution. This log lists all control statement input, with comments if any, and system messages and return codes.

Other reports must be requested:

- Cluster master report—for network administrators, system programmers, application programmers and designers, operators, and programming service representatives at the host. The information in this report could serve as background for application design and as input for host access method and network control program system generation. Network administrators will use this report for network planning, scheduling the installation of new applications, and for scheduling engineering changes with programming service representatives.

  This report gives the cluster's current status, information about the configuration (such as which controllers, stations, and features it has), what software is assigned to the cluster, and the date and time it was assigned, installed, or deleted.

- Cluster incident report—presents all the information entered about user-noted problems, such as problem description, status, and person or organization responsible.

- Library directory report—a complete listing of the directory of the DSX libraries, by resource name, in alphameric order. The report lists resources by type: programs, maps, and command lists.

- Group table report—lists the current assignments of clusters, programs, command lists, and so on, to groups as made through previous user control statements.

- Session schedule report—prints out the current contents of the transmission control file, that is, all sessions scheduled and all sessions completed and not deleted, with full details for each session.

- Session statistics report—available in two forms. One gives statistics for each session that has been completed and lists the functions performed during that session. The second form of the report gives cumulative statistics from the statistics file cluster by cluster.

- Error log report—summarizes the error records retrieved from DPPX/SP and stored in a DSX holding file.

- Command list report—prints out the full text of DPPX/SP command lists stored in the DSX resource library.

- Security report—available only to authorized personnel, this report lists the current password for each DSX central library support function, and current logon ID and password for each 8100-DPPX/SP cluster.

Advantages of DSX

The features of DSX making it a valuable tool for controlling a distributed system network are:

- Central library support—Since modules and functions can be stored by version and modification level, the network administrator can recover earlier copies for backup and store future versions before they are needed.

- Network inventory—DSX maintains complete, orderly records of configurations and resource assignments.

- Problem analysis—Host access to storage dumps, as well as to information on other user-noted problems, helps in problem analysis.

- Flexibility—Users are given many options in report writing; definition of user-noted problems; initiation of transmission by time, date, cluster, groups, or some combination.

- Means of controlling change—With the network equipment and data inventory maintained in files, DSX becomes a useful tool for keeping track of 8100 configurations and modifications.

  Since DSX checks the cluster master file before scheduling transmission (to see if a program is assigned to a particular cluster, for example), users are

prevented from making some time-consuming mistakes. Reports of cluster configurations, assigned programs, data sets, and so forth, and incidents occurring in the clusters may help to identify and correct problems resulting from incorrectly planned or implemented changes.

- Reporting—Users can obtain complete, current information on a specific cluster, data library contents, scheduled transmission sessions, abend dumps, or recorded problems. Reports can be current summaries or can show historical details and averages.

- Productivity—DSX contributes to the productivity of everyone concerned with maintaining a distributed network by giving them: a common control statement language for all maintenance functions; syntax checking to prevent erroneous file updates; meaningful reports showing the status of files and DSX execution results; reusable session schedules; and operator commands to allow retry, restart, and skipping of problem transmissions without terminating the session.

- Auditability—DSX provides four features that allow for audit trails:

  - Dated status entries in the cluster master file show information such as when a program was assigned to a cluster, when installed, and when deleted. Status changes are automatically recorded after a DSX transmission.

  - When a DSX record is created or updated, the time date, and program name are stamped on the record.

  - A logging exit is provided for a user-written logging program or recovery system.

  - DSX archives the results of each session that is started.

- Security—Session logons and passwords for DPPX/SP users can be password protected. DSX batch functions may also be password protected.

- Recovery—DSX can restart transmission of a partially transmitted DPPX/SP data set without retransmitting all of the data.

Bibliography

These manuals contain additional general information about programs and hardware that are mentioned in this manual.

8100 Hardware

An Introduction to the IBM 8100 Information System, GA27-2875

Host Programs for Management of Networks That Have 8100s

Host Command Facility General Information, GC27-0453

Distributed Systems Executive General Information, GH19-6229

Host Programs for General Network Management

Network Communications Control Facility General Information, GC27-0429

Network Problem Determination Application General Information, GC34-2010.

