IBM System/3
Model 10 Disk System
Communications Control Program
Planning Manual

Features 6032–6033
Program Number 5702–SC1

GC21-7579-0

IBM System/3
Model 10 Disk System
Communications Control Program
Planning Manual

Features 6032–6033
Program Number 5702–SC1

GC21-7579-0

Preface

This publication describes the Communications Control Program feature (the CCP) of the IBM System/3 Model 10 Disk System and provides information to aid customers, IBM Systems Engineers, and IBM Marketing Representatives in planning for use of the CCP.

The CCP, using the facilities of Disk System Management, provides the control program services needed to operate a communications-based information processing system. The CCP incorporates the IOCS programs for the Multiple Line Terminal Adapter (MLTA) RPQ and the Multiline/Multipoint (MLMP) Binary Synchronous Communications programs for the Binary Synchronous Communications Adapter (BSCA) — whichever is required — both, if necessary. The CCP allows you to write communications application programs in COBOL, FORTRAN IV, RPG II, and Basic Assembler (or equivalent).

This publication guides you in designing a communications-based system using the IBM System/3 and the CCP and describes the following aspects of the CCP:

- Writing application programs to run under the CCP
- Generating the CCP system
- CCP operation
- Using the CCP, from the point of view of the system operator and terminal operators

Prerequisites

You are assumed to be familiar with basic computer and teleprocessing concepts, but you do not need extensive knowledge of or experience with either. Basic understanding of the following teleprocessing concepts will be especially helpful in reading this manual (all of these terms are defined in *IBM Data Processing Glossary*, GC20-1699):

- Multipoint networks
- Switched and nonswitched point-to-point networks
- Polling, selection, and addressing
- Binary synchronous transmission
- Start-stop transmission

Related Publications

Related publications are listed in *Appendix C: Bibliography.*

Note about Using this Manual

Chapter 1 is an introduction to the overall operation of and facilities provided by the CCP. The purpose of the *Introduction* is to:

- Provide a general understanding of the CCP to those who do not wish to read the entire manual.
- Provide the background knowledge required to understand the individual topics in this publication in their proper context. Terms and concepts introduced in chapter 1 are explained in detail later in the manual.

First Edition (December 1972)

Changes to the information herein are made periodically. Before using this publication to operate an IBM system, refer to the latest IBM System/3 Newsletter, GN20-2228, for the editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

Contents

The Communications Control Program (the CCP) is a system control programming feature that allows the Model 10 Disk System to support an online network of terminals. It enables terminals to call application programs as needed and permits those programs to access a common set of disk files. If sufficient main storage is available, the CCP permits several application programs to be executing concurrently, though independently of one another; that is, the CCP provides for multiprogramming. With the CCP, System/3 users have available, for the first time, those control program services needed to operate a communications-based information processing system.

The CCP is designed to make a communications-based system as easy and inexpensive as possible to establish and operate. The CCP can be tailored to suit diverse data processing environments involving batch and online applications.

Note: For a more basic introduction to the CCP that also contains background information concerning telecommunications, see *IBM System/3 Model 10 Disk System Communications Control Program General Information Manual*, GC21-7578. If you are not acquainted with the terms used in this introduction, you can find them explained either in the *General Information Manual*, in *Appendix B: Glossary* at the end of this manual, or in *Data Processing Glossary*, GC20-1699.

FACILITIES OFFERED BY THE CCP

The CCP performs the complex control program services which make the communications-based system easy to use for terminal operators, application programmers, and the system operator.

Control Program Services

The CCP performs four types of control program services:

- Task management
- Communications management
- Program management
- File management

Task Management: The CCP provides for *multiprogramming* within the program level in which it operates. That is, several user application programs, initiated independently of one another, can be executing concurrently under the CCP.

It is possible for two or more copies of the same application program to be executing concurrently and independently of one another under the CCP. Therefore, the term *task* is used to refer to each independently executing body of code. Each user program currently in execution under the CCP is considered a task; in the case of multiple copies of a program executing concurrently, *each* copy is considered a task.

The management of tasks, as performed by the CCP, permits each task to be initiated and terminated independently and permits those tasks to operate concurrently with the functional results of each identical to those which would have occurred had each task run alone. Task management implies the control, by the CCP, of access by each task to:

- Processing time
- Disk
- System services

Communications Management: The CCP controls input from and output to terminals in the system:

- Monitors terminals for input and selects them for output
- Manages input/output buffers and control blocks
- Provides for referencing terminals by symbolic names
- Provides services that allow application programs to be nearly independent of different requirements of individual terminal devices
- Translates data codes for both input and output
- Permits the testing of terminal while the system is operating

Program Management: The CCP handles requests from terminals and the system operator's console (5471 printer/keyboard) to execute application programs:

- Verifies that system resources required by the requested program — such as terminals, disk files, main storage — are available.
- Allocates the required resources.
- Loads the requested program from a disk library.
- Frees the resources used by the requested program when it terminates.
- Optionally, maintains a count of the requests for each program.

File Management: The CCP manages access by user programs to user data files:

- When the CCP is started, it allocates and opens all disk files needed by the programs which may run under the CCP.
- Controls access to unit record devices.
- Schedules I/O operations.
- In a multiprogramming CCP system, the CCP manages shared access to disk files by several concurrently running programs.

Programming Facilities

Programs that run under the CCP can be written in any of four programming languages:

- RPG II
- COBOL
- FORTRAN IV
- Basic Assembler

Although the design of programs written for the CCP may be different from those the programmer has been writing, the programming statements used for terminal input/output are already familiar to programmers:

- In COBOL or FORTRAN: the CALL statement
- In RPG II: either the EXIT operation or a SPECIAL file
- In Basic Assembler: a macro-instruction is provided that is processed by the Disk System Management Macros Feature

Along with each request for terminal I/O, the programmer provides a list of parameters that tell the CCP which specific operation to perform, which terminal to use, and what data area to use.

The CCP allows the programmer to identify terminals by symbolic names. If a particular terminal is unavailable for any reason, the system operator can reassign the symbolic name to a different terminal; thus, the program need not be changed or recompiled.

Other facilties offered by the CCP to the programmer are:

- Access to attributes of individual terminals
- Support for overlay programs
- Automatic translation of transmission data codes
- Dynamic, program controlled allocation and deallocation of terminals
- Access to communications I/O error or exception information

Terminal Operator Facilities

Under control of the CCP, the operator of a terminal can:

- Request programs
- Specify whether a program request should be rejected if the program cannot be executed immediately, or whether the request should be placed on a queue
- Specify the disk files to be used by a particular program or series of programs whose execution he requests
- Change the symbolic name or his terminal to one of a group of predetermined names
- Send a message to the system operator
- Cancel his communication with a program in order to enter another program request or command

In order to do any of the operations above, the terminal operator must first "sign on" to the CCP. A sign-on is a message initiated by the terminal operator signifying that he wishes to begin requesting services of the CCP. If the system has a password security feature (an option selected during CCP generation – see index entry *password security feature*), the terminal operator must correctly enter a password with his sign-on request.

Once a terminal operator is in communication with an application program *he requested*, he enters data, as required by the program. While sending data from his terminal to the application program, he can re-establish communication with the CCP in order to:

- Send a message to the system operator
- Resume sending data to the program
- Cancel the program

When a terminal operator has finished making a series of requests from services of the CCP, he will normally "sign off" his terminal. This action restores the terminal to an initial status, such that it must be signed on again (with a password, if that option was chosen) before it can request services of the CCP again.

System Operator Facilities

The system operator initiates and terminates the activity of the CCP and controls the operation of the communications-based system. After the CCP has been loaded into main storage, it asks the system operator one or more questions that allow the system operator to identify the *set* of files, programs, terminals, communication lines, and terminal names to be used by the CCP on the current run (one or more of these sets have been defined prior to the current run of the CCP – see index entry *assignment stage*). These questions allow the system operator to modify a selected set to suit a particular run of the CCP.

During the operation of the CCP, the system operator exercises his control over the system through the 5471 printer/keyboard. He can:

- Monitor the status of the system at any moment
- Determine the unfulfilled requests for programs or system operator replies in the system at any time
- Send messages to terminals
- Cancel, suspend, and resume activities of programs
- Change the actual terminal referenced by a terminal name
- Request the online test of terminals to determine whether they are operating correctly
- Cause the immediate or controlled termination of the CCP

The system operator can also perform some of the functions of a terminal operator. He can request programs, but he is restricted in the way he enters data for a program, since the console must be readily available for other system operating requirements.

Because of the extent of the control exercised by the system operator, he must be thoroughly trained in the operation of the CCP, applications of the CCP in his installation, and the specific tasks to be performed under control of the CCP.

RELATIONSHIP TO OTHER PROGRAMS

Disk System Management: The CCP uses the facilities of Disk System Management, including the I/O Supervisor for disk I/O devices.

The CCP can operate in either program level of a Dual Programming Feature (DPF) system, but not in both levels. While the CCP is operating in one level, the opposite level can contain a user application program operating under control of Disk System Management, if it does not use resources used by the CCP level.

The CCP code, including the incorporated IOCS code described below, occupies a program level's main storage area and is not part of the resident supervisor. Thus, before the CCP is started, or after it has been shut down, the CCP occupies no main storage, and the program level is free to be used in any way it might be used without the CCP.

Communications IOCS: System/3 programs that are currently available to support communication devices are not obsoleted by the CCP:

- MLMP IOCS for the BSCA (incorporated in the CCP if BSCA lines are used)
- MLTA IOCS (incorporated in the CCP if MLTA lines are used)
- RPG II Telecommunications Feature (RPG II support for batch terminals – not part of the CCP)
- Remote Job Entry Work Station Support (not part of the CCP)

The CCP incorporates the appropriate BSCA and MLTA IOCS routines, depending upon the communication adapters used and the line configuration of the system, and allows access to them from any user program. Another teleprocessing program not running under CCP control can be co-resident with the CCP in systems that have the Dual Programming Feature; however, it must be in the opposite program level from the CCP and its use of communication lines must not conflict with the CCP.

Telecommunications Application Programs: These programs operate under control of the CCP. They are loaded by the CCP and receive control from the CCP. Requests by these application programs for system services are received by the CCP. Some of the requests are performed by the CCP; some are passed from the CCP to disk system management to be performed.

DEVICES AND PROGRAMS SUPPORTED

The terminal devices, system devices, and system programs required and supported by the CCP are listed in *Appendix A.*

ESTABLISHING AND OPERATING THE CCP

The CCP can be tailored to suit each unique operating environment. Establishing and operating the CCP in a particular environment is accomplished in three stages:

- Generation
- Assignment
- Operation

Generation Stage

CCP generation is the process by which a user creates his individual version of the CCP. The purpose of generation is to create a set of CCP object modules and subroutines, unique to that user's requirements, on the user's disk pack. The process of generation involves:

1. Describing the type of equipment to be used by the communications system, the maximum number of concurrent user programs possible, disk files, and related information.
2. Creating a set of control routines whose specific content may be unique to the user's installation.
3. Joining the routines by a linkage edit process.
4. Copying appropriate additional supporting routines.
5. Initializing the control file which the assignment stage and the operational stage use.

Assignment Stage

Assignment is a special, brief CCP run during which the user specifies one or more *sets* of specific environments in which the CCP can run. Each set includes:

- Specific items of information pertaining to the entire CCP, such as the current password
- Programs that may be run under the CCP and the resources each requires
- Files that are accessible to each program
- The current line/terminal configuration
- Symbolic terminal names and the actual terminals to which they apply

The assignment run need be repeated only when the user wishes to change some of the specific information given in a previous assignment run.

Operational Stage

The operational stage begins with *operational startup*, when the CCP is loaded into main storage. During startup, CCP routines open disk files, adapters, and communication lines and complete various tables and control blocks. During *operation,* the CCP is performing its functions of communications management, program management, file management, and task management described earlier (see *Facilities Offered by the CCP*). The operational stage is concluded by *shutdown,* which is initiated by the system operator. During shutdown, the CCP allows currently executing programs to complete processing, then closes communication lines, adapters, and files.

This chapter introduces you to the factors you must consider in designing a communications-based information processing system using the System/3 Model 10 Disk System and the CCP. You must regard the CCP and the communications system as a means to an end, the end being increased accuracy and faster flow of information, greater efficiency in the organization, and increased volume of work. During the preinstallation activity, you must define the overall objectives of the communications system, define the requirements of all departments that will use the system, and produce a detailed plan for preinstallation and installation activity. You should plan applications, use of terminals, data files, programs, and equipment needs prior to installation of the CCP to speed the installation of the CCP and reduce errors. IBM systems engineering aid can be helpful in this activity.

Note: Publications referenced in this chapter and elsewhere in this manual should not be considered a complete bibliography for designing a communications-based system. Many publications are available to describe in detail the design factors summarized in this chapter. IBM Systems Engineers and Marketing Representatives can be of assistance in obtaining publications describing the terminals that can be attached to System/3 via the BSCA and MLTA and concerning systems design. They can also assist in arranging education classes concerning System/3 communications system design. Although many publications currently available are oriented toward larger systems and applications (such as airlines reservations systems), the basic types of applications and techniques of data communication also apply to System/3 with the CCP.

APPLICATIONS

The basic element in any system design process is determining what the applications of the system will be. You probably have already determined that you have a need to perform one or more information processing jobs more accurately and efficiently. For example, perhaps the flow of information and the processing required to perform weekly payroll for a growing number of employees in scattered locations performing different jobs has become inefficient. An information bottleneck has developed in the central payroll office. A network of terminals, communicating the payroll information to the central processor in the payroll office will eliminate the bottleneck, allowing payroll data to be communicated as soon as it is available, to be processed immediately. Payroll inquiries from the remote locations can also be processed immediately.

Assume payroll is identified as an application for your communications-based system. The next step is to determine what related applications can be performed by your system. Perhaps you have a need for more immediate processing of personnel information, which is closely related to the payroll information. Information in the personnel files maintained by the central processor can also be made available to inquiries from the remote terminals. Perhaps you can use the production totals for individuals in the separate work areas in production accounting.

TERMINALS

When you have identified a major application and related applications, you can determine preliminary locations for terminals. Perhaps you locate separate terminals in a manufacturing area, an assembly area, a warehouse area, a shipping and receiving area, a sales office, and the central business office. When you have determined the preliminary locations, you can consider the other possible uses for the terminal in each location. In the manufacturing and assembly areas, perhaps you have a need for parts control; in the warehouse area you may have a need for inventory management; in the shipping and receiving area, a need for a shipping order and invoice processing; in the sales office, a need for purchase order and service order processing and sales analysis; in the central office, accounts receivable, billing, and general accounting.

The possibilities for applications in any kind of organization are many. A terminal in one location may serve more than one application.

In choosing terminals for different locations and uses, you should consider the following:

- Is the terminal to be shared by operators with different requirements? – if so, the terminal type must be compatible with all requirements.

- Is a heavy workload expected? – if so, perhaps more than one terminal is required at the location or a faster line speed is required (line types are described in *IBM General Information – Binary Synchronous Communications*, GA27-3004).

- Will a display-type terminal (IBM 3270) or a typewriter terminal be needed (all terminals on the same multipoint line must be compatible)?

- Will a high volume of activity of the terminal justify special features on terminals, such as the buffer-receive feature on the IBM 2740, Model 2.

Note: Uses for terminals in a communications-based system (data entry, inquiry, inquiry-with-update) are defined and described briefly in the *General Information Manual*, GC21-7578.

DATA FILES

When applications of the communications-based system have been determined, plans must be made for the data files to support those applications. The basic decisions to be made initially are:

- What separate files are needed?

- How should the files be organized to best satisfy the different uses to which the files will be put?

Many applications require separate files containing current information, todate information, and historical information. In a payroll application, for example, the following files might be required:

- A file of daily information (hours worked, production, etc.)

- A file containing necessary information about each employee and the major todate information

- History file, containing employees payroll records for previous years

Other ways of differentiating between files could be:

- Separate files for separate branches of an organization, such as schools in a school system

- Separate files for different product classes

After you have identified the separate files you need, you must find the best file organization for each file according to its use. For example, files that are normally used for online processing may be subject to batch processing when the files are loaded. Analyzing the percentage of online processing time versus the percentage of batch processing time will aid in selecting the file organization that will be most efficient overall. For example, if processing is 90% online and 10% batch, your choice of file organization should be weighted toward direct organization lif you can devise and efficient method of deriving relative record numbers; see *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571, for a description of direct files). If processing is 50% online and 50% batch, indexed organization is probably the best compromise organization. Perhaps you will use the file for online processing in one partition of a system with the Dual Programming Feature and for batch processing in the other partition. In that case, either direct or indexed organization might apply, since both can be porcessed either randomly or consecutively.

The *IBM System/3 Disk Concepts and Planning Guide*, GC21-7571, contains information which will aid you in choosing a file organization and planning disk files.

PROGRAMS

You must also plan how your applications are to be performed by your communications programs. Related applications can be performed by a single program or by separate programs. In some cases, it might be best for you to structure programs into overlays, perhaps with a root segment (remains in storage throughout the execution of the program) and separate overlays to perform related functions. You should consider the provision of the CCP for physical files and symbolic files in applications that involve processing different files on different runs (see index entry *symbolic files*).

For information on program structure and overlays, see the following publications:

- *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561
- *IBM System/3 Subset American National Standard COBOL Compiler and Library Programmer's Guide*, SC28-6459
- *IBM System/3 Disk System RPG II Reference Manual*, SC21-7504
- *IBM System/3 Disk FORTRAN IV Reference Manual*, SC21-6874
- *IBM System/3 Disk System Basic Assembler Program Reference Manual*, SC21-7509

In designing a program to run under the CCP, you must consider the program's use of terminals. Should the program service a request from one terminal at a time or from multiple terminals? Should terminals be selected by the program or should individual terminals request the program when they need it? Perhaps there are security considerations that indicate the program should have a single requestor or a limited number of requestors. How long will the program reamin in main storage? Is it a brief inquiry application or a more time-consuming, data-entry application? If the program will be used frequently, perhaps multiple requesters should be allowed, or perhaps the program should be written as a serially reusable or a never-ending program. Program design under the CCP is described in *Chapter 3, Writing Communications Programs.*

If you will run applications under the CCP concurrently, you must plan the programs so they will run smoothly together, especially during peak times. Programs running together cannot, for example, each require dedicated use of unit record devices. Perhaps you should run batch programs only during particular times of the day. You must plan peak processing times so that system resources will be available to the programs that must execute.

You should test individual programs and the entire system in advance to ensure that all applications execute as planned and to evaluate the performance of the system against its planned performance. If the system does not perform as planned, review your program structures, file organizations, and placement of files and programs on disk.