Information/System General and Pre-Installation Information, GC34-2027

Host Access Methods

Introduction to VTAM, GC27-6987

Introduction to Advanced Communications Function, GC30-3033

ACF/VTAM General Information:Introduction, GC27-0462

ACF/TCAM General Information:Introduction, GC30-3057

ACF/VTAME General Information:Introduction, GC27-0438

Host Transaction-Processing Systems

CICS/VS General Information, GC33-0066

IMS/VS General Information, GH20-1260

Host Remote Job Entry Programs

JES2/NJE General Information, GC23-0010

JES3 Introduction, GC28-0607

VSE/POWER General Information, GH12-5128

3705 Communications-Controller Program

ACF/NCP/VS General Information, GC30-3058

Telecommunication Protocols

Binary Synchronous Communication General Information, GA27-3004

IBM Synchronous Data Link Control General Information, GA27-3093

IBM Implementation of X.21 Interface General Information, GA27-3287

Cross-Industry Information-Handling Systems

IBM 3270 Introduction, GA27-2739

IBM 3767 Component Description, GA27-3096

IBM 3600 System Summary, GC27-0001

IBM 3650 Retail Store System Introduction, GA27-3075

IBM 3680 Introduction, GA27-3199

Other Reading

Systems Network Architecture Concepts and Products, GC30-3072

The IBM Diskette General Information Manual, GA21-9182

Distributed Processing Control Executive General Information, GC22-9075

Distributed Office Support Facility General Information, GC27-0546

Development Management System/DPPX/SP General Information, GH20-5555

Glossary

This glossary defines terms and abbreviations used in this publication. If you do not find the term you are looking for, refer to the index or to the *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems*, GC20-1699.

This glossary includes definitions from:

- The *American National Dictionary for Information Processing*, copyright 1977 by the Computer and Business Equipment Manufacturers Association, copies of which may be purchased from the American National Standards Institute at 1430 Broadway, New York, NY 10018.
- The *ISO Vocabulary of Data Processing*, developed by the International Standards Organization, Technical Committee 97, Subcommittee 1. Definitions from published sections of this vocabulary are identified by the symbol "(ISO)" preceding the definition. Definitions from draft proposals and working papers under development by the ISO/TC97 vocabulary subcommittee are identified by the symbol "(TC97)," indicating that final agreement has not yet been reached among its participating members.
- The *ISO Vocabulary of Office Machines*, developed by subcommittees of ISO Technical Committee 95. Definitions from published sections of this vocabulary are identified by the symbol "(ISO)" preceding the definition. Definitions from draft proposals and working papers under development by ISO/TC95 subcommittees are identified by the symbol "(TC95)," indicating that final agreement has not yet been reached among its participating members.
- The *CCITT Sixth Plenary Assembly Orange Book, Terms and Definitions*, and working documents published by the International Telecommunication Union, Geneva, 1978. These are identified by the symbol "(CCITT/ITU)" preceding the definition.

A

access method. A technique for moving data between processor storage and input/output devices. TCAM and VTAM are examples of access methods.

ACF. Advanced Communication Function.

ACF/NCP. Advanced Communication Function for the Network Control Program.

ACF/TCAM. Advanced Communication Function for the Telecommunications Access Method.

ACF/VTAM. Advanced Communication Function for the Virtual Telecommunications Access Method.

ACF/VTAME. Advanced Communication Function for the Virtual Telecommunications Access Method Entry.

adapter. In 8100, a connection provided on the 8130, 8140, 8150, and 8101 for attaching displays, printers, magnetic tape units, or another 8101 unit.

address space. The complete range of addresses that are available to a programmer.

administrator. See *system administrator*.

Advanced Communication Function (ACF). A group of IBM program products (principally ACF/TCAM, ACF/VTAM, ACF/VTAME, and ACF/NCP) that uses the concepts of systems network architecture (SNA), including distribution of function and resource sharing.

Advanced Communication Function for the Network Control Program (ACF/NCP). A program product that provides communication controller support for single-domain and multiple-domain networks.

Advanced Communication Function for the Telecommunications Access Method (ACF/TCAM). A program product that provides single-domain network capability and, optionally, multiple-domain capability.

Advanced Communication Function for the Virtual Telecommunications Access Method (ACF/VTAM). A program product that provides single-domain network capability and, optionally, multiple-domain capability.

Advanced Communication Function for the Virtual Telecommunications Access Method Entry (ACF/VTAME). A program product that provides single-domain and multiple-domain network capability for 4300 systems that may include communication adapters.

alert. A unit of information, usually indicating the loss of a system resource (RECFMS RU type 00), passed from a machine or program to a host to signal an error.

American National Standards Institute (ANSI). An organization for the purpose of establishing voluntary industry standards.

ANSI. American National Standards Institute.

APAR. Authorized program analysis report.

APL. A Programming Language. DPPX APL is a general-purpose language. It can be used in data processing, system design, and mathematical and scientific computation. APL has a compact way of expressing problems and is suited to interactive problem solving.

application data structure. The names and other data specified for variable fields in a map developed with the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product. They are structured into a form that is acceptable in a programming language and can be copied into a source program to define an input/output area. The application data structure is produced by the mapgroup generator step of DPPX/SP IMD.

application development system. A system used to develop user-written application programs that are used later in a production system.

application program. (1) A program written for or by a user that applies to a particular application. (2) In data

communication, a program used to connect and communicate with terminals in a network, enabling users to perform application-oriented activities.

ASP. Asymmetric multiprocessing system.

Asymmetric multiprocessing system (ASP). An extension to the IBM System/360 operating system that provides increased automation of computer operations for large-scale data processing installations.

Assembler. A computer program for (1) translating programs expressed in an assembly language into a machine language and (2) linking subroutines.

Authorized Program Analysis Report (APAR). A request for correction of a problem caused by a defect in a current unaltered release of a program. A program fix or corrected code is issued to the customer and a corresponding correction is incorporated into subsequent releases of the program.

automatic logon. The means of remotely initiating and terminating a session between the system and a terminal operator.

auxiliary storage. Data storage other than main storage; for example, storage on magnetic tape or direct access volumes.

B

basic data exchange format. A standard data format for the IBM Diskette 1 and the IBM Diskette 2, used to exchange data between different systems.

basic license. A type of license for using an IBM licensed program. It includes these things for a designated 8100 processor: distribution of licensed program material and service updates, program service, access to a central service facility through an IBM support center, and APAR processing support via the IBM support center.

basic telecommunications access method (BTAM). An access method that permits read/write communication with remote devices.

batch mode. The condition established so that batch processing can be accomplished.

batch processing. (ISO) The processing of data or the accomplishment of jobs accumulated in advance in such a manner that each accumulation thus formed is processed or accomplished in the same run.

Binary Synchronous Communication (BSC). Communication using data transmission in which synchronization of characters is controlled by timing signals generated at the sending and receiving stations.

break-point. In DPPX/SP, an instruction address stop that can be established by command. See *instruction address stop*.

BSC. Binary synchronous communication.

BTAM. Basic telecommunications access method.

C

catalog. In DPPX/SP, a data set containing information describing data sets, other catalogs, and the direct-access storage assigned to data sets and catalogs.

CCITT. International Telegraph and Telephone Consultative Committee.

CICS/VS. Customer Information Control System/Virtual Storage.

CLIST. Command list.

cluster. A station that consists of a control unit (cluster controller) and the terminals attached to it.

COBOL (COmmon Business-Oriented Language).. An English-like programming language designed for business data processing applications.

collating sequence. (ISO) A specified arrangement used in sequencing.

Command Facility. The part of DPPX/SP that establishes interactive terminal sessions and processes all commands.

command list (CLIST). In DPPX/SP, a data set in which commands and possibly subcommands and data are stored for subsequent execution.

communication controller. A type of communication control unit whose operations are controlled by one or more programs stored and executed in the unit. It manages the details of line control and the routing of data through a network.

compiler. A program that translates a computer program expressed in a problem-oriented language into a machine language program to be executed at a later time.

configuration. The collection of programs and devices that make up a particular data processing system.

controller. (TC97) A device that directs the transmission of data over the data links of a network; its operation may be controlled by a program executed in a processor to which the controller is connected or by a program executed within the device.

control unit. A device that controls input/output operations at one or more devices.

conversational. Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain his train of thought.

Cross System Product/Application Development for DPPX/SP. A program product that guides users online in writing application programs by requesting responses to prompts at display terminals. It can be used instead of a programming language.

Cross System Product/Application Execution for DPPX/SP. A program product that executes application programs developed using Cross System Product/Application Development for DPPX/SP.

cursor. A movable spot of light on the screen of a display device, usually indicating where the next character is to be entered, replaced, or deleted.

Customer Information Control System/Virtual Storage (CICS/VS). A System/370 or 4300 program product that enables transactions entered at remote terminals to be processed concurrently by user-written application programs. It also includes facilities for building, using, and maintaining data bases.

D

data area. A storage area used by a program to hold information.

data base. A collection of data fundamental to a system or enterprise.

Data Capture and Management System (DCMS) for DPPX. A program product that allows bulk, full-screen data entry.

data communication. The transmission and reception of data.

data definition (DD) statement. A job control statement that describes a data set associated with a particular job step.

data dictionary. A centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format. It assists company management, data base administrators, system analysts, and application programmers in effectively planning, controlling, and evaluating the collection, storage, and use of data.

data exchange. A process whereby information is written on a diskette at one system and used in another system.

data integrity. (1) (TC97) The quality of data that exists as long as accidental or malicious destruction, alteration, or loss of data are prevented. (2) Preservation of data for its intended purpose.

data link. The physical connection and the connection protocols between the host and communication controller nodes via the host data channel.

data-link-attached loop. In 8100, a transmission loop used to attach I/O devices to the system through a data-link facility rather than directly by cables. Contrast with *directly attached loop*.

data security. (CCITT/ITU) The protection of data against unauthorized disclosure, transfer, modification, or destruction, whether accidental or intentional.

DCMS. Data Capture and Management System.

DD statement. Data definition statement.

deadlock. Unresolved contention for the use of a resource.

debug. (ISO) To detect, to trace, and to eliminate mistakes in computer programs or in other software.

dialog. In an interactive system, a series of interrelated inquiries and responses analogous to a conversation between two people.

direct access storage. (ISO) A storage device that provides direct access to data.

direct connection. In 8100, the attachment of another 8100 system, a terminal, or other I/O device through a selected communication interface and a limited-length cable. No modem is required.

directly attached loop. In 8100, a loop that connects to the loop adapter by cables, rather than through a data link, and allows attachment of a variety of I/O devices. Contrast with *data-link-attached loop*.

disk. See *magnetic disk*.

diskette. A thin, flexible magnetic disk and a semirigid protective jacket, in which the disk is permanently enclosed.

display and printer attachment feature. In 8100, control logic that allows the connection of various I/O devices to the 8101 Storage and Input/Output Unit.

display image. (TC97) In computer graphics, a collection of display elements or display groups that are represented together at any one time in a display space.

distributed data processing. Data processing in which some or all of the processing, storage, control, input, and output functions are situated in different places and connected by transmission facilities.