UPDATING THE SYSTEM

In time, the uses of the communications system may change. You must plan for possible updating and additional tailoring of the system as you gain experience with the system. You should make allowance, for example, for the "turnpike effect," a phenomenon observed after the first modern super-highways were planned and built. Use of the new highways was greater than anticipated, since drivers tended to stop using the old routes in favor of the new highway. Overall traffic flow increased beyond expectations because of the convenient new highway. The turnpike effect has been observed in previous communications systems and is a factor to consider in planning for a System/3 communication-based system.

TOTAL EQUIPMENT NEEDS

When you have considered all factors — applications, use of terminals, data files, programs, and provisions for system updating — you can make decisions concerning the total equipment needs of your organization:

- How much main storage is required?
- Is a DPF system required? Compilers and disk system management programs whose names start with $, such as utilities, and application programs not designed to be run under the CCP must not operate under the CCP. In a DPF system, some of these programs can be run in the opposite program level (if they do not require dedicated use of the system).
- How much disk storage is needed — of what type? How much space is needed for libraries, how much for files? The IBM 5445 Disk Storage Drive can be used for files, but not for libraries. Detailed storage estimates for CCP modules will be provided in a later publication.
- What terminals are needed?
- What communications equipment and lines are needed?
- What is the total cost?
- When can deliveries be made?
- What unit record devices are needed — how fast should they be?

Complete descriptions of the IBM terminals available under the CCP are contained in the manuals listed in *Appendix C: Bibliography.* Information concerning Teleprocessing equipment characteristics communications concepts, common carriers, network design, and other useful information is contained in the following publications:

- *IBM Data Communications Primer*, C20-1668
- *IBM System/360 Introduction to Teleprocessing*, C30-2007

Chapter 3: Writing Communications Programs

CCP services enable a programmer to write communications programs (programs that communicate with one or more online terminals) in three high-level programming languages – COBOL, FORTRAN IV, RPG II – and Basic Assembler. The complexities of the telecommunications environment are managed for the programmer by the CCP. The programmer need not be concerned with the fact that his program might be competing with other programs for system resources such as terminals, disk files, and unit record devices; the CCP allocates these resources to the program before it allows the program to execute.

Each terminal allocated to a program is available exclusively to that program until the program releases it (see index entry *Release Terminal Operation*) or until the execution of the program has ended. When either of these events has occured, that terminal is free to be allocated to another program or to enter commands (see index entry *command terminal*).

Because the CCP also allocates the use of unit record devices to a program and manages the use of disk devices, the programmer can code I/O operations using these devices as though his program has exclusive control of the devices. See *Chapter 2: Designing Your Communications-Based System* for considerations in planning the programs to run under the CCP.

Although the CCP performs many services for the programmer, writing communications programs is different from writing programs that do not communicate with terminals in two important aspects:

1. The manner of requesting terminal operations
2. Typical design of programs that communicate with terminals

Note: In this discussion, *terminal* refers to terminal I/O devices and attached host and subhost systems.

REQUESTING TERMINAL OPERATIONS

Program requests for terminal operations are made using statements of a type already available in COBOL, FORTRAN IV, RPG II, and Basic Assembler. The programmer supplies a *parameter list* with each request to describe the details of the operation to be performed (see *Parameter List*). In addition, the programmer provides a *record area* in which he identifies the terminal upon which the operation is to be performed and which, if the operation involves the transfer of data to or from the terminal, contains the output data or space for the input data (see *Record Area*).

Since the programming languages do not include special statement types for terminal operations, the CCP supplies a *communications service subroutine* to each user program (except Basic Assembler programs, for which a macro-instruction is provided that is processed by the System/3 Macros Feature). This subroutine translates the program's request into a standard request to the CCP communication facilities.

Terminal operations are requested as follows (see *Examples of Programming for Terminal Operations*, later in this section, for examples of requests in each language):

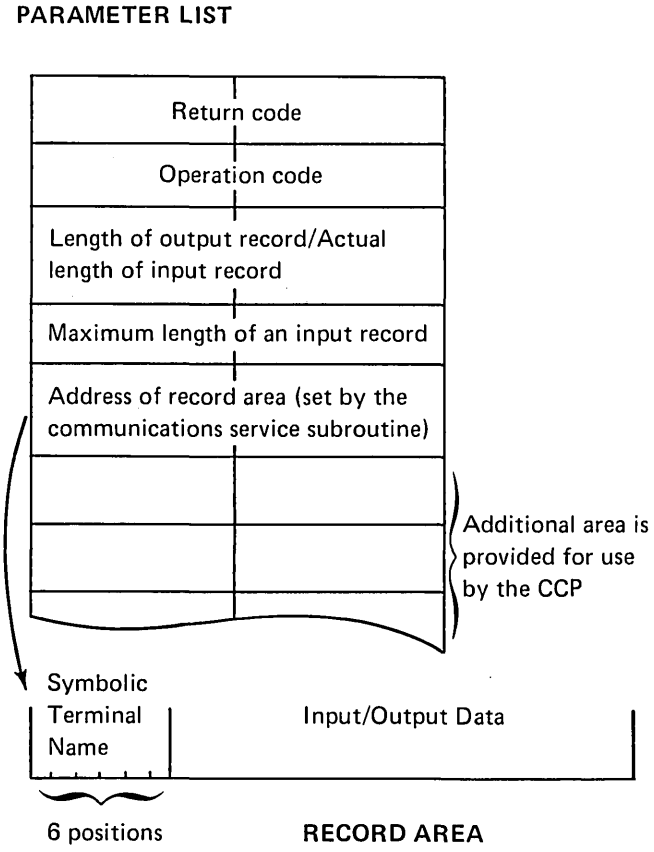
- *COBOL and FORTRAN IV:* The program issues a CALL statement to the communications service subroutine and provides a parameter list.
- *RPG II:* Terminal operations can be requested in two ways: (1) by defining a SPECIAL file for each communications file type used (Input, Output, Combined) and an array for each SPECIAL file to contain the parameter list; (2) using EXIT and RLABL operations to exit to the communications service subroutine and define the parameter list.
- *Basic Assembler:* The program provides a parameter list and record area. Through the use of a macro-instruction (processed by the System/3 Macros Feature), the program: (1) sets the necessary control information in the parameter list and record area and (2) branches to the proper CCP routine to cause the operation to be performed.

Record Area

The programmer supplies a record area in his program (Figure 1) with each terminal operation. The record area always contains, in its first six positions, the identification of the terminal upon which the operation is performed. The remainder of the record area is used in operations which involve the transfer of data into or from the user program. On input operations, the CCP translates the input data to EBCDIC, if not already in that code, and makes it available to the user program in its record area.

On output operations, the data to be sent to a terminal is provided by the user program in its record area in EBCDIC; the CCP performs any required translation to transmission data code. If the user desires, translation between EBCDIC and transmission data code can be suppressed.

RPG II programmers define a record area within their programs when using a SPECIAL file to communicate with terminals by specifying a record and block length in File Description specifications.



Note: In RPG II, the format of the parameter list and record area are somewhat different. See *Examples of Programming for Communications Operations* for the format of the RPG II parameter list and record area.

Figure 1. Data Areas Provided Within User Programs for Communications Operations

Line control characters are always removed from input data by the CCP before the data is passed to the user program. Similarly, line control characters are added to data by the CCP before data is transmitted. Device-dependent control characters such as tab characters are the programmer's responsibility; however, the CCP will include carriage return and idle characters in output data to MLTA typewriter terminals unless the programmer specifies otherwise (see index entry *operation code modifiers*). On input, lowercase characters are translated to equivalent uppercase EBCDIC characters unless the programmer specifies otherwise.

When input records are received that are shorter than the input length specified in the user program, the CCP places blanks in the remainder of the record area before the record is made available to the program. When input records are received that are longer than the input length specified in the user program, only those initial positions of the input record up to the specified length are placed in the record area; an exception condition is communicated to the user program in the return code area of the parameter list.

Symbolic Terminal Name

In all communications operations, the first six positions of the record area are reserved for the symbolic name of the terminal (or the system operator's console) with which the operation is performed. In most operations, the name is placed there by the programmer to specify the terminal with which to operate; in certain operations, however, the name is placed there by the CCP to inform the programmer of the terminal with which the operation took place.

Each terminal is assigned a symbolic name during the CCP assignment stage (see index entry *TERMNAME statement*); the system operator's console always has the preassigned name, CONSOL. The use of symbolic names for terminals allows programs to be relatively independent of the specific terminals with which they are communicating. However, the programmer must be aware of the type of terminal he is using since he must know the record length of the device; whether the terminal is capable of input only, output only, or both input and output; and other information (see index entry *Get Attributes*). The system operator can reassign symbolic names to different terminals during operation of the CCP (perhaps the terminal is out of order or offline, see index entry *offline*), to avoid preventing the execution of the programs using those names.

Parameter List

The parameter list supplied by the programmer for each communication operation includes the following (see Figure 1):

Return Code Area: The programmer must supply this area in the parameter list, but the CCP ignores its contents at the beginning of an operation. The CCP puts information about the results of an operation into the return code area after the operation is performed to inform the program:

- Whether the operation completed normally
- Whether the operation resulted in an error
- Whether the operation resulted in some exception condition

The specific return codes will be listed and explained in detail in a later publication.

Operation Code: See *Operations*, later in this section.

Length of Output Record/Actual Length of Input Record: On output operations, the parameter list gives the actual length of the data. On input operations, the CCP places the actual length of the input data in this area. Neither length includes the six positions for the symbolic terminal name or the four additional positions reserved in RPG II for the return code.

Maximum Length of an Input Record: On an input operation, the programmer supplies the maximum length of input records the user program will accept from a terminal. This length does not include the six positions for the symbolic terminal name or the four additional positions reserved in RPG II for the return code.

Address of the Record Area: The programmer causes the address of his record area to be placed in the parameter list. The method by which this is done varies among the programming languages. Except in Basic Assembler, the communications service subroutines actually sets this address in the parameter list. The address is that of the first character of the symbolic terminal name; therefore, the data actually begins at the address given, plus six (plus ten in RPG II).

Note: Exceptions to the parameter list contents for specific operations and the form of the parameter list for each language will be described in a later publication.

Operations

The programmer specifies which operation he wants to perform by means of an operation code in the parameter list. Using basic operation codes and operation code modifiers, the programmer can do a variety of operations with terminals.

Input Operations

See *MLTA/BSCA Communications*, later in this chapter, for a specific description of how the input operations apply to each type of transmission.

Get Specific: This is a request from the program for data from the terminal whose name is specified in the record area. The program does not continue until the operation is complete, that is, until the CCP returns data from that terminal in the record area and provides a return code.

Invite Input: This operation is a request to allow the terminal whose name is specified in the record area to begin sending input data. The program continues execution immediately upon acceptance of the request by the CCP; this operation does not make the input it requests available to the program. The Invite Input operations is used in conjunction with the Get General operation (see below), which waits for completion of an input and makes data available to the program.

The program specifies in the parameter list the maximum input length it will accept from the selected terminal. Since input data is not passed to the program by this operation, the record area specified for this operation need only include the symbolic terminal name.

An Invite Input may be issued to more than one terminal. Once an Invite Input has been issued to a particular terminal, the next operation issued for that terminal may be only a Get General or a Stop Invite Input operation (see below).

Get General: This operation is a request from the program to gain access to input data sent by any terminal to which an Invite Input was previously issued. If input was invited from several terminals, the operation provides the earliest completed record to the program. The program does not continue until a record has been presented by one of those terminals. As a result of a Get General, the program receives:

- A symbolic terminal name in the record area, identifying the terminal from which the data was received
- An input record in the record area
- A return code
- The actual length of the input data, in the parameter list

The completion of a Get General operation makes input data available from the first terminal which has completed input among the set of terminals from which input has been invited. The Invite Input to that terminal is satisfied by the Get General; input is no longer solicited from that terminal until another Invite Input is issued to it. However, Invite Input operations that were outstanding to other terminals when the Get General was issued remain in effect; each of those inputs is eligible for presentation to the program on subsequent Get General operations. See *Chapter 2: Designing Communications Programs* in this chapter for illustrations of the use of Invite Input and Get General.

An Invite Input is not needed prior to the first Get General in a program if the program permits data to be entered along with the program request from the terminal (see index entry *Program Request command*).

Stop Invite Input: This operation attempts to cancel a previous Invite Input operation issued to the terminal identified in this operation. This operation is used when some event has occurred in the program such that the program no longer wants input from that terminal, If data was already received from the terminal, the program is notified that data is available and the data is moved into the program's record area, The program must be prepared to process data received before the Stop Invite Input was issued. The operation is therefore identical to a Get Specific, if the invited input cannot be stopped.

Output Operations

See *MLTA/BSCA Communications*, later in this chapter, for a specific description of how the output operations apply to each type of transmission.

Put: This is a request from the program to write output data to a particular terminal. The program must supply the name of the terminal in the first six bytes of the record area. The program does not continue until the CCP determines the result of the operation and supplies the appropriate return code.

Put-Then-Get: This is a request from the program to write output data to a specific terminal and then get input data from the same terminal. The program does not continue until the reply from the terminal is complete. This operation has the same effect as a Put operation followed by a Get Specific operation to the same terminal. The same record area is used for input and output.

Put-No-Wait: This is a request from the program to write output data to a specific terminal. The program must specify the terminal name, record area, and parameter list. The Put-No-Wait operation allows overlap of the output operation with continued program execution. The user program regains control as soon as the CCP accepts the operation. The program cannot check whether the operation completed successfully at the terminal.

Non-I/O Operations

Get Attributes: This is a request from the program for information about a specific terminal. The following information is returned in the record area in encoded form:

- Terminal type (2740 Model 1, 2740 Model 2, 3270, etc.)
- Whether the terminal is already allocated to a program
- Location of an area that contains other specific information, such as the buffer length required for the terminal and the line number to which the terminal is attached

Acquire Terminal: This is a request from a program to allocate a terminal to that program. The program must specify the terminal name and a parameter list. This operation allows a program to allocate terminals to itself while it is running. The operation results either in the program gaining use of the terminal or in a notification that the terminal is not available to his program. (See index entry *program-selected terminals* for additional information.

Release Terminal: This operation releases the terminal from control of the program so that it is available to other programs. The operation is valid for requester terminals and program-selected terminals (see index entry *terminal types*). When this operation is issued, a return code indicates whether there are outstanding invite inputs for this program.

Operation Code Modifiers

The programmer can specify operation code modifiers to override aspects of the normal operation performed by the CCP. Normal operations can be modified as follows:

- On output operations to typewriter terminals on MLTA lines, the CCP normally supplies the required control information for carriage-return and idle characters at the end of each output line. These operations can be modified so the CCP does not supply carriage control information.

- On output operations to typewriter terminals on MLTA lines, the CCP normally assumes that the output data begins on a new line by inserting carriage-return and idle characters at the beginning of the data, if necessary. These operations can be modified so that the new line is not forced.

- The CCP normally performs translation of data codes on input and output operations if the transmission data code is different from EBCDIC. Translation is always to uppercase. The programmer can modify input and output operations so translation does not occur, but he then must be prepared to deal with data in non-EBCDIC.

- When the CCP translates input data, it normally transforms lowercase alphabetic characters into the corresponding uppercase characters. The programmer can modify input operations so this transformation to uppercase does not occur.

MLTA/BSCA Communications

Data communications between programs running under the CCP and MLTA terminals differs from data communication with BSCA terminals. The differences affect how the program must provide for transmission and receipt of data from the different terminals and what types of I/O operations are performed.

Note: In this discussion, the term "MLTA terminals" refers to any of the terminals listed in *Appendix A* as supported by the multiple line terminal adapter (MLTA) RPQ. MLTA terminals perform asynchronous (start/stop) communications with programs through the CCP and the MLTA input/output control system (IOCS), which is included in the generated CCP if MLTA terminals are to be used. See *IBM System/3 Model 10 Disk System Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual*, GC21-7560, for a complete description of the MLTA IOCS.

The term "BSCA terminals" refers to any of the terminals (including host and subhost systems) listed in *Appendix A* as supported by the binary synchronous communications adapter (BSCA). BSCA terminals perform binary synchronous communications with the Model 10 Disk System through the CCP and the multiline/multipoint (MLMP) BSCA IOCS, which is included in the generated CCP if BSCA terminals are to be used. See *IBM System/3 Model 10 Disk System Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573, for a complete description of the MLMP IOCS.

Additional information regarding asynchronous and binary synchronous communications can be found in publications listed in *Appendix C: Bibliography.*

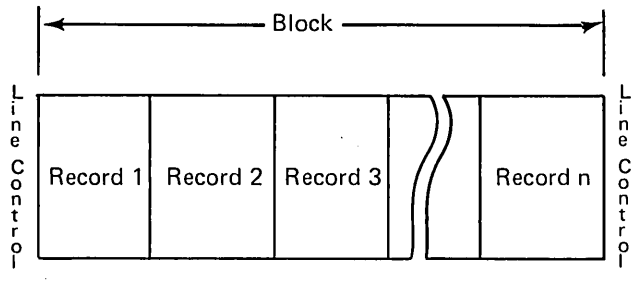
Communicating with MLTA Terminals

Programs communicate with MLTA terminals in a record-by-record manner; that is, each I/O operation in a program results in a record being sent or received. The program has effective control of the line only while a record is being sent or received. When the record has been communicated, another program and/or terminal can use the line. The communication operations and modifiers apply as described earlier in this chapter (see *Operations*) to communication with MLTA terminals. Line control characters are inserted into data to be sent to MLTA terminals by the MLTA IOCS; the programmer neither provides space for line control in the program's record area nor manipulates line control characters in his program.

Communicating with BSCA Terminals

When communicating with BSCA terminals, programs send or receive *blocks* of data. A block is the physical unit of data that is actually sent or received in each individual transmission on a BSCA line. Each block of data is preceded and followed by the line control characters necessary to control the transmission of data on the BSCA line (see data-link control in *General Information – Binary Synchronous Communications*, GA27-3004).

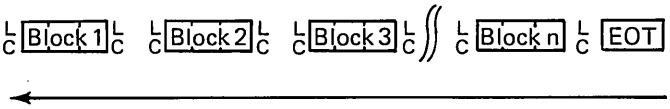
A block of data can be composed of one or more data records (Figure 2). Collecting records into blocks saves time when similar operations are performed on each record, since it is faster to send and receive some multiple of records at a time than to send and receive records individually.



In binary synchronous communications, a block of data can contain one or more records.

Figure 2. Blocking in Binary Synchronous Communications

A program communicating with a BSCA terminal continues to process input data until it receives an end-of-transmission (EOT) signal from the terminal (Figure 3). The EOT signal indicates the terminal has completed its current transmission. Likewise, a program must send an EOT signal when it has finished transmitting to a BSCA terminal.



In binary synchronous communications, each block of data is transmitted separately. The program retains control of the line until EOT indicates that all transmission is complete.

Legend: ← = Direction of transmission
LC = Line control characters

Figure 3. Data Transmission on BSCA Lines

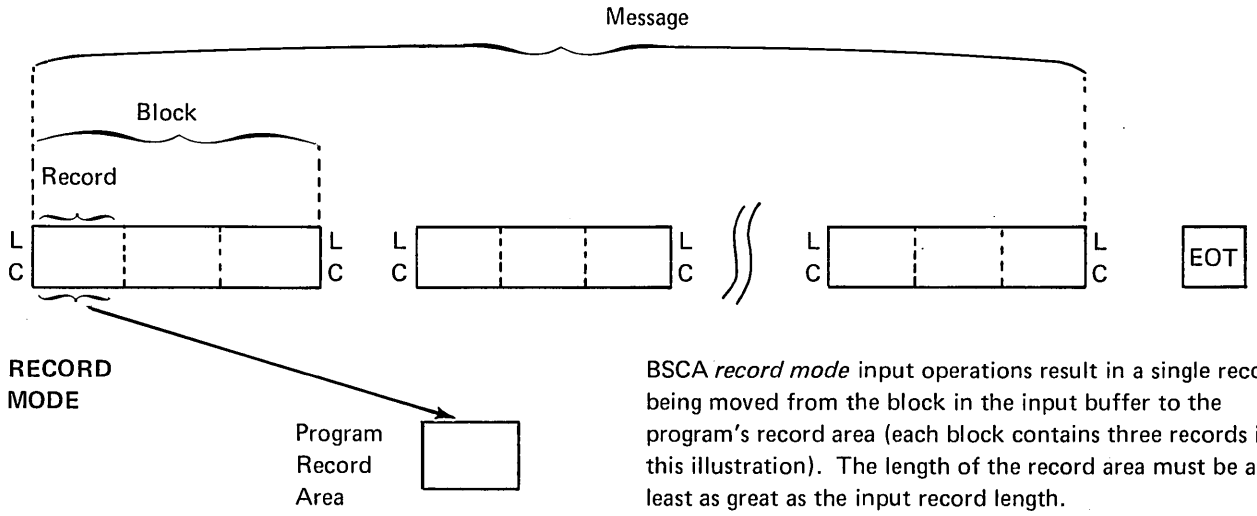
A BSCA line is dedicated to the program once communication is initiated and is not freed for use by another program or terminal until EOT is transmitted. A program that is receiving data from a BSCA terminal cannot transmit data to the terminal or communicate with any other terminal on that line until the terminal sends EOT. Likewise, when a program is transmitting to a terminal on a BSCA line, that line cannot be used by any other program or terminal.

BSCA Input Operations

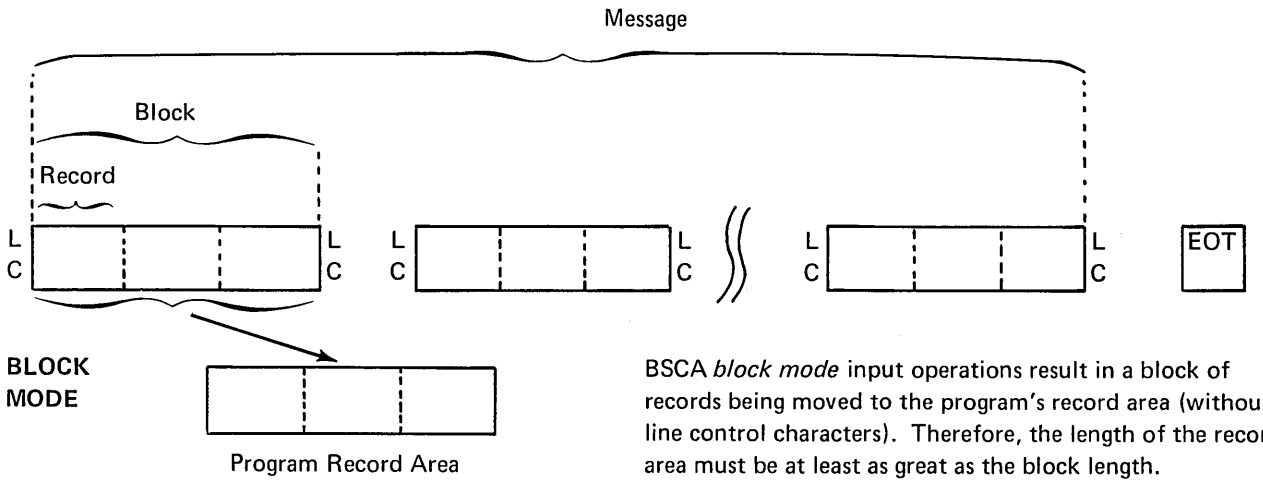
The CCP provides three levels (modes) of input operations for communication with BSCA terminals: *record mode, block mode,* and *message mode.* The mode of input used by a program with a terminal is specified during the CCP assignment stage. The actual input operations are used as described under *Operations*, earlier in this chapter.

Records and blocks have already been described; a *message* consists of a limited number of blocks of data, followed by an EOT, that constitute a complete span of information that can be received by a program as the result of a single Get operation. In message mode input operations, the CCP attempts to read all input data until it receives EOT before moving the data to the program's record area. In this way, the BSCA line is freed for use by another terminal as quickly as possible. Thus, message mode should be used only when a limited quantity of data is expected (ideally, a single block) on each input operation.

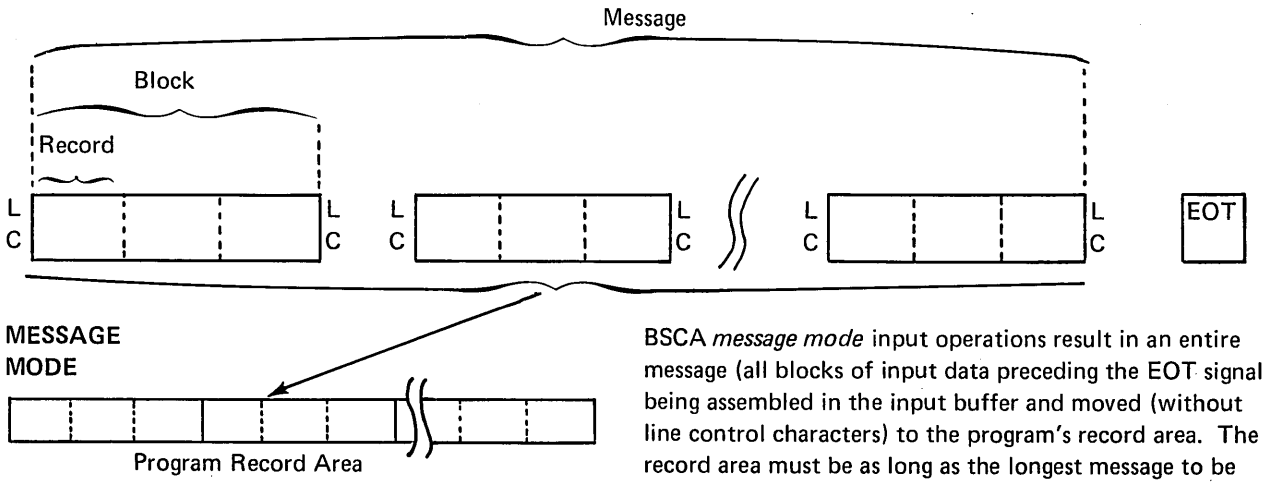
Figure 4 shows the results of input operations in record mode, block mode, and message mode.



BSCA *record mode* input operations result in a single record being moved from the block in the input buffer to the program's record area (each block contains three records in this illustration). The length of the record area must be at least as great as the input record length.



BSCA *block mode* input operations result in a block of records being moved to the program's record area (without line control characters). Therefore, the length of the record area must be at least as great as the block length.



BSCA *message mode* input operations result in an entire message (all blocks of input data preceding the EOT signal) being assembled in the input buffer and moved (without line control characters) to the program's record area. The record area must be as long as the longest message to be received, or the excess portion of the message is dropped.

Figure 4. BSCA Input Operation Modes

BSCA Output Operations

The CCP provides three types of Put operations for use with BSCA terminals (in addition to Put-Then-Get and Put-No-Wait operations): *Put Record, Put Block,* and *Put Message.* Use of these operations in a program is not restricted by the program's mode of input operations (see *BSCA Input Operations*).

Put Record: The Put Record operation is used to send a record to the terminal specified in the program's record area. If block length equals record length, each Put Record operation results in a record being transmitted on the BSCA line. If each block can contain several records, the block is transmitted when it does not have space for another record. Thus, several Put Record operations may be performed by the program before a block of data is transmitted. In order to send EOT following Put Record operations, the program must issue a Put Message operation (see *Put Message*).

Put Block: The Put Block operation causes the current block in the output buffer to be transmitted, whether or nor the block contains all the records it can hold. The next record Put by the program starts a new block. A Put Block operation may either be:

- Accompanied by a final record to be placed in the block before it is sent, or
- Issued with a record length of zero, which simply causes the block to be sent.

Put Message: Put Message causes all data to be transmitted, followed by an EOT. A Put Message operation may either be:

- Accompanied by the final data to be sent before EOT, or
- Issued with a message length of zero, which simply sends the EOT signal to the terminal.

Put-Then-Get and Put-No-Wait Operations: These operations have the same basic function as described earlier in this chapter (see *Operations*). Put-Then-Get causes data (record, block, or message) to be transmitted to a specific terminal, followed by EOT and a Get operation for the terminal. The result of the Get operation depends on the mode of input specified at assignment time; the result may be the equivalent of a Get Record, Get Block, or Get Message to the terminal. Put-No-Wait can be issued at the record, block, or message level; the result is the same as Put Record, Put Block, or Put Message, except that the program neither waits for nor receives a return code.

Switched Lines

A *switched line* is a communication line in which the connection between the system and the terminal is established by dialing. After the connection is completed, data can be transmitted. Disconnecting a terminal on a switched line is a function of the Release Terminal operation in the program (see *Operations*). This operation can specify whether to keep the line allocated to the program or to "return" it to the CCP.

Under the CCP, a data set (telephone) number must be established at assignment time for each terminal name assigned to a terminal that might be called by an application program running under the CCP. Also, at assignment time, each symbolic terminal name is assigned an attribute of auto/manual call and auto/manual answer.

Auto Call

If a terminal is defined as auto call, an I/O operation to the terminal name in a program causes the CCP to place the call automatically. Auto call cannot be used with MLTA terminals. In order to use auto call on BSCA terminals, the Auto Call feature must be installed on the BSCA hardware.

Manual Call

A program that uses a manual call terminal must be requested by the system operator or by another terminal already in communication with the CCP. An I/O operation to a manual call terminal in the program causes the CCP to inform the system operator that he must dial a certain data set number.

Auto Answer

In auto answer, the system verifies the line connection and the CCP waits for a call from the terminal.

Manual Answer

In manual answer, the system operator answers a call from a terminal. To answer a manual call, the system operator and terminal operator place their data sets in data mode; the system then verifies the line connection and the CCP waits for input from the terminal.

Examples of Programming for Communications Operations

Figures 5 through 8 are examples of programming for CCP communications operations in the System/3 programming languages. The examples show methods of defining the record area and parameter list in each language and how those data areas are used in requesting terminal operations.

Figures 5 through 7 illustrate terminal operations in COBOL, FORTRAN IV, and Basic Assembler, respectively. Each example is a partial program that Puts a 50-byte message to a terminal and then Gets a response up to 50 characters long from the same terminal.