Distributed Office Support Facility (DOSF). An IBM program product for text processing and paperwork management at an IBM 8100 Information System. It runs under the Distributed Processing Control Executive (DPCX) operating system. DPPX/SP supports attachment of DPCX and DOSF.

Distributed Presentation Services (DPPX/DPS). A program product that helps programmers specify how an application program should format data on display terminals and printers.

Distributed Processing Control Executive (DPCX). An operating system for the IBM 8100 Information System. DPPX/SP supports attachment of DPCX.

Distributed Processing Programming Executive (DPPX). A comprehensive collection of program products that make up an operating system for 8100 Information System hardware. DPPX includes the DPPX Base and other licensed programs that provide programming languages, application support, and host network access.

Distributed Processing Programming Executive System Product (DPPX/SP). A program product that schedules and supervises the execution of programs written for the IBM 8100 Information System computers. DPPX/SP and the 8100 are designed to distribute processing among many computers in a cooperative network. Along with associated program products, DPPX/SP forms an interactive operating system for running application programs on 8100 computers.

Distributed Processing Programming Executive System Product Interactive Map Definition (DPPX/SP IMD). An optional program product supported on DPPX/SP. It is used during application program development. It issues prompts to guide the programmer in designing run-time screen and printer layouts.

distributed system. A data processing system in which processing, storage, and control functions, in addition to input and output operations, are distributed among remote locations.

Distributed Systems Executive (DSX). A program product that runs in a 308x, 303x, 4300, or System/370 host and provides central library support and batch transmission control for 8100 Information System processors (with DPPX/SP, DPPX, or DPCX) and 3790 Communication System controllers. It lets a user store, manage, and distribute programs and data throughout a distributed data processing network.

distributed systems license option (DSLO). An option that permits a customer with a basic license to copy certain IBM licensed materials to a designated 8100 for the purpose of installing multiple systems. It provides a charge lower than the basic license charge and program service through the customer location designated for the basic license.

DMS/DPPX/SP Definition. Development Management System/DPPX/SP Definition.

DMS/DPPX/SP Execution. Development Management System/DPPX/SP Execution

domain. All the resources that are controlled by a given system services control point (SSCP).

DOS/VS. Disk Operating System/Virtual Storage.

DOSF. Distributed Office Support Facility.

down time. The time interval during which a machine cannot operate because of a fault.

downstream. Referring to a machine that is not directly attached to a host processor.

DPCX. Distributed Processing Control Executive.

DPPX. Distributed Processing Programming Executive.

DPPX/SP. Distributed Processing Programming Executive System Product.

DPPX/SP IMD. Distributed Processing Programming Executive System Product Interactive Map Definition.

DPS. Distributed Presentation Services.

DSLO. Distributed systems license option.

DSX. Distributed Systems Executive.

dump. The contents of storage, or a part of storage, usually written from main storage to an external medium. A dump is requested for a specific purpose, such as to allow other use of the storage, as a safeguard against faults or errors, or in connection with debugging.

E

environment. A named collection of logical and physical resources used to support the operation of a function.

error log. In DPPX/SP, a data set used to record information about certain hardware and programming events.

execute. (ISO) To perform the execution of an instruction or of a computer program.

execution debug monitor. A debugging aid that intercepts DPPX/SP requests to a display or printer device and displays information about them. The monitor can be used on field-formatted and logical-record connections.

Execution Manager (EM). A component of the DPPX Presentation Services for 3640 Terminals program product. It is used during application program execution. It controls the transfer of data between an application program and a terminal, based on the maps defined using the Interactive Transaction Generator.

exit routine. A routine that receives control when a specified event occurs, such as an error.

F

fix. A correction of an error in a program. It is usually a temporary correction or bypass of defective code.

fix package. In DPPX/SP, program temporary fixes resident on one or more diskettes that, along with the appropriate documentation, provide DPPX/SP program product corrections.

FM control structure. A set of names supplied as part of the DPPX/SP Interactive Map Definition (DPPX/SP IMD) program product in a form that can be copied into a source program and used to define a control area. The structure is used to control the interaction and other aspects of presentation from the application program.

FORTRAN (FORmula TRANslation). A programming language primarily used to express computer programs by arithmetic formulas.

full-screen editing. An editing mode that allows changes to be made in any area of a keyboard-display terminal's screen. (Contrast with a line editor, where changes can only be made on one line.)

functional enhancement package (FEP). Program code and related material that improve the function of a DPPX/SP program product.

G

general-purpose register. (ISO) A register, usually explicitly addressable within a set of registers, that can be used for different purposes, for example as an accumulator, as an index register, or as a special handler of data.

generated mapgroup. The ouput of a logically related set of maps produced by the mapgroup generator in IMD.

GEN3644. Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit.

H

HCF. Host Command Facility.

hardware. (ISO) Physical equipment used in data processing, as opposed to computer programs, procedures, rules, and associated documentation.

HASP. Houston Automatic Spooling Program.

high-level language. (1) (TC97) A problem-oriented language that requires little knowledge of the computer on which a computer program written in the language is to be run; that facilitates translation of computer programs in this language into several different machine codes; and that usually results in many machine instructions for each statement in the source program; for example, ALGOL, COBOL, COGO, FORTRAN, PL/I, SIMSCRIPT. (2) In DPPX/SP, the high-level languages are COBOL, FORTRAN, APL, and PL/I.

host (or host computer). The central or controlling processing unit in a configuration with more than one processing unit. For the 8100 system, a host is either a 308x, 303x, 4300, or System/370.

Host Command Facility (HCF). A program product that allows a user at a System/370- or 4300-connected terminal to access one or more 8100s. When used with DPPX/SP, the Host Command Facility user can issue any DPPX/SP command that does not require operator presence (such as inserting a diskette or mounting a tape would). Host Command Facility can also be used with Distributed Processing Control Executive (DPCX) operating system.

Houston Automatic Spooling Program (HASP). A computer program that provides supplementary job management, data management, and task management functions such as control of job flow, ordering of tasks, and spooling.

I

IMD. DPPX/SP Interactive Map Definition.

IMS/VS. Information Management System/Virtual Storage.

index. A table used to locate the records of an indexed data set.

indexed data set. A type of data set in which records are stored and retrieved on the basis of keys that are within each record and are part of the data record itself.

Information Management System/Virtual Storage (IMS/VS). A System/370 or 4300 program product that assists computer users in implementing telecommunication and batch-type data processing operations.

initial program load (IPL). The initialization procedure that causes an operating system to start operation.

instruction address stop. An instruction address which, when fetched, causes execution to stop.

integrity. See *data integrity*.

interactive. Pertaining to an application in which each entry calls forth a response from a system or program. The DPPX/SP commands can be used interactively.

interactive debug. In DPPX/SP, a component that supports the monitoring of program execution through the use of the DEBUG.PGM command and its subcommands.

interactive editor. In DPPX/SP, a component that permits the rearrangement, modification, and deletion of data through the use of an EDIT command and its subcommands at a keyboard display or keyboard printer.

Interactive Transaction Generator (ITG). A feature of the DPPX Presentation Services for 3640 Terminals program product. It is used during application program development. It guides a programmer at a display terminal in the creation of run-time formats (maps), which include prompts, edits, light control, and I/O control.

International Telegraph and Telephone Consultative Committee (CCITT). A committee responsible for worldwide standardization of international telecommunication facilities.

IPL. Initial program load.

IPO. See *System Installation Productivity Option*.

J

JES. Job entry subsystem.

JES2. A functional extension of the HASP II program that receives jobs into a host system and processes all output data produced by the job.

JES3. A functional extension of the ASP program that receives jobs into a host system a processes all output data produced by the job.

K

Kanji. The Japanese pictographic script.

L

language translator. A general term for any assembler, compiler, or other routine that accepts statements in one language and produces equivalent statements in another language.

licensed program. Any separately priced program that bears an IBM copyright and is offered to customers under the terms and conditions of the Agreement for IBM Licensed Programs. Types of licensed programs are: program product (PP), industry application program (IAP), field-developed program (FDP), installed user program (IUP), and programming request for price quotation (PRPQ).

line-control discipline. See *link protocol*.

line editing. In DPPX/SP, an editing mode that supports the rearrangement, modification, and deletion of data from a keyboard printer or keyboard display.

link protocol. (CCITT/ITU) The set of rules by which a logical data link is established, maintained, and terminated, and by which data is transferred across the link. It includes the format by which control information is passed, and the rules by which it is interpreted, in order to transmit data across the link.

load module. (ISO) A program unit that is suitable for loading into main storage for execution. In DPPX/SP, it is the output of the linkage editor.

locking. In DPPX/SP, a method of ensuring the uninterrupted use of a data area or code by one thread.

log. A chronological record of the changes made to a set of data; the record may be used to reconstruct a previous version of the set. Synonymous with journal.

logical record. (ISO) A record independent of its physical environment. Portions of the same logical record may be located in different physical records, or several logical records or parts of logical records may be located in one physical record.

logical storage. The concept of storage space that may be regarded as addressable main storage by an application program in which logical addresses are mapped into real addresses through control blocks.

log off. To terminate a session.

log on. To initiate a session.

loop. In 8100, an arrangement that consists of 8100 Information System control units and displays connected directly or by data links to the processor.

loop attachment. In 8100, circuitry that allows devices using a directly attached loop to communicate with the system.