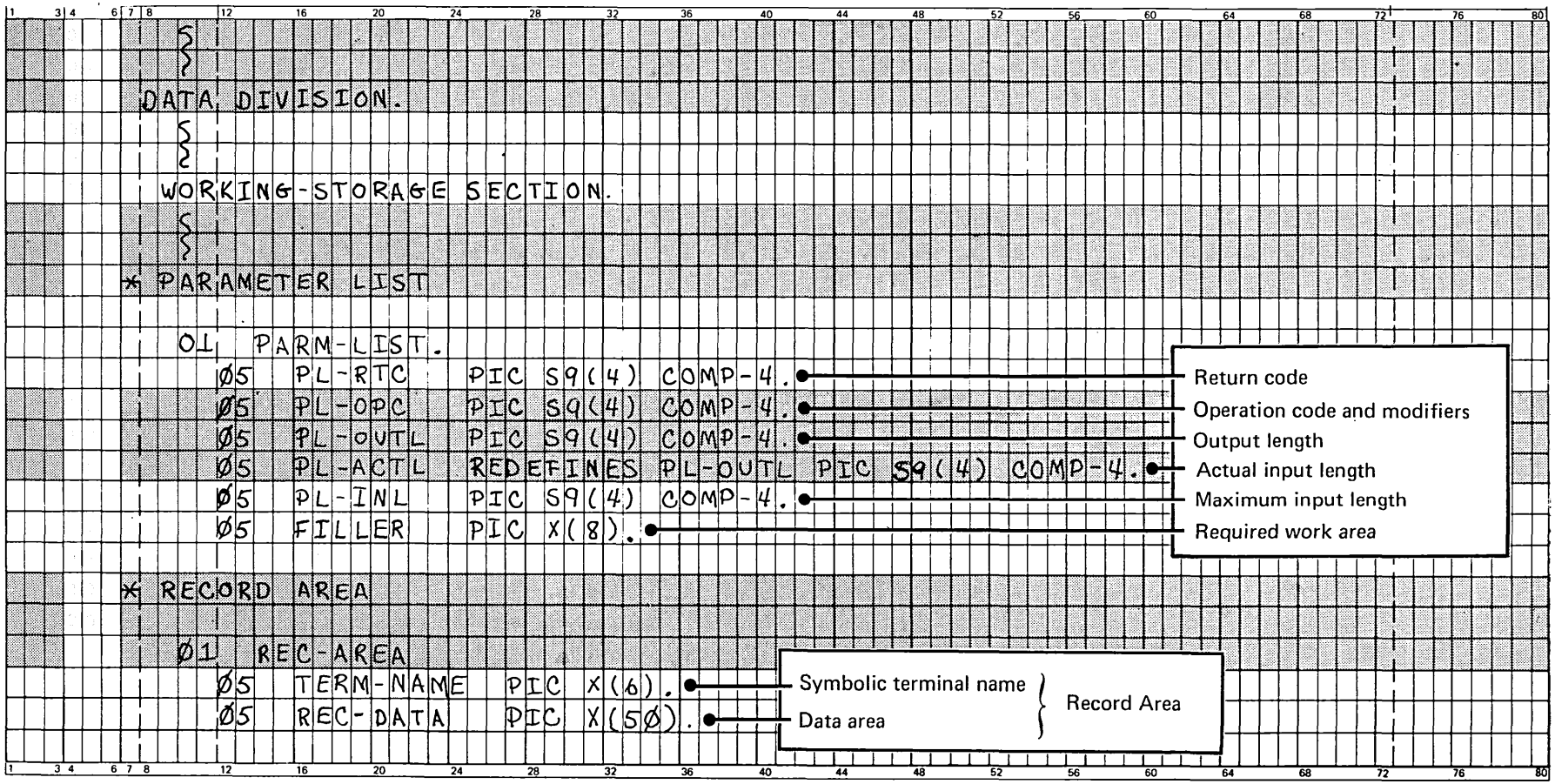


Figure 5 (Part 1 of 2). COBOL CCP Communications Programming Example

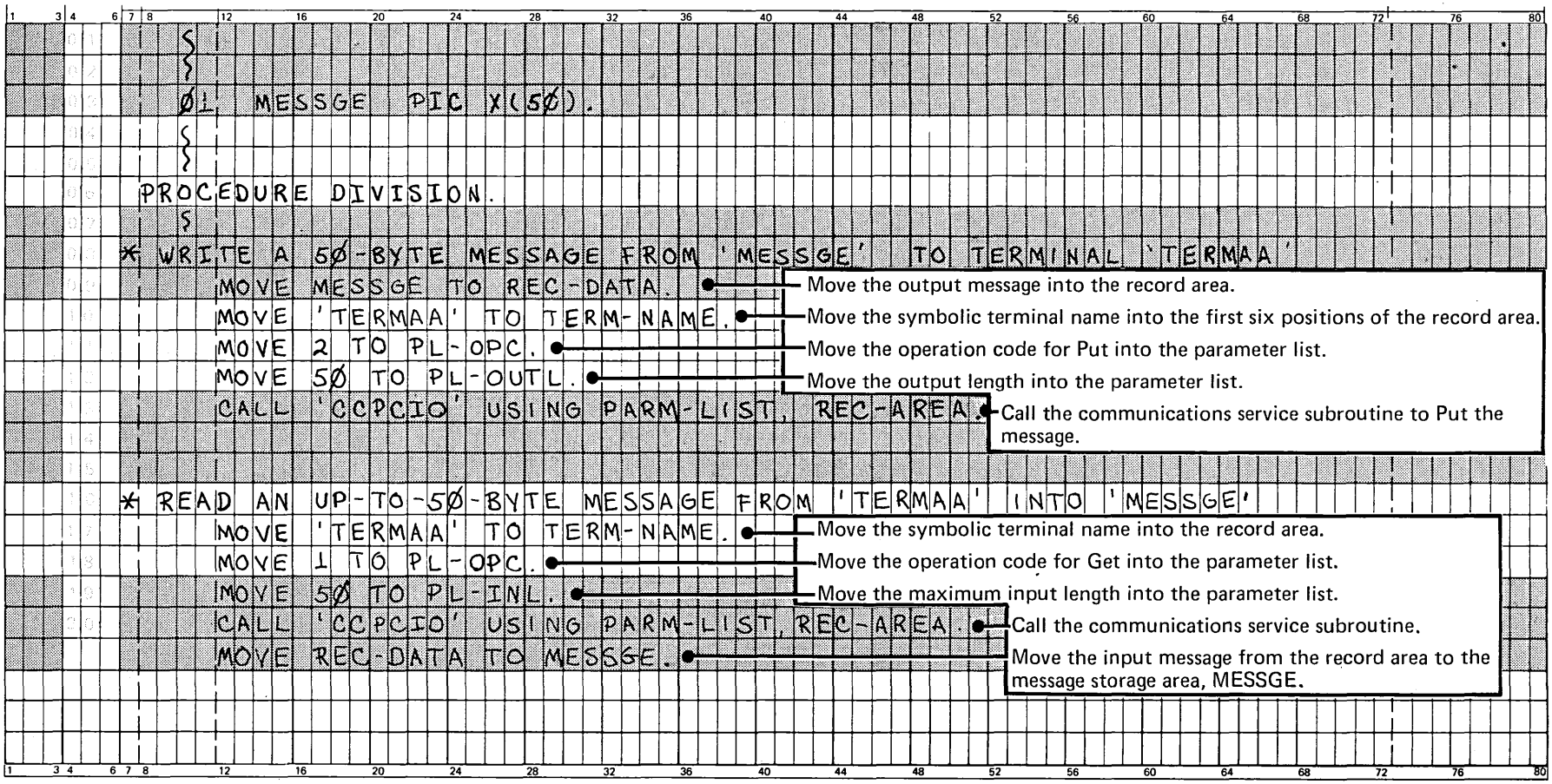


Figure 5 (Part 2 of 2). COBOL CCP Communications Programming Example

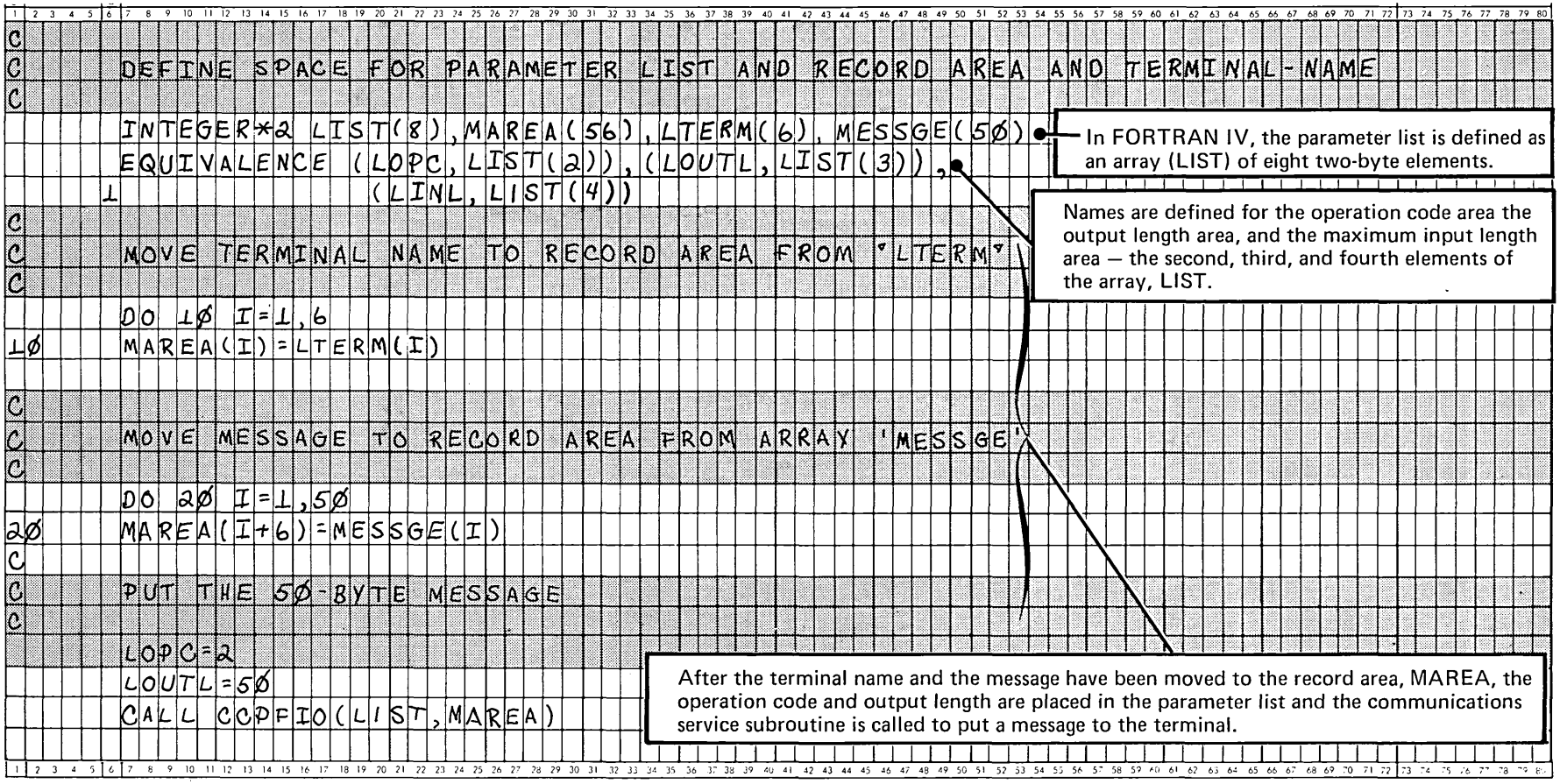
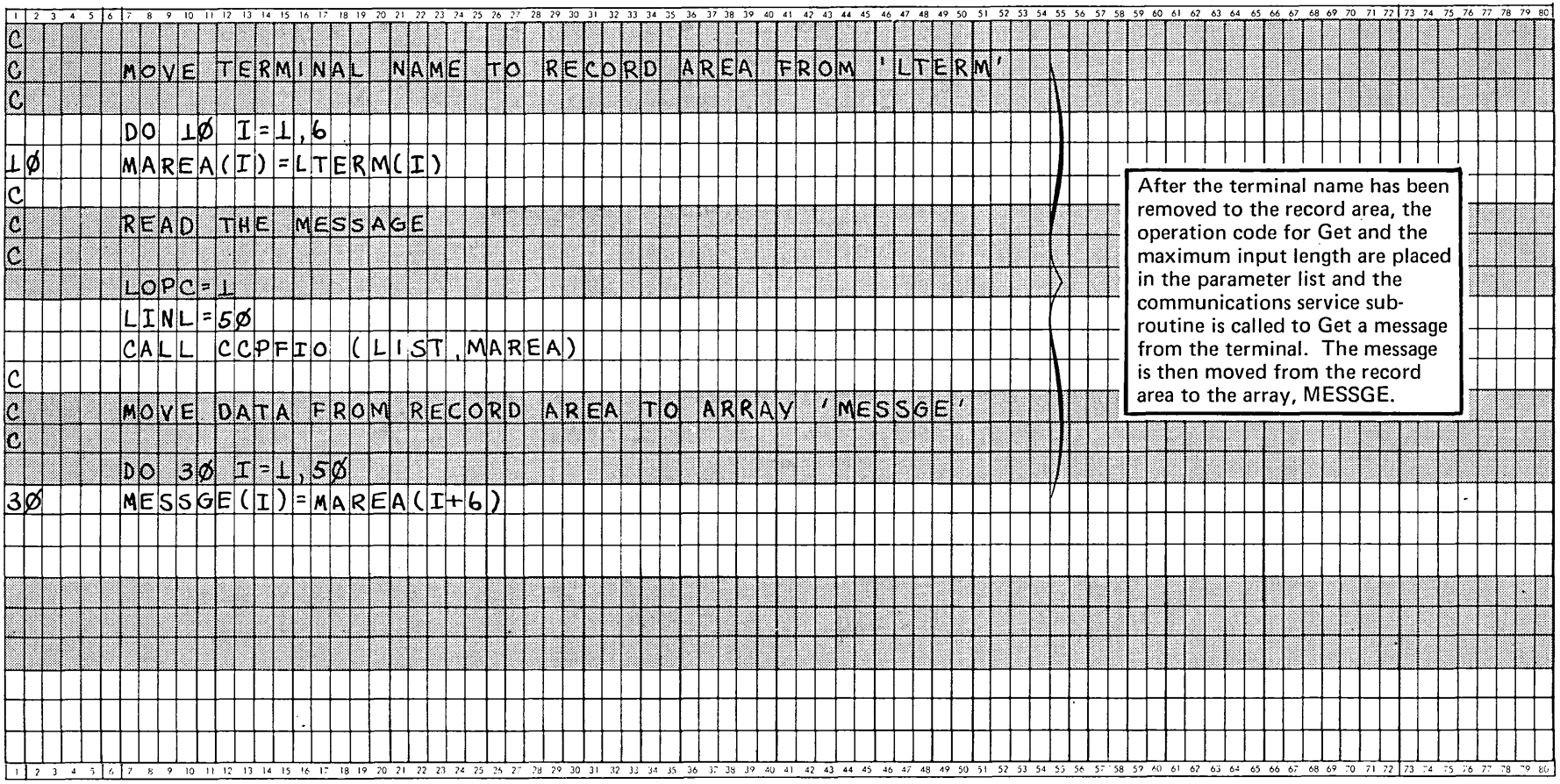


Figure 6 (Part 1 of 2). FORTRAN IV CCP Communications Programming Example



```
C
C     MOVE TERMINAL NAME TO RECORD AREA FROM 'LTERM'
C
      DO 10 I=1,6
10    MAREA(I)=LTERM(I)
C
C     READ THE MESSAGE
C
      LOPC=1
      LINL=50
      CALL CCPFIO (LIST,MAREA)
C
C     MOVE DATA FROM RECORD AREA TO ARRAY 'MESSGE'
C
      DO 30 I=1,50
30    MESSGE(I)=MAREA(I+6)
```

After the terminal name has been removed to the record area, the operation code for Get and the maximum input length are placed in the parameter list and the communications service subroutine is called to Get a message from the terminal. The message is then moved from the record area to the array, MESSGE.

Figure 6 (Part 2 of 2). FORTRAN IV CCP Communications Programming Example

IBM System/3 Basic Assembler Coding Form

Form X21-9107
Printed in U.S.A.

PROGRAM CCP Basic Assembler Coding Example | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE OF

PROGRAMMER | DATE | CARD ELECTRO NUMBER

STATEMENT: Name | Operation | Operand | Remarks | Identification-Sequence

```
* PARAMETER LIST
PARM    EQU     *           HIGH-ORDER ADDRESS OF PARAMETER LIST
        DS      XL2         RETURN CODE
        DS      XL2         OPERATION CODE AND MODIFIERS
        DS      XL2         OUTPUT LENGTH/ACTUAL INPUT LENGTH
        DS      XL2         MAXIMUM INPUT LENGTH
        DS      XL2         ADDRESS OF RECORD AREA
        DS      3XL2        REQUIRED WORK AREA

* RECORD AREA
RECA    EQU     *
        DS      CL6         SYMBOLIC TERMINAL NAME
DATA    DS      CL50        DATA AREA

* WRITE A 50-BYTE MESSAGE FROM 'MESSGE' TO TERMINAL 'TERMAA'
        MVC     DATA(50),MESSGE                 MOVE DATA TO RECORD AREA
        $NCIO   PLIST-PARM,OP-PUT,OUTLEN-50,TNAME-TERMAA   PUT THE MESSAGE

* READ A 50-BYTE MESSAGE INTO 'MESSGE' FROM TERMINAL 'TERMAA'
        $NCIO   PLIST-PARM,OP-GET,INLEN-50,TNAME-TERMAA    GET THE RETURN MESSAGE
        MVC     MESSGE(50),DATA                 MOVE DATA TO MESSGE FIELD
```

A communications service subroutine is not used in Basic Assembler. Instead, a macro ($NCIO) is used to initialize the parameter list and record area and to call CCP communications facilities for each operation.

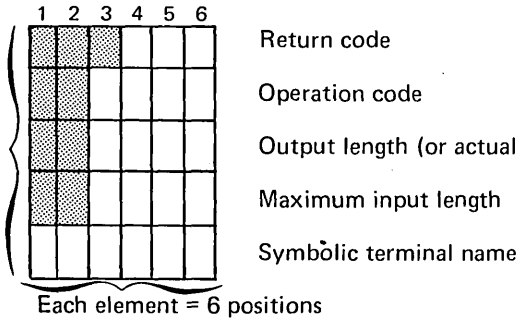
Figure 7. Basic Assembler CCP Communications Programming Example

Figure 8 is an example of the use of SPECIAL files to communicate with a terminal in an RPG II inquiry application. (Examples of EXIT and RLABL will be available in a later publication.) Figure 8, Part 1, shows the format of the RPG II arrays that contain the communications parameter lists and shows how the arrays are loaded during RPG II compilation.

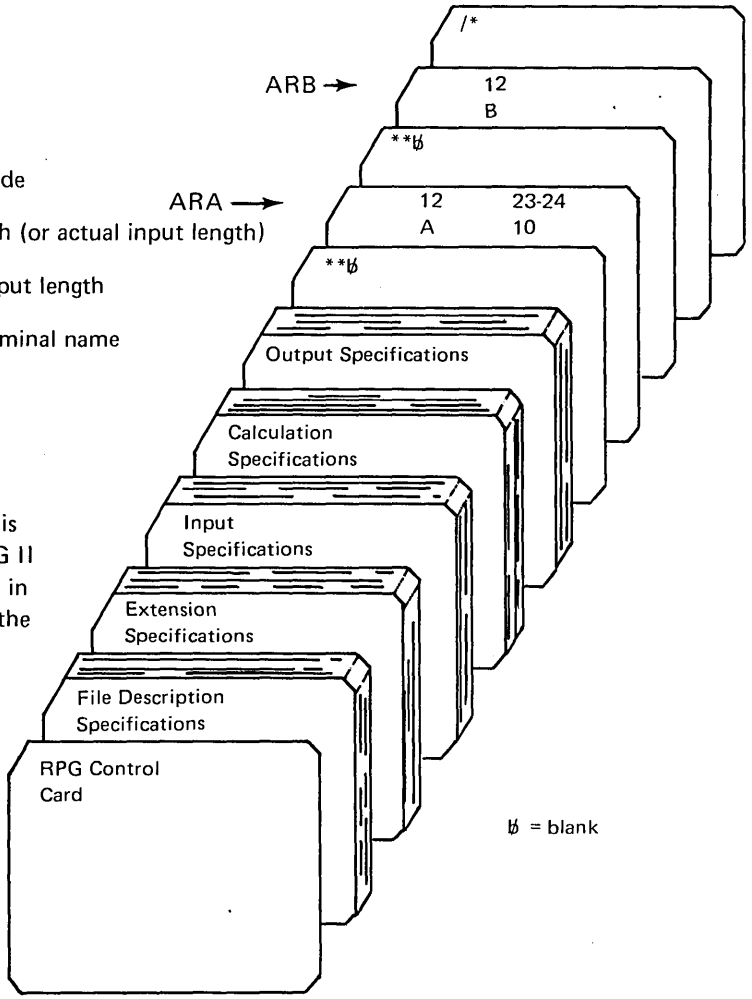
Space must be reserved in the RPG II communications record area for a return code, in addition to the space reserved for the terminal name. Thus, the input or output record must begin in position 11.

Figure 8, parts 2 and 3, show the RPG II coding to Get an inquiry from a terminal, chain to an inventory file using the input stock number (STOCKN) as the chaining field, and return the inventory amount (INVAMT) to the requester terminal. If an inventory item is not found (resulting indicator 05 is turned on), a "not in stock" message is Put to the requester terminal. If an error or exception return code is received on either the Get or the Put operation, a corresponding message is Put to the system operator's console.

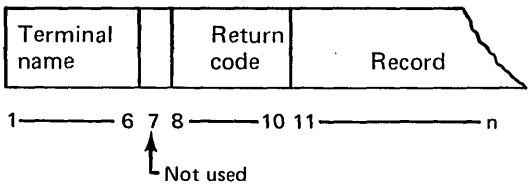
RPG II Parameter List Array



The compilation time arrays (ARA and ARB) for this example are placed as shown on the right in the RPG II source deck. Each array contains an operation code in position 12 (A = Get; B = Put). ARA also contains the maximum input length (10) in positions 23 and 24.



Communications I/O Record Format



In RPG II, 10 positions in each communications record are reserved for the terminal name and the return code.

Figure 8 (Part 1 of 3). RPG II CCP Communications Programming Example

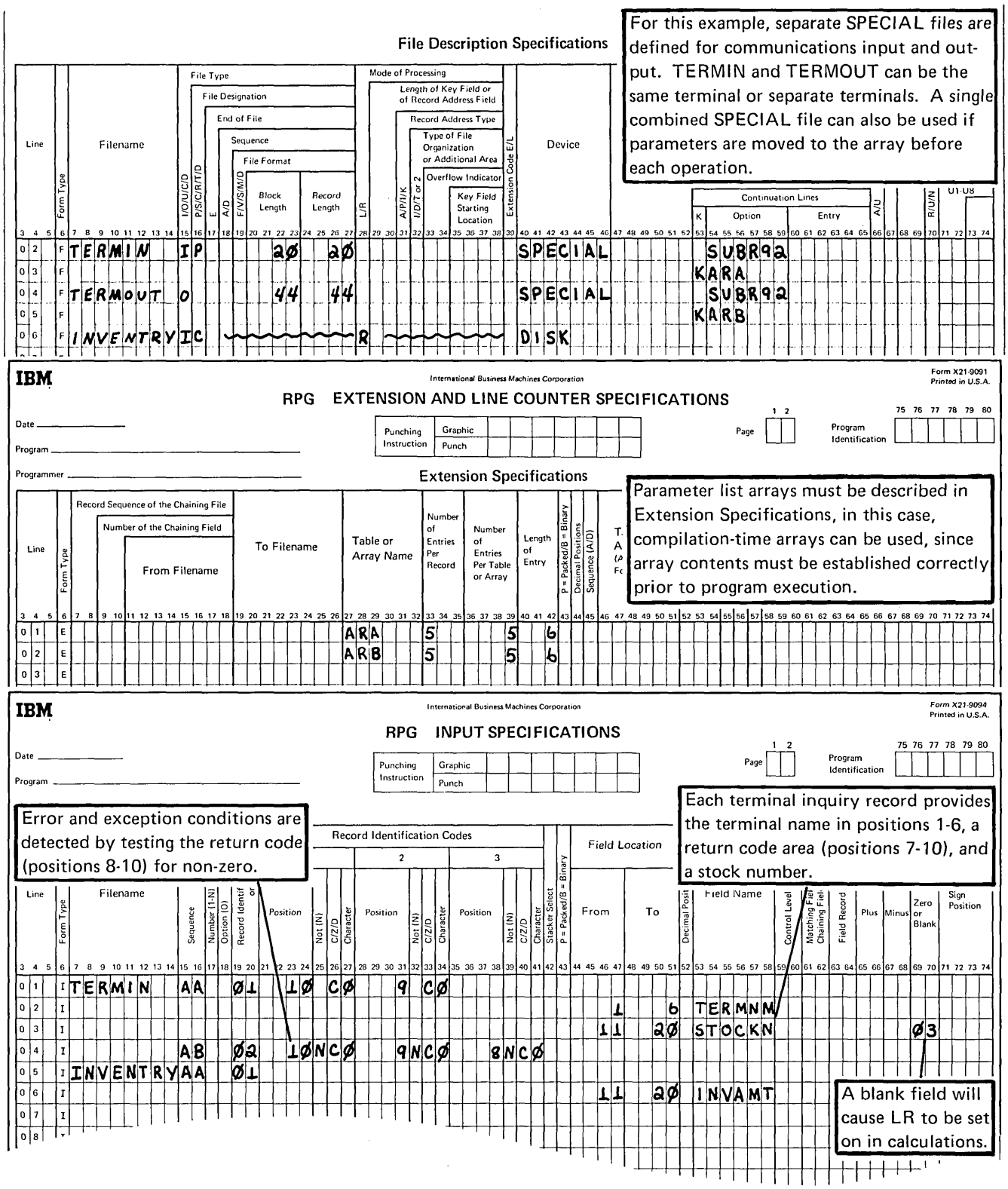
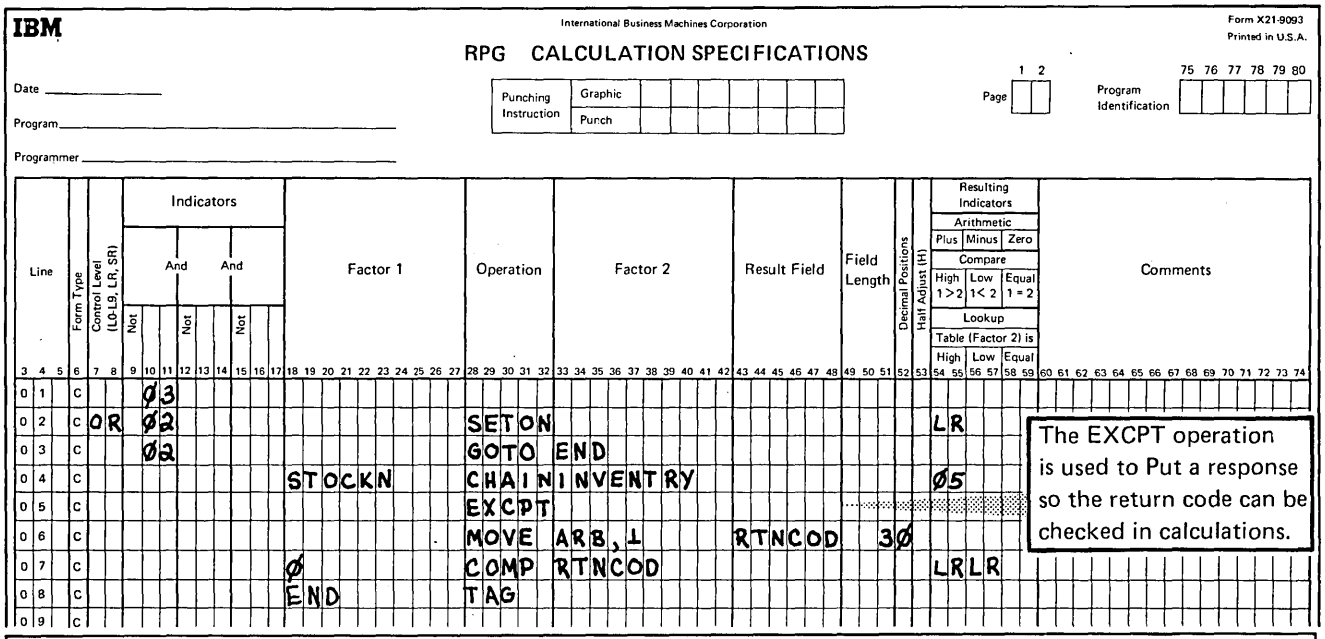