M

machine language. A language that is used directly by a machine.

macro instruction. An Assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macro instruction in the program.

magnetic disk. (ISO) A flat circular plate with a magnetizable surface layer on which data can be stored by magnetic recording.

magnetic tape. (ISO) A tape with a magnetizable surface layer on which data can be stored by magnetic recording.

main storage. (ISO) Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

map. In DPPX/SP IMD and DPPX DPS, data that controls the relationship between names of program variables and the position of the fields in which their values will appear on a display device. The map also contains constant data and other formatting information.

mapgroup. A named collection of maps. All maps used in a single logical message (and hence all maps that are used on a single screen or partition) must come from the same mapgroup.

map specification library. The data set in which IMD object maps are held in their source form.

menu. A display on a display screen of a list of available program functions for selection by a terminal user.

merge. (ISO) To combine the items of two or more sets that are each in the same given order into one set in that order.

microcode. The instructions of a microprogram.

microprogram. (ISO) A sequence of elementary instructions that corresponds to a specific computer operation, that is maintained in special storage, and whose execution is initiated by the introduction of a computer instruction into an instruction register of a computer.

MODEM (MOdulator-DEModulator). A device that modulates and demodulates signals transmitted over data communication facilities.

Monitor. A feature of the DPPX Performance Tool program product. It collects performance data to be printed by the Reporter feature.

multileaving support. Fully synchronized two-directional transmission of a variable number of data streams between terminals and a computer, using BSC facilities.

multipoint connection. (TC97) A connection established among more than two data stations for data transmission. The connection may include switching facilities.

N

NCCF. Network Communications Control Facility.

network. In data communication, a system consisting of two or more interconnecting computing units.

Network Communications Control Facility (NCCF). A System/370 or 4300 program product consisting of a base for command processors that can monitor, control, and improve the operation of a data communication network.

Network Problem Determination Application (NPDA). A System/370 or 4300 program product that assists the user in identifying network problems from a central control point using interactive display techniques.

nonswitched connection. A connection that does not have to be established by dialing. Contrast with switched connection.

NPDA. Network Problem Determination Application.

O

object code. Output from a compiler or assembler that is itself executable machine code or is suitable for processing to produce executable machine code.

object map. A map that has been generated by the DPPX/SP Interactive Map Definition program product and will be used by DPPX/SP at program execution time. The object map has the format of a load module.

object program. A fully compiled or assembled program that is ready to be loaded into the computer.

online. (1) Pertaining to a user's ability to interact with a computer. (2) Pertaining to a user's access to a computer via a terminal.

operand. Information entered with a command name to define the data on which a command processor operates and to control the execution of the command processor.

operating system. (ISO) Software that controls the execution of computer programs and that may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and related services.

Operating System/Virtual Storage 1 (OS/VS1). A virtual storage operating system that is an extension of OS/MFT, the operating system that supports multiprogramming with a fixed number of tasks.

Operating System/Virtual Storage 2 (OS/VS2). A virtual storage operating system that is an extension of OS/MVT, the operating system that supports multiprogramming with a variable number of tasks.

operator panel. A display control panel that enables the user to input information, display system status, control powering and IPL from a host, control panel access, and override normal IPL parameters.

OS/VS1. Operating System/Virtual Storage 1.

OS/VS2. Operating System/Virtual Storage 2.

outboard formatting. Display formatting where constant data is held in an 8100 and variable data in a System/370 or 4300 host. The variable data mapped by Customer Information Control System/Virtual Storage (CICS/VS) in the host is merged with the constant data mapped by DPPX/SP in the 8100 and displayed as a unit.

P

panel. A predefined display image that defines the locations and characteristics of display fields on a display surface. With the DPPX/SP Interactive Map Definition program product, data values from one presentation data definition merged with formatting information from one map produce one panel.

Parameter Table Generation Facility for the IBM 3644 Automatic Data Unit (GEN3644). DPPX GEN3644—A program product that helps programmers efficiently customize the IBM 3644 Automatic Data Unit.

partition. A subdivision of a screen that can be used independently of other such subdivisions by a single application program.

password. In DPPX/SP, a one-to-six-character symbol that the user may be required to supply while logging on.

peer. (TC97) In network architecture, any functional unit that is in the same layer as another entity.

Performance Tool. DPPX Performance Tool—A program product that collects and reports information about DPPX/SP performance.

PL/DS. Programming Language for Distributed Systems.

PL/I. A programming language designed for use in a wide range of commercial and scientific computer applications.

point-to-point connection. A connection established between two data stations for data transmission. The connection may include switching facilities.

POWER. A program that improves throughput in a VSE system by separating unit-record input and output operations from internal computing operations. VSE/POWER allows VSE/ICCF users to submit jobs for execution in VSE batch partitions.

preconfigured definition. A set of system parameters, with corresponding command lists, that (1) define and activate a certain hardware arrangement for an 8100 Information System, including terminals and storage units and (2) determine certain programming characteristics.