Figure 8 (Part 2 of 3). RPG II CCP Communications Programming Example



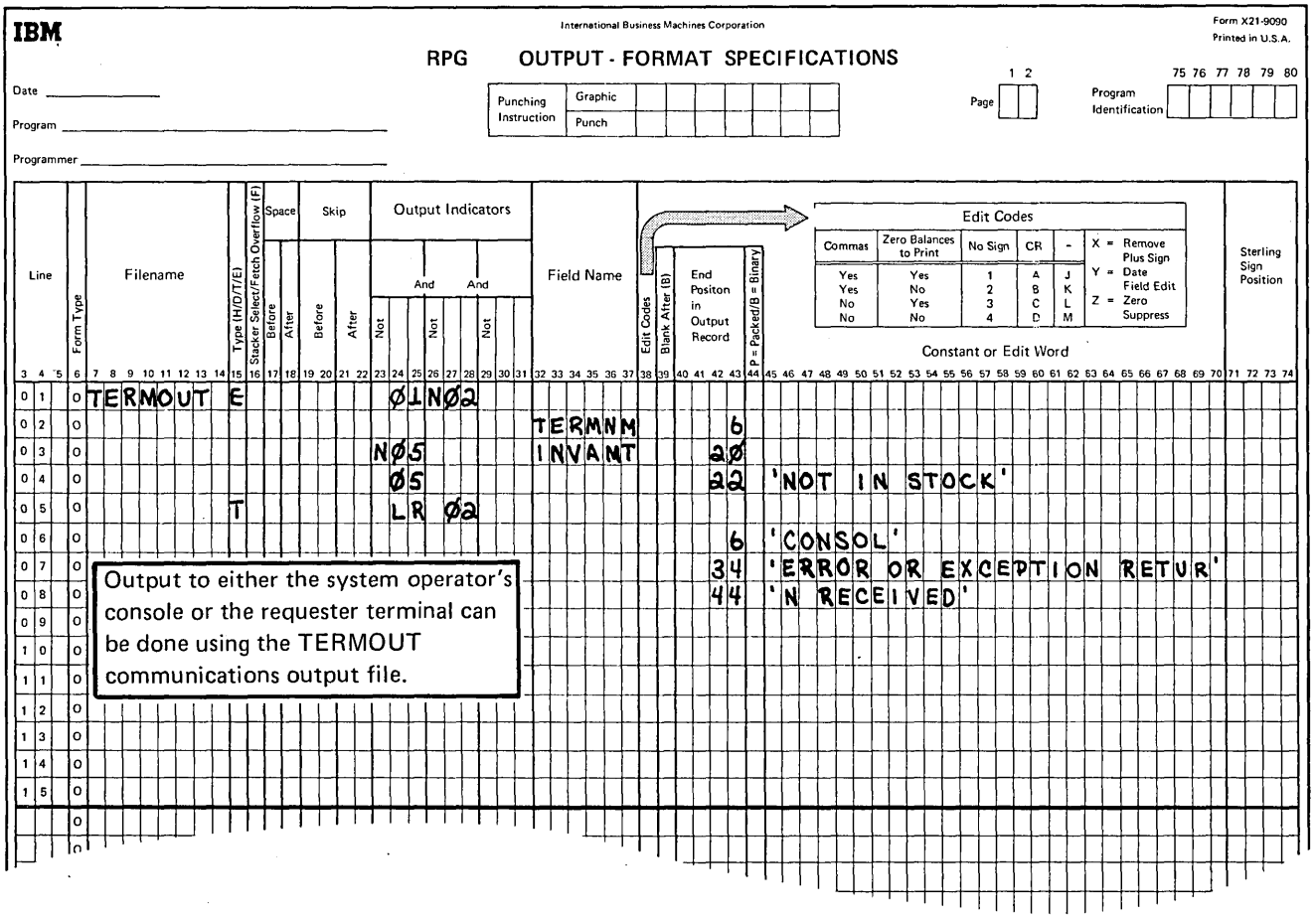


Figure 8 (Part 3 of 3). RPG II CCP Communications Programming Example

DESIGNING COMMUNICATIONS PROGRAMS

The other significant difference (besides the manner of requesting terminal operations) between writing communications programs and writing programs that do not communicate online with terminals is the design of the programs.

Programs that do not communicate with online terminals are most often designed to run in *batch processing* mode; that is, one program completely finishes its processing before the next program begins to run. Often, the program processes a large number of data records which contain similar data in a similar format. Such a program probably uses only a few data files; perhaps it builds a temporary file and updates a permanent file. (Communication terminals can also be used to advantage in batch processing mode.)

Most communications programs, on the other hand, are designed for a very different environment, characterized by *online processing*; that is, data enters the computer directly from the point of origin and is transmitted directly to where it is used. The communications environment often includes several terminals, each making requests in a random manner, each request requiring the execution of a different program. Each program might process only a single transaction at a time for the terminal, affecting several different files. The majority of communications programs utilize this type of processing, which requires a program design different from that required for batch processing.

Terminal Types

Important factors that influence the design of communications programs under the CCP are the capabilities of terminals the program uses and how the program uses terminals.

Capabilities of Terminals

Two types of terminals are identified, based on whether or not the terminal is capable of entering commands to the CCP. Terminals are designated either command terminals or non-command terminals at assignment time (see index entry *TERMINAL statement*).

Command Terminals: In the CCP environment, a command terminal is a terminal that is capable of commanding the CCP to perform special services, the most significant of which is requesting a program. Command terminals must be capable of both input and output, since they must be able to transmit commands to the CCP and receive messages from the CCP. Once a command terminal has requested a program, it is capable of sending and receiving data under direction of the program. The terminal remains under control of the program until the program releases it or terminates. At that time, the terminal is allowed to enter additional commands.

For switched lines, a line (rather than a terminal) is designated command or non-command at assignment time. When a command line is not under control of a program, the CCP awaits calls from command terminals on that line.

Non-Command Terminal: A non-command terminal is not capable of requesting CCP services, but is only capable of transmitting and receiving data under control of the program that uses the terminal. When a program releases the terminal, the terminal is not used until it is selected by another program.

A non-command switched line is under complete control of application programs. Connections are established (answers or calls) when the program performs I/O operations referencing the symbolic name of a terminal on that line.

A non-command terminal might have only input capability, only output capability, or both input and output capability. Terminals that have both input and output capability can be designated non-command terminals at assignment time.

Program Use of Terminals

The CCP discriminates between a user program's terminals based on how the terminals became allocated to the program:

- *Requester Terminal:* From the point of view of the application program, a requester terminal is a terminal that has requested the program. Therefore, a requester terminal is always a command terminal. Once connected to the program, the terminal is directed by the program to transmit data, receive data, or both.

- *Program-Selected Terminal:* A program-selected terminal is a terminal that did not request the program, but rather is selected by the program to transmit input data, receive output data, or to both transmit and receive data. A terminal is selected either by being specified at assignment time as required for the program or by being acquired by the program during its execution. A program-selected terminal can be either a non-command terminal or a command terminal not currently "signed on" to issue commands to the CCP.

A program may communicate concurrently with both a terminal that requested it and with terminals it has selected. A program may be written to accommodate one or more additional requesters while it remains in communication with the first requester; each requester can be released as it finishes its interaction with the program.

Examples of Application Program Logic

The following examples illsutrate the logic required to deal with:

- A single requester terminal
- A single requester terminal and program-selected terminals
- Multiple requester terminals
- Multiple requester terminals and program-selected terminals

Single Requester Terminal

A communications program might be written to deal with only the terminal that requested it each time it is executed (Figure 9). One example is an inquiry program that processes one or more messages from its requester and then terminates, using system resources only briefly. The program may access or update several different files in order to complete its processing. A single requester program might also transmit batched data to a host system, where the host is the requester of the program.

❶ If data is expected with the program request, no Invite Input is required, since the first Get General will return the requester's name and data in the record area.

❷ A Put operation with a blank terminal name causes the name of the requester terminal to be returned in the record area. Invite Input can then be issued to that terminal.

❸ The first Get is a Get General; subsequent Gets can either be Get General or Get Specific operations. If Get General is used, Invite Input must be issued first.

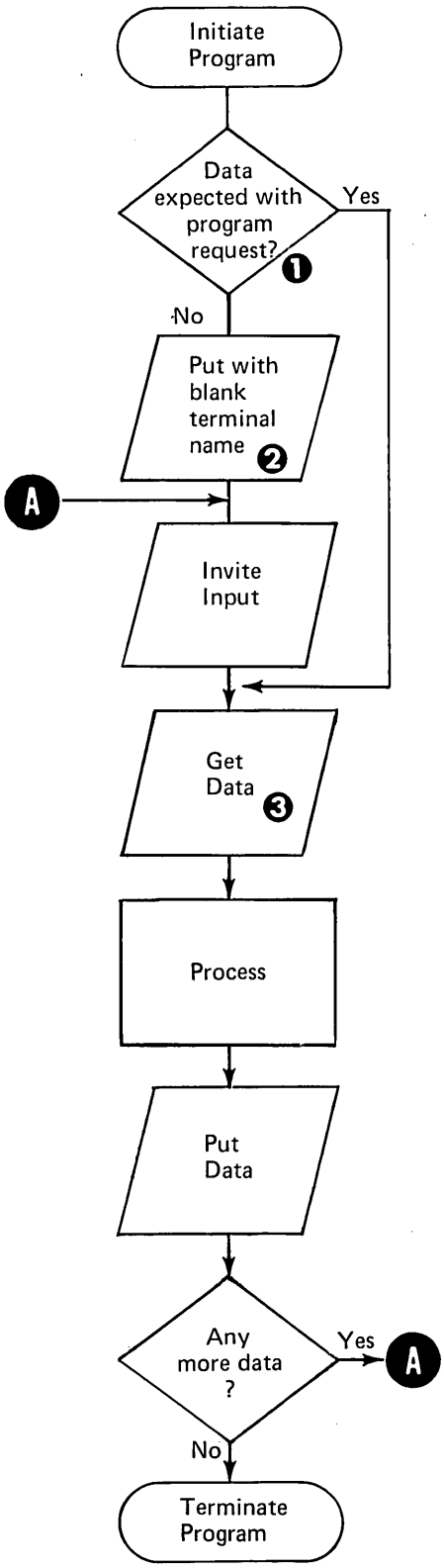


Figure 9. Program that Communicates with a Single Requester Terminal

Single Requester and Program-Selected Terminals

Communications programs can also be written to contact one or more program-selected terminals while processing a request from a single requester terminal (Figure 10). Perhaps the requester wants information from several terminals or wants to send information to several terminals. An example of such a program might be an inquiry program that serves a credit office application. The requester terminal asks for information about a customer from terminals in other offices by issuing a message to program-selected terminals in those offices specifying the customer identification. The attached offices reply with the latest credit information.

Another example of such a program is an order-entry application where the program-selected terminals are allowed to enter orders only when the program has been requested by the system operator.

Such a program can also be used in a batch application. For example, it could transmit batched data from a System/3 to a System/360 or System/370 host system. In this case, the system operator would probably be the requester.

❶ These steps (enclosed by the broken line) are performed only the first time through the program. A program can determine which terminals to select in various ways:

- The terminals required by the program are specified at assignment time and the terminals have been allocated to the program before it gains control.
- The program knows which terminals it needs, but must acquire them itself.
- The program does not know which terminals to select. The program might have to obtain this information from the system operator, a terminal, or from the data he is processing.

When the program knows which terminals to select, it can acquire them (if not already allocated by assignment) and Put a "Hello" message and Invite Input, as required.

❷ When a program is capable of handling more than one terminal, it must set aside a separate work area for each. The program must retain enough information to remember what it has previously received from each terminal. When, for example, input data consists of more than one part, a separate routine often processes each different part. A complete input message might consist of a customer name or number, an order number, item numbers, quantities, prices, and other information, entered as separate lines of input data and, in fact, as separate transmissions from the terminal. The program must be able to determine which portion of the data it is processing, where to store that data in the work area, and which routine processes that portion of the data.

❸ When a program-selected terminal has completed a message sequence, the program must determine whether it can invite additional input from the terminal. For example, if the program has received an end-of-input signal or if the system operator has issued the SHUTDOWN command, the program should not issue an Invite Input to the terminal.

❹ If any other program-selected terminals have input messages to transmit (have outstanding Invite Input operations issued to them), the program finishes processing them. When all input from the program-selected terminals and the requester has been processed, the program terminates.

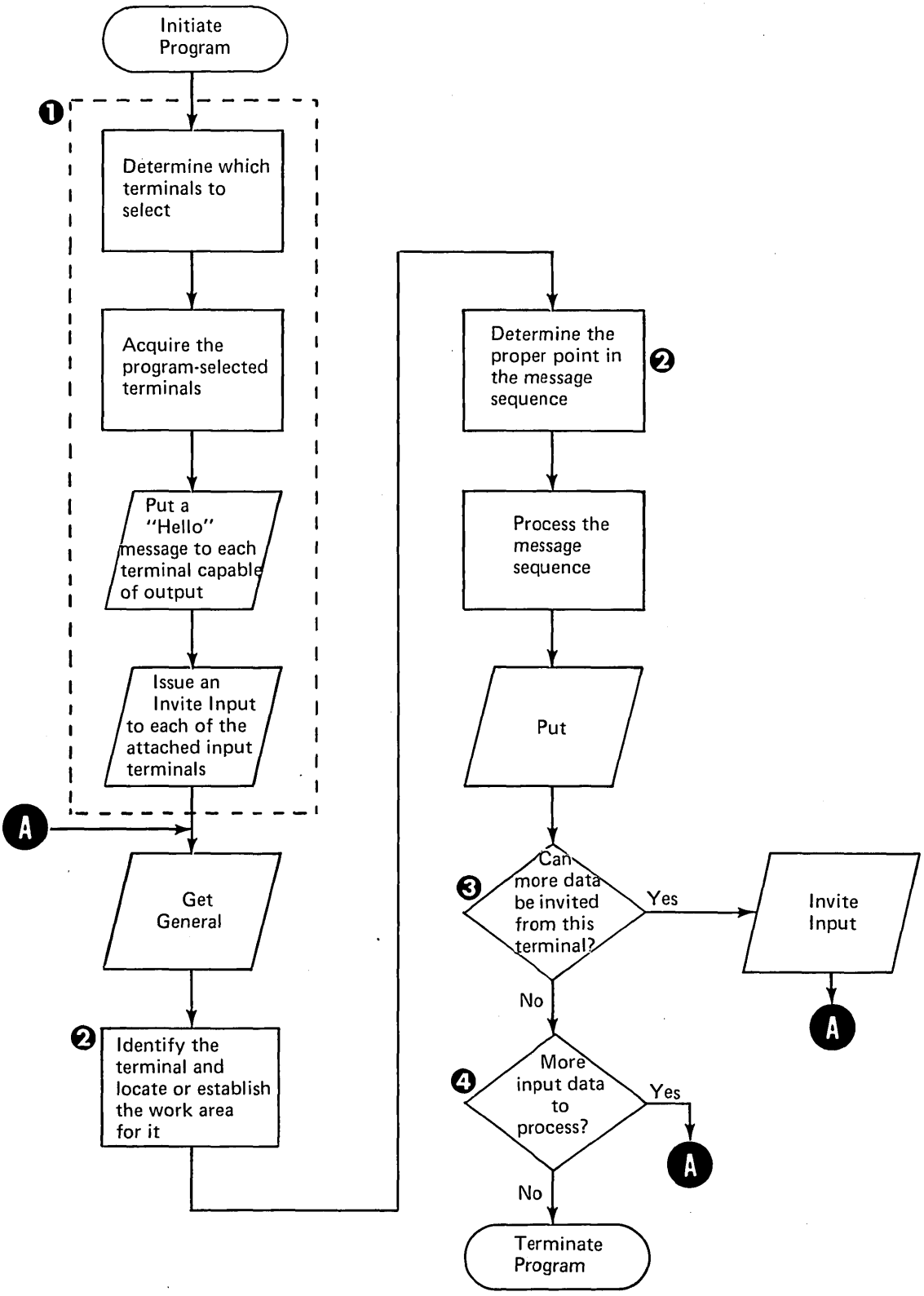


Figure 10. Program that Communicates with a Single Requester Terminal and Program-Selected Terminals

Multiple Requester Terminals

A communications program can be written to process requests from more than one requester terminal each time it is executed. In the example shown in Figure 11, no terminals are program-selected by the program.

Both order entry and inquiry applications might require a program that supports multiple requester terminals, especially if the program is used frequently or remains in main storage for such a length of time that it is likely to be requested by additional terminals while it is executing.

An order entry program might support multiple requesters if it exchanges several messages with each requester. For example, perhaps the program prepares shipping invoices using the order data entered from terminals at remote locations.

When an inquiry program is requested frequently, it might be more efficient if several requesters can use it at the same time without separate copies of the program being in main storage at the same time. Also, if the program can handle multiple requesters, it may not have to be reloaded as frequently. These factors are especially critical in minimum systems (CCP systems that execute only one program at a time and have limited main storage capacity).

❶ The Invite Input is bypassed the first time through since it is not known which terminal requested the program until after the Get General operation.

❷ When the program has received the final portion of a message sequence from a particular terminal, it must determine whether a new message can be accepted from the terminal. If, for example, the terminal has indicated that this is the last message it will send or if the system operator has issued the SHUTDOWN command to shutdown the CCP, the program should not issue an Invite Input to the terminal.

❸ If no Invite Input was issued to this terminal, the terminal is released from the program.

❹ If requests from other terminals are in process, they must be completed before the program terminates.

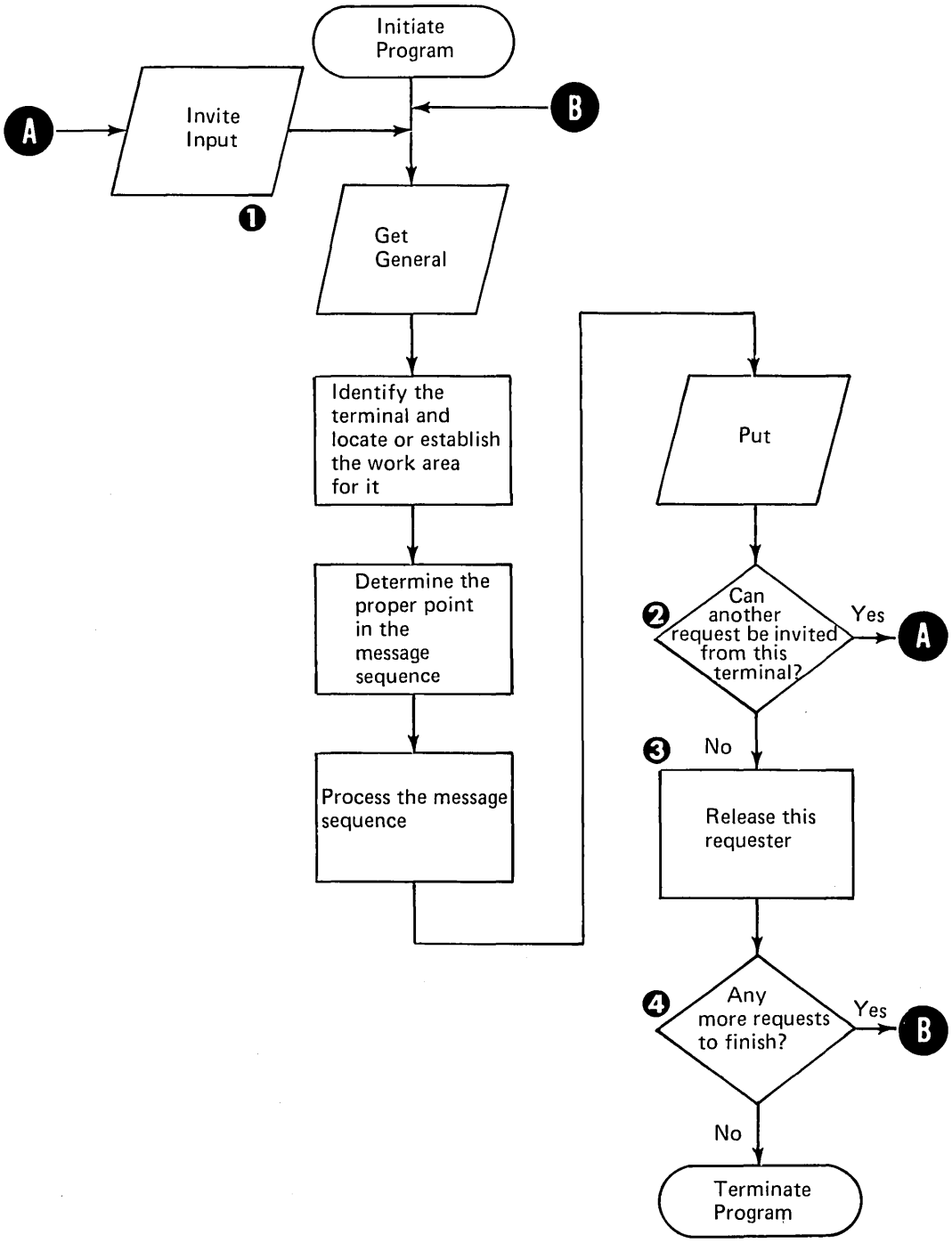


Figure 11. Program that Communicates with Multiple Requester Terminals and No Program-Selected Terminals

Multiple Requester and Program-Selected Terminals

Communication programs can also be written to accept concurrent requests from several requester terminals and, in satisfying the requests, to contact one or more program-selected terminals (Figure 12). As each requester terminal is satisfied, the program releases it, enabling it to make other requests. The program-selected terminals are probably not released from the program until the last requester terminal has been served and the program is terminated.

This kind of a program might be used, for example, in an inquiry application in a credit office, similar to the example described previously for programs that handle a single requester and program-selected terminals. Serving several requesters concurrently provides increased efficiency when a large number of credit inquiries are made, since the program need not be reloaded for each individual request.

In an order entry application, this kind of a program might accept orders from multiple requester terminals and transmit the output of the program to program-selected terminals.

❶ The first-time processing required when program-selected terminals are used is described in Figure 10.

❷ When a program is capable of handling more than one terminal, it must set aside a separate work area for each. The program must retain enough information to remember what it has previously received from each terminal. When, for example, input data consists of more than one part, a separate routine often processes each different part. A complete input message might consist of a customer name or number, an order number, item numbers, quantities, prices, and other information, entered as separate lines of input data and, in fact, as separate transmissions from the terminal. The program must be able to determine which portion of the data it is processing, where to store that data in the work area, and which routine processes that portion of the data.

❸ When a program-selected terminal has completed a message sequence, the program must determine whether it can invite additional input from the terminal. For example, if the program has received an end-of-input signal or if the system operator has issued the SHUTDOWN command, the program should not issue an Invite Input to the terminal.

❹ When the program has received the final portion of a message sequence from a particular terminal, it must determine whether a new message can be accepted from the terminal. If, for example, the terminal has indicated that this is the last message it will send or if the system operator has issued the SHUTDOWN command to shutdown the CCP, the program should not issue an Invite Input to the terminal.

Other Program Attributes

Reusable Programs

CCP efficiency can be increased by using programs that are written to be *serially reusable.* A serially reusable program is written to ensure data integrity even if it is re-executed without being reloaded. The reusable attribute could be especially important in programs that are requested frequently. Programs written in Basic Assembler or COBOL to handle the previous logic examples can be written to be reusable.

Never-Ending Programs

If a user program is requested very frequently throughout the CCP run and sufficient user program area is available, it can be written as a never-ending program. Once a never-ending program is loaded, the main storage it occupies is permanently unavailable for other programs, even if it terminates. Therefore, a never-ending program with no work will wait until it is requested again.

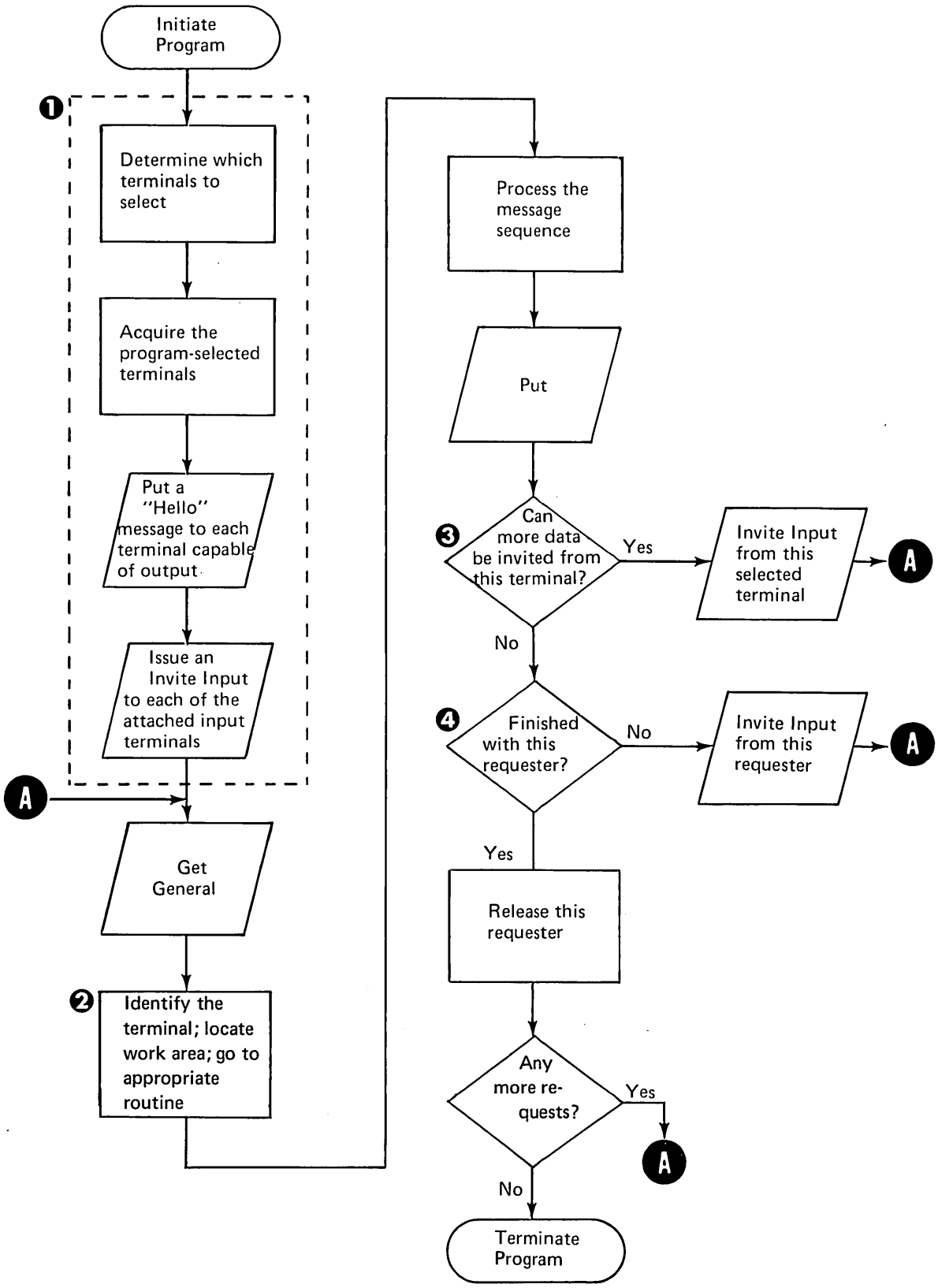


Figure 12. Program that Communicates with Multiple Requester Terminals and Program-Selected Terminals

TESTING COMMUNICATIONS PROGRAMS

Communications programs written to run under the CCP could be tested using the following testing procedure:

1. Compile or assemble the program.

2. Store the program to be tested in an object library on the program pack that contains the CCP or on the system pack.

3. Ensure that the CCP assignment file includes the program name in the assignment set under which the CCP will run, describing the resources required by the program (including any test files required by the program).

4. Perform the CCP startup procedures with FILE statements among the OCL statements for any test files that were described to the CCP at assignment time (see index entry *startup*).

5. Enter a request for the program from a terminal or from the system operator's console (see index entry *program requests*).

6. Enter test input data as required by the program.

7. If test files are to be printed using the Disk Copy/Dump utility (see *IBM System/3 Model 10 Disk System Control Programming Reference Manual* , GC21-7512), the CCP must be shut down prior to running the utility, since the utility can neither be executed under the CCP nor in the opposite level from the CCP in a DPF system, as all test output data may not be available to it.

8. Evaluate the tested program by inspecting test files and other output from the program.

Caution: Care should be taken during testing that a malfunction in the test program does not destroy other programs or data in main storage.

CCP *generation* is the process whereby the user selects the portions of the distributed CCP which will give him the capabilities he wants in his version of the CCP. Generation is the first stage in creating the CCP, when the user establishes its maximum physical size (*main storage space required*) and its maximum capability. Further tailoring is done at assignment time (see *Making an Assignment Run*) and at operational startup (see *Running the CCP*).

Performing CCP generation is similar to performing system generation for the Model 10 Disk System. The user describes his system configuration and the functions he wants by means of a series of statements which consist of keywords with associated values. Some of these statements describe the system configuration (main storage size; input/output units; terminal and line capability) within which the CCP will operate. Other generation statements give the user the ability to control the maximum capabilities the CCP is to have (in terms of types of programs, number of concurrently executing programs, and file sharing) and whether the CCP is to have certain optional features such as password sign-on and user program request counts.

Generation creates a control file, which is completed at assignment time as the assignment file ($CCPFILE). This file must be on the same disk pack on which the CCP is generated.

PROCEDURE FOR GENERATION

Figure 13 outlines the procedure for generating the CCP. The procedure assumes that disk system management has been generated and disk system IPL has been performed previously. The key step for the user in the procedure is step 3, where he describes the kind of CCP he wants by modifying the sample generation statements. In order to modify the statements, the user must know certain facts about his system and about the version of the CCP he wants to generate. These facts are determined during the CCP system design. Design considerations are given in *Chapter 2: Designing Your Communication-Based System*, earlier in this publication. The following is a checklist of the facts a user must know before performing generation:

- Type of card device attached to the system (MFCU and/or 1442)
- Disk configuration of the system (5444 and 5445 disks)
- Maximum number of concurrently-executing user programs
- Maximum length of the user's commands to the CCP
- Whether the system has DPF or not
- Is disk file sharing allowed?
- Is a program usage count to be kept?
- The six user-specified data mode escape characters
- What type of sign-on security will be used, if any?
- Maximum number of assignment sets planned for $CCPFILE
- Maximum number of programs and files in an assignment set
- Maximum number of terminals planned
- Main storage size of processing unit
- MLTA lines and line features supported
- MLTA terminal devices to be supported
- MLTA terminal features to be supported
- BSCA adapters, lines, and line features to be supported
- BSCA terminal devices to be supported
- MLTA and/or BSCA translation tables required
- Disk unit and pack name on which disk system management resides (F1 or R1)
- Disk unit and pack name on which the distributed (ungenerated) CCP modules reside
- Disk unit and pack name on which the CCP will be generated
- Disk unit(s) and pack name(s) where work file space can be found during generation