Presentation Services for 3640 Terminals (PS3640). DPPX PS3640—A program product that helps programmers manage the layout of data for certain 3640 terminals.

Printer Sharing. A DPPX/SP function that allows interactive and batch users to share the same printers.

problem determination procedure. A prescribed sequence of steps taken to accomplish problem determination. Such procedures frequently include steps aimed at recovery from, or circumvention of, problem conditions.

procedure. (TC97) A sequenced set of statements that may be used at one or more points in one or more computer programs, and that usually has one or more input parameters and yields one or more output parameters.

processing and control element (PCE). The controlling center of an 8130, 8140, or 8150 Processor. It contains the sequencing and processing controls for instruction execution, interrupt control, dynamic address transformation, and other control and processing functions.

processor(ISO). In a computer, a functional unit that interprets and executes instructions.

production system. A system where application programs that are already developed and tested run on a regular basis.

profile. In DPPX/SP, a data area that describes the characteristics of certain DPPX/SP resources. Examples of these resources are: data sets, programs, catalogs, commands, and users.

program function key. A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

programmed symbols. Symbols that can be defined by an installation and used instead of the normal alphabetic and numerical characters.

Programming Language for Distributed Systems (PL/DS). An IBM-developed system programming language used to code most of the DPPX program products.

program product. A type of licensed program.

program temporary fix (PTF). A solution of a problem diagnosed by IBM field engineering as the result of a defect in a current unaltered release of the program.

prompt. A message issued to a terminal user requesting information necessary to continue processing.

protected data set. A data set that has the "protected" attribute in its data set profile.

PS3640. Presentation Services for 3640 terminals.

PTF. Program temporary fix.

Q

queue. A line or list formed by items in a system waiting for service; for example, batched jobs to be executed.

R

real storage. (ISO) The main storage in a virtual storage system. Physically, real storage and main storage are identical. Conceptually, however, real storage represents only part of the range of addresses available to the user of a virtual storage system. Traditionally, the total range of addresses available to the user was that provided by main storage.

reenterable. See *reentrant.*

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more threads.

register. See *general-purpose register.*

remote entry services (RES). In OS/VS1, the set of functions added to the job entry subsystem that allows jobs and their associated data to be entered from remote devices, processed at the central system, and then transmitted back to remote devices.

Remote Spooling Communications Subsystem (RSCS). The component of VM/370 that transfers spool files between VM/370 users, remote stations, and remote and local batch systems via HASP-compatible telecommunication facilities.

Reporter. A feature of the DPPX Performance Tool program product. It generates reports from data collected by the Monitor feature.

RES. Remote entry services.

resident. Pertaining to programming that is permanently located in main storage.

response time. (ISO) The elapsed time between the end of an inquiry or demand on a data processing system and the beginning of the response, for example, the length of time between an indication of the end of an inquiry and the display of the first character of the response at a user terminal.

RSCS. Remote Spooling Communications Subsystem.

run. (1) (ISO) A single performance of one or more jobs. (2) A single, continuous performance of a computer program or routine.

S

save/restore IPL. An initialization procedure in which data is restored. After a save initial program load (IPL) is done, if the system must be IPLed again, a restore IPL is done. Data is automatically restored to the state it was in when the save IPL was performed. The restore IPL is performed more quickly than a normal IPL.

screen image. See *display image.*

SDLC. Synchronous Data Link Control.

security. See *data security.*

service level update. A complete replacement of the basic machine readable material for a program product. It is the same as the previous service level of the program product with the latest fix package incorporated.

session. In SNA, a logical connection, established between two network addressable units (NAUs) to allow them to communicate. The session is uniquely identified by a pair of network addresses, identifying the origin and destination NAUs of any transmissions exchanged during the session.

session log. A data set, created by a set of commands, which contains a record of a user's terminal session.

Shadow file. A process by which each write to disk is copied in a parallel ("shadow") file, ensuring easily accessible duplicate copies of data sets.

SNA. Systems network architecture.

software (ISO). Computer programs, procedures, rules, and possibly associated documentation concerned with the operation of a data processing system.

sort. To arrange a set of items according to keys that are used as a basis for determining the sequence of the items; for example, to arrange the records of a personnel file into alphabetical sequence by using the employee names as sort keys.

source map. A map in the form in which it is held before it is generated by IMD. All editing is done on the source map.

source program. (1) (ISO) A computer program expressed in a source language. (2) Contrast with *object program.*

SPCE. Specified process check exit.

specified process check exit (SPCE) routine. In DPPX/SP, a program that gains control when certain events occur during the execution of a program. An SPCE routine could gain control when, for example, a system check or program exception occurs within a given program.

spooling. (ISO) The use of auxiliary storage as a buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer.