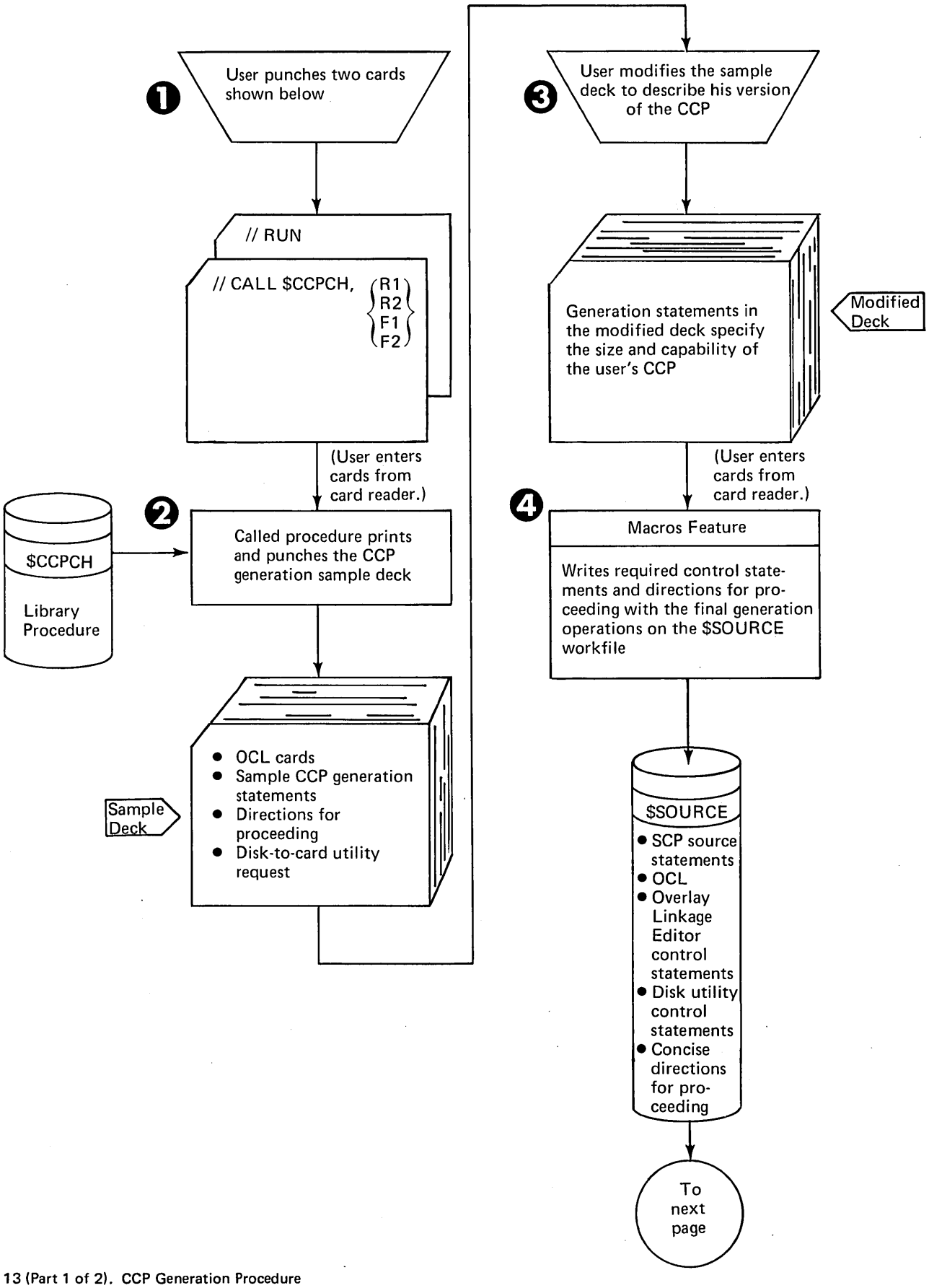


Figure 13 (Part 1 of 2). CCP Generation Procedure

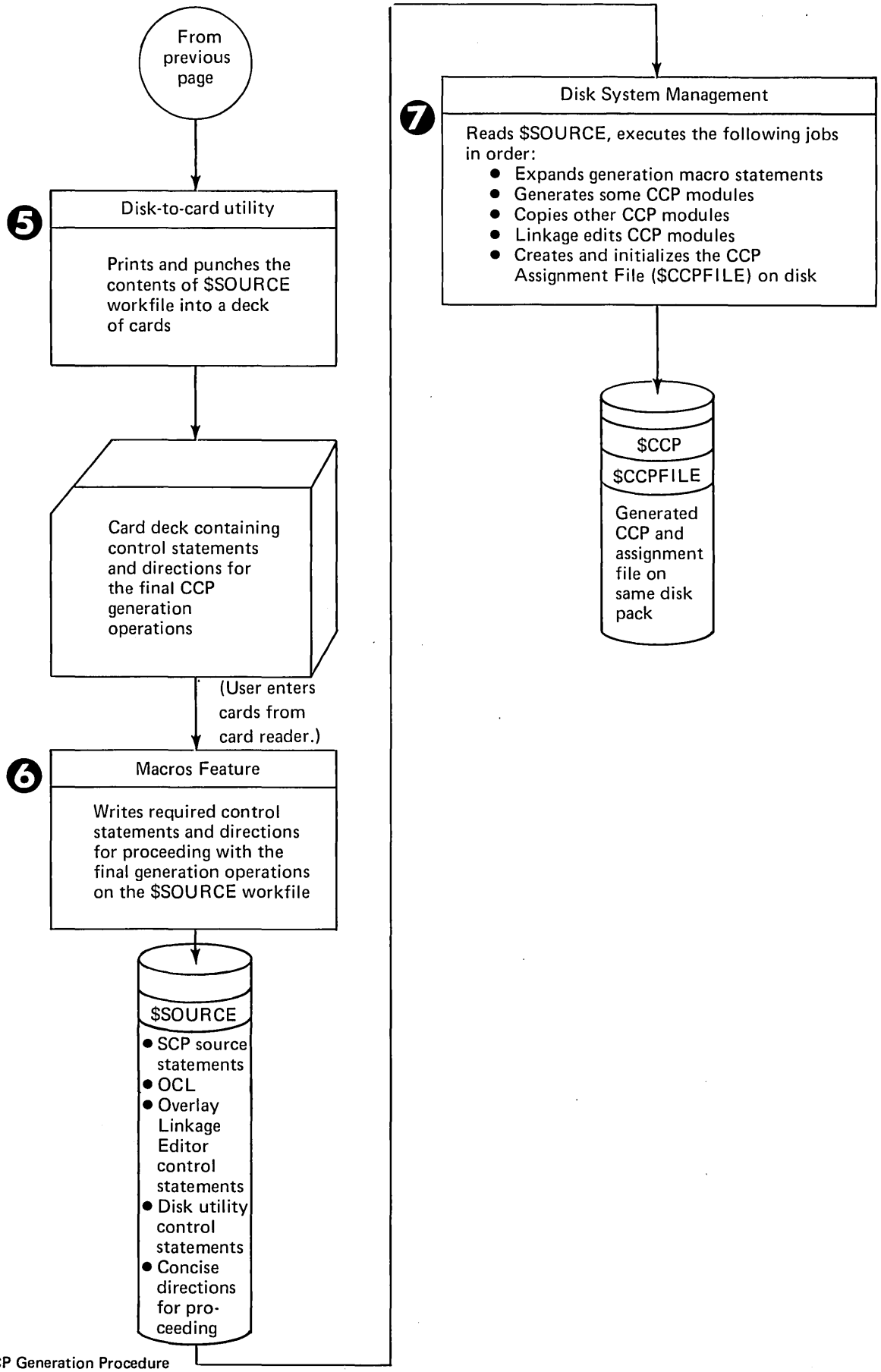


Figure 13 (Part 2 of 2). CCP Generation Procedure

DIAGNOSTIC MESSAGES

The CCP generation stage ($$CCPGEN) analyzes the generation statements and prints diagnostic messages to inform the user of errors he has made in describing his system or his version of the CCP. If errors are encountered in the statements, neither the CCP nor the assignment file ($CCPFILE) are generated. In order to continue with generation, the user must correct the statements that are in error and reenter the deck.

SAMPLE PROGRAM

A sample program is provided with the distributed CCP. This program should be executed to verify a satisfactory generation before the generation procedure is considered complete. A complete description of the sample program and the procedure for running it will be provided in a later publication.

PLANNING FOR GENERATION

To accomplish CCP generation smoothly, planning must begin during the system design process so that the physical configuration of the communication-based system is closely mated to, yet adequate for, its planned applications. This requires careful system design and careful evaluation and comparison of the various data transmission codes, line speed, line types, terminal types, and communication adapters. In this way, all of the facts about the system and the CCP that are required for generation can be available before the actual generation run.

The advice and aid of an IBM Systems Engineer will be helpful in planning for CCP generation. Once the facts needed for the generation statements have been determined, the generation process can easily be performed by the system operator.

Requirements For Generation

In order to generate the CCP on a given System/3 Model 10 Disk System, the system must meet minimum requirements of machine configuration, programs, and files. These requirements are not as great as the requirements for operating the CCP.

Machine Requirements

In order to allow the generation process to be performed, a system must meet the following minimum machine requirements:

- 16K of main storage available for the CCP generation programs to execute (5410 Model A15 Processing Unit, or larger)
- One 5444 Model 2 Disk Storage Drive
- 5424 Multi-Function Card Unit or 1442 Card Read/Punch
- 5203 or 1403 printer

Program Requirements

Generation of the CCP requires IBM System/3 Model 10 Disk System Management (5702-SC1), including:

- Macros Feature (feature codes 6020, 6021)
- Overlay Linkage Editor Feature (feature codes 6026, 6027)
- Programming support for the desired communication adapters: MLMP Feature (feature codes 6030, 6031) for the BSCA; Multiple Line Terminal Adapter Feature (PSHRPQ number 5799-WAU).

File Requirements

Sufficient disk file space is required for the following files during the generation process:

- Work files
- $CCPFILE, the control file built for the CCP during generation and completed during assignment

Specific file size estimates will be provided in a later publication.

During the CCP generation stage, the user fixes the maximum size and capabilities of the CCP. He describes the CCP operating environment in more detail during the *assignment stage* of the CCP.

During assignment, the user defines one or more "sets" of terminals, files, programs, and system environments that are available to the CCP. These assignment sets are recorded in the assignment file ($CCPFILE) allocated during the generation stage (see *Chapter 4: Generating the CCP*). The CCP will run under one of the assignment sets; that is, CCP has access to a particular group of terminals, files, and programs. The user can vary these resources by specifying a different set of assignments during operational startup (see index entry *operational startup*). This allows him, for example, to control which programs are eligible to be called during a particular CCP run or to restrict the use of certain files during a run.

The information placed in an assignment set during the assignment stage is valid for any number of CCP runs, until a terminal, program, or file must be added to the set or removed or until aspects of the system environment change. When this occurs, the contents of the assignment file can be modified by repeating the assignment run, without regenerating the CCP.

ASSIGNMENT STATEMENTS (ASSIGNMENT BUILD PROGRAM)

In order to place an assignment set in the assignment file, $CCPFILE, the user must run the Assignment Build Program, $CCPAS. The user's input to the assignment program consists of a series of statements, similar to OCL statements, identifying the programs, files, terminals, and certain system options that constitute a CCP operating environment. The assignment statements are printed by $CCPAS on the system logging device.

The assignment program analyzes the statements to ensure that they are valid and writes information from them into the assignment file as an assignment set. The assignment program can be used to create new assignment sets, delete existing assignment sets, or replace existing sets with new specifications. Different assignment sets can be created, deleted, and replaced in the same run of the assignment program.

A set of assignment statements begins with a statement identifying the assignment set and whether it is to be created, deleted, or replaced. Only this statement is required to delete a set. Subsequent statements provide the specific information to be placed in the assignment set. When an item in an assignment set is replaced, a complete set of assignment statements must be entered.

The parameters accompanying the assignment statements will be described in a later publication.

SET Statement

The SET statement identifies the assignment set to be operated upon and which operation is to be done: replace, create, or delete. This is the only statement required to delete an assignment set.

SYSTEM Statement

The SYSTEM statement defines several facts about the system environment in which this assignment set will be used:

- Maximum number of concurrently-executing user programs allowed
- Minimum total main storage space required for the user programs which must execute under this assignment set
- Whether the printer, MFCU, and 1442 will be dedicated to the CCP program level or not (DPF system)
- Password for this set, if the user chose the option at CCP generation
- The number of bytes to be set aside as the CCP buffer area for communicating with terminals
- Amount of main storage set aside for the other program level in a DPF system (if a PARTITION statement will not be entered prior to startup)

BSCALINE Statement

The BSCALINE statement defines a BSCA communication line to be used and the features of the line. A BSCALINE statement must be followed by at least one TERMINAL statement defining a terminal attached to the line. Up to two BSCALINE statements may be included; if no BSCALINE statements are included, at least one MLTALINE statement must be included.

Each BSCALINE statement contains a complete description of the line:

- Line type (see index entry *line type*)
- Transmission code
- Order in which terminals are to be polled (terminals can be mentioned more than one in the polling list)
- Other features of the line

MLTALINE Statement

The MLTALINE statement defines a MLTA communication line to be used and the features of the line. Like the BSCALINE statement, MLTALINE statement must be followed by one or more TERMINAL statements. Up to eight MLTALINE statements can be used; if no MLTALINE statements are included, at least one BSCALINE statement must be present among the assignment statements. MLTALINE statements must follow BSCALINE statements.

Each MLTALINE statement contains a complete description of the line:

- Line type
- Transmission code
- Polling order of terminals
- Other features of the line, such as line speed

TERMINAL Statement

TERMINAL statements define the attributes of the terminals on a communication line. The TERMINAL statements following a BSCALINE or MLTALINE statement provide information about the terminals on that BSCA or MLTA communication line:

- CCP terminal reference identification (used in assignment and during CCP operation as the identification of actual terminals when CCP is communicating with the system operator)
- Terminal type (2740, 1050, . . .)
- Action (HOLD or DROP) to be taken when a terminal operator of a command terminal issues an OFF command (see index entry *OFF command*)
- Whether the terminal is online
- Whether or not the terminal is a command terminal
- Other characteristics of the terminal, such as record checking

TERMNAME Statement

The TERMNAME statement associates a symbolic terminal name with a specific terminal. One or more names must be assigned to every terminal during assignment; that is, for every TERMINAL statement, there must be at least one TERMNAME statement assigning a name to that terminal. If more than one name is assigned to a terminal, the first is known as the *primary name*. When a terminal is signed on or active, it is known to the CCP by its primary name, if no /NAME command is entered (see index entry *Terminal Name Command*). A terminal can be known to the program using it by only one symbolic name.

The terminal reference identification is optional in a TERMNAME statement. If a reference identification is not given, any program using the symbolic name given on the TERMNAME statement cannot execute until the system operator has associated the name with a specific terminal during execution of the CCP (see index entry *Assign a Symbolic Terminal Name to a Terminal*).

DISKFILE Statement

The DISKFILE statement describes a disk data file that can be used by programs running under this assignment set. A DISKFILE statement must be given for every file named in every program defined in the PROGRAM statements for this set. Both physical files and symbolic files must be defined by DISKFILE statements (see index entry FILE command for a description of physical and symbolic files).

The DISKFILE statement gives the following information about the file:

- File name
- Device (5444 or 5445 disk drive)
- Organization
- Record length
- Key length and key location (for indexed files)
- Physical file names that can be referred to by this file name, if the file name defined in this statement is for a symbolic file
- Other information about the file

PROGRAM Statement

The PROGRAM statement describes a program which can be requested by a terminal during execution of the CCP under this assignment set. The following information about the program is given:

- Name
- Whether the program is capable of handling multiple requesting terminals
- Whether the program is serially reusable
- Whether the program is a never-ending program
- Whether the program requires a dedicated program level (that is, whether it must be the only program running under a version of the CCP in which more than one program can execute concurrently)
- Whether the program uses the printer, MFCU, and 1442
- Names of terminals which must be available to the program (required program-selected terminals)
- Disk files used by the program and disk access method of each
- Whether input data may be entered at the same time as the program request command
- For multiple requester programs, the maximum number of requesters that can be serviced at one time

ASSIGNMENT LIST PROGRAM

The Assignment List Program ($CCPAL) can be executed anytime after the assignment stage, $CCPAS, has been executed. It has a twofold purpose:

1. List either the contents of all the assignment sets in the assignment file, $CCPFILE, or the contents of any individual set in the file, to show the contents of the set or sets.
2. List the usage count for each program in the assignment file, if the program usage count option was chosen at generation time. The usage counts can be listed either separately for each set or as a total for all sets. The user can clear the usage count to zero either after listing the counts or without listing the counts.

LIST Statements

The Assignment List program is controlled by the LIST statement, which the user enters from the system input device. The LIST statement specifies:

- What is to be listed – an individual set, all sets, only the control file directory, or only the CCP configuration
- Whether or not the program use count should be printed for one or all of the sets in the file
- Whether the program use count should be reset to zero for one or all of the sets in the file

ASSIGNMENT DIAGNOSTICS

The CCP assignment stage, $CCPAS, analyzes each of the assignment statements for invalid specifications. If an error is found in any statement of a set, that set of statements is not processed by the assignment stage; however, the remaining statements in that set are also analyzed for errors.

The Assignment List program, $CCPAL, analyses the LIST statement for errors. Error messages are written to the system logging device.

PLANNING FOR ASSIGNMENT

The assignment run must be repeated each time a new program or file is added to a set, each time the group of terminals available to the CCP changes, or when certain other aspects of the CCP configuration such as the password or changes. The assignment stage of the CCP will be run frequently or infrequently, depending upon how often the CCP environment changes.

If the CCP will run under various sets of assignments in a given period of time, more than one set of assignments can be placed in the assignment file. For example, a certain set of terminals, files, and programs may be available during the day, with a restricted set available during night-time hours. Or, perhaps, a certain group of files, terminals, and programs are required for the week-end, month-end, or year-end operations.

Information from the assignment file, $CCPFILE, is required by various people. The system manager and the system operator must be aware of all current system assignments so they can properly control and maintain the communication-based system. The terminal operator should be aware of some aspects of the system assignments, including which symbolic names are assigned to his terminal and what the current password or other sign-on security information is. Programmers also require current information about the system assignments, such as the symbolic file names and the physical files they reference. In order to make this information available to those who need it, provisions must be made to distribute all or part of the information from the assignment file whenever it changes.

Requirements For Assignment

In order to execute the assignment stage of the CCP, the system must meet minimum requirements. These requirements are less than are needed for the CCP operational stage. The requirements are:

- 16K of main storage available for the CCP assignment stage to execute (5410 Model A15 Processing Unit, or larger)
- One 5444 Model 2 Disk Storage Drive
- 5424 Multi-Function Card Unit or 1442 Card Read/Punch
- 5203 or 1403 Printer

The Assignment Build Program, $CCPAS, and the Assignment List program, $CCPAL, operate under control of the System/3 Model 10 Disk System management. The assignment statements are entered in card form. The user must provide two FILE OCL statements when loading the Assignment Build program, one for the assignment file, $CCPFILE, and one for a work file.

After the CCP has been tailored to the user's environment and needs by the generation and assignment stages, it can be put into operation. The CCP operational stage occurs in three parts: startup, operation, and shutdown.

If the CCP is being started in operation for the first time, all user application programs available to be requested must be placed in an object library prior to CCP startup. If a new program is being added, it must be placed in the object library and the CCP Assignment Build program must be run to update the assignment file ($CCPFILE) prior to CCP startup.

STARTUP

In order to initiate the CCP startup routine, the operator enters the following OCL cards from the reader:

```
// LOAD $CCP, (unit)

// FILE (One FILE statement for each physical user file
.       to be accessed furing the current CCP run)
.
.

// RUN
```

After the CCP startup routine is loaded into main storage, it asks questions of the system operator that allow him to exercise several options. See index entry *Initiating the CCP* for a description of the questions.

Startup performs the following initialization functions:

- Loads the supervisor portion of the CCP
- Loads and updates various tables from the current assignment set in $CCPFILE
- Builds control tables in main storage for files and terminals
- Verifies that adequate main storage is available
- Allocates and opens the files that will be used during the current run
- Locates all user programs that may be requested during the current run
- Opens communications adapters and lines

Startup issues diagnostic messages if the system requirements for startup have not been met, if the user has entered invalid instructions, or if it cannot complete its initialization for some other reason.

The opposite program level of a DPF system is not allowed to run during CCP startup.

OPERATION

During its operation, the CCP manages the environment in which telecommunications application programs run and provides services upon which they can call. The management functions of the CCP are of four types: task management, communications management, file management, and program management. As shown in the following descriptions, these management functions often overlap.

Task Management

The CCP can manage the execution of several independent programs in main storage concurrently. Each program performs a unit of work called a *task.* The programs may be CCP system programs or user programs; their respective tasks are classified as either *system tasks* or *user tasks.* An example of a system task is CCP communications management, which processes all requests for terminal I/O (see index entry *communications management*); an example of a user task is a sales order entry application program, loaded into main storage by request from a terminal operator or the system operator.

The number of user tasks that can be in main storage concurrently depends upon the size of the system. It is the responsibility of CCP task management to route control among the tasks and to schedule work requests for each task.

Processing Unit Control: CCP task management assigns control of the processing unit according to the following priority:

1. I/O completions from terminal devices or the system operator's console are serviced before any other task.
2. The highest priority system task with work to perform is given control. All system tasks are assigned a priority.
3. The next user task with work to perform is given control.

If no tasks are ready to execute, the CCP waits for an I/O completion. In a DPF system, control is given to the other program level.

Request Scheduling and Routing: CCP task management receives, schedules, and routes all requests from the current tasks for:

- Terminal I/O
- Disk I/O
- Unit record I/O
- Nonsharable disk data records
- User data files
- Main storage for programs

Communications Management

CCP communications management includes all services related to requests from user programs and CCP programs for terminal I/O. The CCP does not actually perform the physical I/O, but performs services for the requester which simplify the use of the MLTA IOCS and MLMP BSCA IOCS routines which perform the physical I/O. Among the communications management services performed by the CCP are:

- Terminal monitoring and selection
- Buffer management
- Symbolic terminal naming
- Data code translation
- Terminal testing
- Other services which allow the application programmer to be nearly independent of differences between individual terminals.