SSCP. System services control point.

stand-alone. Pertaining to operation that is independent of another device, program, or system.

stand-alone data processing system. A data processing system that is not served by communication facilities.

start-stop transmission. (TC97) Asynchronous transmission such that a group of signals representing a character is preceded by a start element and is followed by a stop element.

storage. A device, or part of a device, that can retain data.

store-and-forward transmission. A manner of transmitting messages through a computer network. Messages are stored before transmission toward the final destination.

subcommand. A request for an operation that is within the scope of work requested by a previously issued command.

subroutine. (ISO) A sequenced set of statements that may be used in one or more computer programs and at one or more points in a computer program.

switched connection. (1) (TC97) A mode of operating a data link in which a circuit or channel is established to switching facilities, as, for example, in a public switched network. (2) A connection that is established by dialing. (3) Contrast with nonswitched connection.

Synchronous Data Link Control (SDLC). A discipline for managing synchronous, transparent, serial-by-bit information transfer over a communication channel. Transmission exchanges may be duplex or half-duplex over switched or nonswitched data links. The communication channel configuration may be point-to-point, multipoint, or loop.

system administrator. The person at a computer installation who designs, controls, and manages the use of the computer system.

system-assisted linkage. In DPPX/SP, a method of passing control from one program to another in which the supervisor assists by saving program status and passing parameters to the called program.

system integrity. See *data integrity*.

system parameters. In DPPX/SP, variables that control the operational and execution characteristics of the system.

system residence volume. The volume on which the resident programming of the operating system and the highest-level index of the catalog are located.

system services control point (SSCP). In SNA, a focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Multiple SSCPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units and logical units within its domain.

systems network architecture (SNA). The total description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the communication system.

System/370. An upward-compatible extension of the IBM System/360, offering new function and improved price/performance.

T

TAF. Terminal Access Facility.

tape. See *magnetic tape*.

TCAM. Telecommunications access method.

telecommunications access method (TCAM). A method used to transfer data between main storage and remote or local stations. Application programs use either GET and PUT or READ and WRITE macro instructions to request the transfer of data, which is performed by message handlers.

Terminal Access Facility. A feature of the Network Communications Control Facility program product, which lets the NCCF operator control CICS/VS, IMS/VS, TSO, and HCF subsystems from one terminal. Through HCF the NCCF operator can control 8100 DPPX and DPCX operating systems.

thread. A collection of processes whose order determines the process eligible for execution. A thread is the element that is scheduled, and to which such resources as execution time, locks, and queues may be assigned.

throughput. (ISO) A measure of the amount of work performed by a computer system over a given period of time, for example, jobs per day.

transaction. An exchange between a terminal and another device that accomplishes a particular action or result, for example, recording sales items, processing refunds, recording coupons, handling voids, verifying checks before accepting as tender, and arriving at the amount to be paid by or to a customer.

transient area. A storage area used for temporary storage of programs or routines that do not reside in main storage.

tuning. The process of adjusting system control variables to make the system divide its resources most efficiently for the workload.

type H data exchange format. A standard data format for the IBM Diskette 2D, used to exchange data between different systems.

U

unit record. A card containing one complete record; a punched card.

user. A person requiring the services of a computing system.

user ID. A symbol identifying a system user.

V

viewport. That part of a screen on which data from a partition is displayed.

Virtual Storage Extended (VSE). An operating system that is an extension of DOS/VS, consisting of VSE/Advanced Functions (the minimum operating system support for a VSE-controlled installation), and other IBM program products.

virtual telecommunications access method (VTAM). A set of programs that control communication between terminals and application programs running under VSE, OS/VS1, and OS/VS2.

volume. That portion of a single unit of storage that is accessible to a single read/write mechanism, for example, a disk, a diskette, or a magnetic tape reel.

VSE. Virtual Storage Extended.

VTAM. Virtual telecommunications access method.

W

workstation. A configuration of input/output equipment at which a person works.

X

X.21. A recommendation of the International Telegraph and Telephone Consultative Committee (CCITT) for data transmission over public data networks. It defines a circuit switched protocol for synchronous operation between data terminal equipment and data communication equipment.

4

4300 processor. Small and moderately sized processors that have evolved from System/370.

8

8100 Information System. A collection of processors and devices that can be connected together to form a system for distributed processing or for general use. These systems can be used stand-alone, connected to other 8100 Information Systems, and connected to System/370 or 4300.

8100 processor. An 8130, 8140, or 8150 processor.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

3

5

8

DPPX/SP
General Information

Order No. GC23-0600-2

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

[ ] As an introduction
[ ] As a reference manual
[ ] For another purpose (explain) _____

[ ] As a text (student)
[ ] As a text (instructor)

_____

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? _____ _____

Newsletter number of latest Technical Newsletter (if any) concerning this publication: _____

If you wish a reply, give your name and address:

IBM branch office serving you _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

GC23-0600-2

Reader's Comment Form

Cut or Fold Along Line

Fold and tape | **Please Do Not Staple** | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department 52Q
Neighborhood Road
Kingston, New York 12401

Fold and tape | **Please Do Not Staple** | Fold and tape

IBM

IBM

DPPX/SP General Information

Printed in U.S.A.

GC23-0600-2