Terminal Monitoring and Selection: Terminals that are designated command terminals during CCP assignment are monitored for commands by CCP communications management. For multipoint lines, the user must specify a "polling list" at assignment time (see index entries *BSCA LINE statement* and *MLTA LINE statement*). This list gives the order in which the terminals on a line are to be "polled" (interrogated) for commands when none of the terminals on the line are busy.

For BSCA switched lines, the user may specify a list of valid switched line identification characters. When a connection is established to a terminal on the line, the identification characters of the terminal are validated against the list.

Communications management also monitors program-selected terminals for input and selects them for output. *Selection* is the specific addressing of a terminal by communications management in order to transmit output data to the terminal.

See index entry *terminal types* for additional information about command terminals and program-selected terminals.

Symbolic Terminal Naming: CCP communications management allows application programs to refer to terminals by 6-character symbolic names, allowing the application program to be independent of the specific terminal among a group of like terminals (all are 2740 Model 1; all are 3270; . . .) with which it is communicating. Communications management resolves the symbolic name into the physical address of the terminal based on the entries in an internal table which is filled in at assignment time (see index entry *TERMNAME statement*). The actual terminal assigned to a symbolic name may be changed by the system operator during the running of the system (see index entry *assign a symbolic terminal name to a terminal*). In order to change the name of a command terminal, the terminal operator must also enter a command (see index entry */NAME command*).

Communications Service Requests: The CCP provides a subroutine to application programs written in RPG II, COBOL, and FORTRAN IV which these programs call whenever they require a communications service of some kind. This *communications service subroutine* puts the user's request into a standard format (independent of the language in which the request was made) which can be interpreted by the CCP. See *Writing Communications Programs.*

After encoding the information for a communications service request, CCP communications management calls the IOCS routine to perform the physical I/O. The CCP schedules I/O operations, determining which request will be honored next by chaining the parameter lists provided by the various requestors.

Data Code Translation: CCP communication management translates the transmission line data code to the internal EBCDIC code required for System/3 processing and vice versa. The user has the option of specifying that no translation take place (see index entry *translation*).

Buffer Management: CCP communication management reserves and releases main storage areas for I/O buffers as required by programs running under the CCP. Since the buffers are allocated on an as-required basis, a program may be temporarily suspended if sufficient main storage is not immediately available to satisfy an I/O request to a terminal. The program will be resumed when sufficient main storage becomes available. See *MLTA/BSCA Communications* for additional information about transmission modes, blocking, and buffer allocation.

Online Terminal Testing: CCP communications management provides the terminal operator or system operator with the means to initiate MLTA and BSCA online terminal testing to test communication line connections and terminal operation (see index entry *Start and Stop Online Terminal Test*). Results from the MLTA online test appear at the terminal for the terminal operator to analyze. BSCA online test results may appear at the terminal (depending on the terminal type) and are also logged on the system output device for the system operator to analyze.

File Management

CCP file management includes all control functions provided by the CCP which are related to the use of disk and unit record files by user tasks. CCP file management handles the special scheduling problems that arise when two or more concurrently-executing programs are using the same disk device. In addition, CCP file management protects data in the files from errors that could result from conflicting operations by contending programs.

Opening and Closing Files: CCP begins its file management functions at startup time (see index entry *startup*), when all files available to potential programs are opened. The CCP retains sufficient file information in main storage so that when a user program requests the file, a minimal amount of time is required to perform open operations for the program. When the requesting program is finished with the files, they are "CCP-closed" by CCP file management (in reality, they are not finally closed until CCP shutdown).

Intercepting I/O Requests: During CCP operation, the CCP file management function intercepts all requests from user application programs for I/O operations. This function is performed by a CCP service subroutine, link-edited with the user program. For disk I/O, the subroutine informs the CCP file management function that an I/O operation is to begin or that an operation has just ended so that contention for disk devices can be managed by the CCP. For unit record I/O, the CCP ensures that the device is ready (if not, CCP issues a message to the system operator, telling him to ready the device) and calls the appropriate Disk System data management routine.

Sharing Access to Disk Files: When two or more user tasks are executing concurrently, it is possible for them to share the use of data files. Sharing of data files is managed by CCP file management. Some tasks can share the use of files while others cannot, depending on how they process the file. In general, input files can be shared, while output files cannot be shared. Update files can be shared, but CCP file management manages contention for the file that might cause conflicting updates. Only one of the programs sharing a disk file may add records to the file, but another add program will be allowed to add records to the file when the first add program terminates.

The CCP enables tasks to share a disk file by protecting the disk sectors actually being operated upon by one task from being accessed by other tasks. Requests from other tasks for disk sectors already in use are queued by CCP file management. The protected sectors are released and access to them is given to the first task on the queue when the using task finishes processing the data in the sectors.

Managing Physical and Symbolic Files: CCP file management is also responsible for associating the proper physical file with a symbolic file when the terminal operator specifies a FILE command prior to his program request. See index entry *FILE command* for a description of the use of symbolic files.

Program Management

The CCP program management functions include the verification of terminal operator requests for programs, loading of the programs, allocating the necessary system resources to programs, initiating operation of the programs, deallocation of system resources, purging programs from the system, and maintaining a record of the use of programs.

Program Requests: When a terminal operator or the system operator request the execution of a program, the CCP first validates that the requested program was defined to the system during assignment (see index entry *PROGRAM statement*). When the request has been validated, it is placed on a queue with other program requests which are pending (if queuing was specified — see index entry *queue and no queue commands*). The queued requests are honored by the CCP in FIFO (first-in, first-out) sequence, as the resources required by each program are available. Second and succeeding requests for queued or active multiple requester programs, however, are honored before other requests.

Allocation/Initiation/Termination: When a pending program request reaches the top of the queue, the CCP allocates system resources to the program based on the description of the program given during the CCP assignment stage. Program execution is initiated by the CCP when *all* the required resources, main storage space, disk and unit record devices, and terminals are available and when file usage requirements can be met. Unit record devices are allocated to only one program at a time; disk devices may be allocated to more than one program at a time if the processing to be done by the programs does not preclude sharing (see index entry *sharing access to disk files*). If the program currently executing under the CCP requires dedicated use of the user program area, the next program cannot be initiated until the current program has terminated, even though all the required resources have been allocated to it.

Requests for program which can support multiple requester terminals (see index entry *multiple requester program*) may be queued (see index entry *queue and no queue commands*) if the program is already servicing the maximum number of requestors.

When a program has completed its processing and no other requests for the program are pending, the CCP releases the resources used by the program and makes the main storage area occupied by the program available for use by another program. The exception is never-ending programs, whose main storage remains unavailable for the duration of the CCP run.

Program Usage Count: If this option is selected during generation, CCP program management will accumulate a record of the number of times each program is requested. The count can be printed and/or reset to zero by running the Assignment List program (see index entry *Assignment List program*).

SHUTDOWN

CCP shutdown is initiated by command from the system operator (see index entry *SHUTDOWN command*). The function of shutdown is to terminate the operation of the CCP after all user programs running under the CCP at the time the SHUTDOWN command is issued have completed their execution.

The following specific operations are performed at CCP shutdown:

- Communication lines and adapters are closed
- Disk file DTFs in main storage are restored
- Disk data files are closed
- If the program usage count option was selected during generation, the number of requests for each user program during the CCP run are added to the previous accumulated count
- Executing programs are notified to go to end-of-job
- The CCP sends a closing message to the system operator
- The CCP exits to the system end-of-job routine

PLANNING FOR CCP OPERATION

Planning for operation of the CCP is primarily a system design function. Factors which promote efficient operation of the CCP include organization and placement of disk files; the number, size, and complexity of programs; types of communications adapters and terminals used; size of system versus the amount of work to be performed by the CCP; and many other factors. System design for the CCP is described briefly in *Chapter 2: Designing Your Communication-Based System.*

Minimum System Requirements

The minimum system requirements for CCP operation are listed in *Appendix A.*

Chapter 7: Using the CCP from a Terminal

Two classes of terminals were defined previously, based on whether or not the terminals are capable of entering commands to the CCP, *command terminals* and *non-command terminals* (see index entry *terminal types*). Non-command terminals are capable only of transmitting or receiving data under control of an application program; they are not capable of commanding CCP services. When non-command terminals are not communicating with an application program, they are in a stand-by mode (not polled by the CCP for input). Since the operator of a non-command terminal does not interact with the CCP, this chapter deals only with the operation of command terminals.

Although there are operating differences among the various terminal types which can be used as command terminals, the functions that can be performed by them are the same. The primary function of any command terminal is to request the execution of application programs. All of the activities a terminal operator performs are related to that function.

INITIAL MODE

When a command terminal is *online,* it is physically attached to the system and logically attached to the CCP. The CCP monitors it continuously for program requests. When the operator wishes to request a program, he "signs on" at the terminal. Signing on involves communication between the terminal operator and the CCP. Before and during this communication, the terminal is in *initial mode.* Commands for CCP services other than system operator communication cannot be issued from a terminal when it is in initial mode. The end of initial mode occurs when the terminal operator has successfully signed on.

From the point of view of the terminal operator, signing on may be as simple as entering the ON command. However, if access to the system from a terminal must be limited to certain authorized people, the sign-on procedure may involve performing the additional steps required by a security feature. The security feature may be either the password security option provided by the CCP (see index entry *password security option*) or a routine written by the user to control access to the system in some other way.

COMMAND MODE

When a terminal operator has successfully signed on, the terminal is in *command mode*. This means the operator can request the CCP to load and execute programs and can issue related commands. Once a terminal is in command mode, the CCP communicates with that terminal until a program request is made from the terminal (see *Terminal Operator Commands*) or until the operator signs off.

DATA MODE

Once the terminal operator has issued a command to load and execute a user application program and the CCP has accepted the command, the CCP loads the program and gives it control. At that point, the terminal enters *data mode*, that is, the terminal is in communication with the application program itself. The nature of the communication is, of course, determined by the application program. Normally the terminal remains in data mode until the application program has completed its processing or has released the terminal, when the terminal is again placed in command mode to issue another program request.

COMMAND INTERRUPT MODE

The operator of a terminal need not wait until a program he requested completes its job in order to interrupt it. By entering a string of six characters which are significant to the CCP (determined by the user at generation time), he can indicate that he wants to enter *command interrupt mode* (see index entry *data mode escape*). While in this mode, he can send messages to the system operator, resume execution of the program, or cancel the program (in the case of a multiple requester program, he cancels his communication with the program).

TERMINAL OPERATOR COMMANDS

Two logical groups of terminal operator commands can be issued after sign-on. First, while the terminal is in command mode, before a program request is actually made, the terminal operator can issue various commands pertaining to the subsequent program request:

- He can tell the CCP hwo to handle his request if it cannot be honored immediately (see *Queue/No-Queue Commands*).
- He can issue commands that indicate which files are to be accessed by the programs he requests (see *File Specification Command*).
- He can tell the CCP by what name his terminal is known to the program he is requesting (see *Terminal Name Command*).
- He can send a message to the system operator (see *Message Command*). This may also be done prior to sign-on.

The second group of commands is used during command interrupt mode. After data mode escape, the terminal operator can:

- Send one or more messages to the system operator.
- Cancel the program (see *Cancel Command*) or cancel his communication with a multiple requester program.
- Resume execution of the program (see *Run Command*).

Sign-On Command (/ON)

The /ON command notifies the CCP that the terminal operator wishes to make a request of the system. If the system uses a security feature, the /ON command must be accompanied by one of the following:

1. The current password required by the CCP password security feature, or
2. Information required by a user-written sign-on routine

The CCP logs every sign-on attempt on the system operator's console, along with an indication of whether or not it was successful. If the sign-on was successful, the CCP notifies the terminal operator and allows him to enter a command. If the sign-on was not successful, the CCP allows the terminal operator to attempt to sign on again.

Once the operator has signed on at the terminal, he can make any number of requests without signing on again. However, if the terminal operator leaves the terminal unattended and access to the terminal is restricted by a security feature, he should sign off when he leaves (see *Sign-Off Command*). If he signs off, he must sign on again when he wants to use the terminal the next time.

Queue/No-Queue Commands (/Q and /NOQ)

The /Q or /NOQ command indicates how the CCP is to handle program requests from this terminal which cannot be honored immediately:

/Q — Place the request on a queue and honor it as soon as possible

/NOQ — Reject the command if it cannot be honored immediately and allow the operator to enter another request

A /Q or /NOQ command remains in effect until the terminal operator enters a different /Q or /NOQ command or until he signs off. If neither a /Q nor a /NOQ command is entered at the terminal, the CCP assumes the /NOQ option.

File Specification Command (/FILE)

The /FILE command specifies which of several data files to use on a current program run. The terminal operator may use the /FILE command to vary the files which are used by the programs he requests. The /FILE command cannot be used with multiple requester programs.

Certain application programs are written to access any of several files containing similar data in the same format by referencing a *symbolic file* in the program. For the program to actually access a file, the name of the symbolic file must be associated with the name of a file which actually exists on disk, a *physical file.*

Suppose, for example, a school system has a separate student records file for each school. A student report program can process the student records file from any of the schools, but it must be told by a /FILE command which of the files to use on a particular run (see Figure 14).

A /FILE command is in effect for all subsequent program requests from that terminal until the terminal operator enters a new /FILE command for the symbolic file or until he signs off from the terminal. Thus, the /FILE command may apply to more than one program.

If a terminal operator enters a /FILE command without naming any files, he cancels all file entries currently maintained for that terminal by the CCP.

If he enters a /FILE command which gives only a symbolic file name with no associated physical file name, that symbolic file entry is deleted from the list of file entries maintained by the CCP for that terminal. Signing off from the terminal cancels all file specifications which were in effect for that terminal.

The CCP informs the terminal operator if the /FILE command he entered was invalid or if the CCP cannot accept additional file specifications.

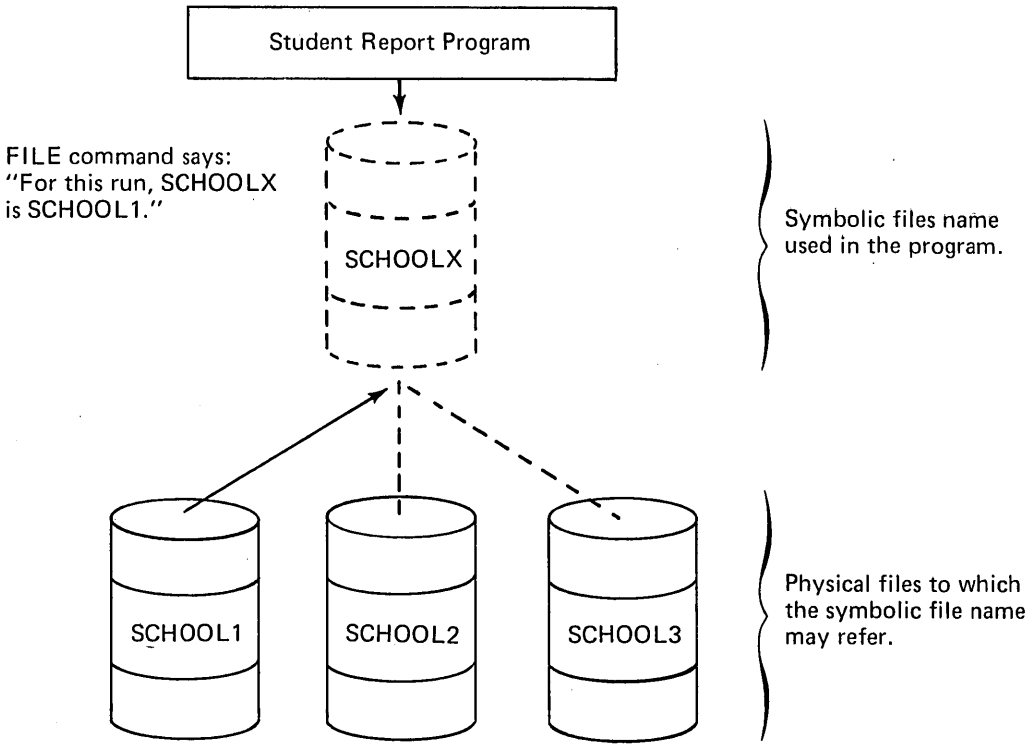


Figure 14. Illustration of FILE Command

Terminal Name Command (/NAME)

The /NAME command tells the CCP which of a set of previously-assigned symbolic names to use as the symbolic name for this terminal during this "session" (from sign-on to sign-off, or until another /NAME command is entered). The CCP passes the symbolic name to any program that requests input from the terminal by means of a Get General I/O operation (see *Chapter 3: Writing Communications Programs* for an explanation of the terminal I/O operations). In this way, the program identifies which terminal requested it.

The CCP maintains a list of symbolic names eligible for each terminal. These names are associated with the terminal during CCP assignment. The primary purpose of the /NAME command is to allow a terminal to assume the name of another terminal (of a similar type) that is currently not operable.

Program Request Command

The terminal operator requests a program to execute by entering the name of the program. The name may be from one to six characters in length. The first character must be alphabetic (including the characters # and @); the remaining characters may be alphabetic (including #, $, and @) and numeric in any combination.

If a program allows, program input data can be entered with the program request.

Before the operator enters a program request, he should consider whether /Q or /NOQ, /FILE, or /NAME commands are required for that program. He should determine this from a document such as a program run sheet for that program (see *Planning Considerations*), if he is not familiar with the requirements of the program.

The CCP informs the terminal operator by a message if it has rejected the program request for any reason or if it has queued the program for execution.

Data Mode Escape Command

The terminal operator can interrupt an application program (when the terminal is in data mode, under control of the program) by entering a predetermined set of six characters. The string may consist of any six characters which have been defined to the CCP as the data mode escape characters during generation. While a requester terminal is in data mode, the CCP continuously monitors input from the terminal for appearance of the characters as the first six input characters.

After the terminal has interrupted the application program, the terminal is in command interrupt mode and the operator can:

- Send one or more messages to the system operator (see *Message Command*)
- Resume execution of the application program (see *Run Command*)
- Cancel the application program he interrupted (see *Cancel Command*).

Cancel Command (/CANCEL)

The /CANCEL command can only be issued while the terminal is in command interrupt mode; it is invalid at any other time.

The /CANCEL command has the following effect:

1. If the requested program is a multiple requester program, the terminal is released from the program and the terminal operator is free to enter another program request or other commands.
2. If the requested program is not a multiple requester program, the program is terminated immediately and the terminal operator is free to enter another program request or other commands.

The CCP notifies a multiple requester program when a terminal has issued a /CANCEL command before it accepts any further commands from the terminal.

Message Command (/MESSAGE)

The /MESSAGE command is used by the terminal operator to send a message to the system operator. The text of the message follows on the same line. The /MESSAGE command can be entered any time the terminal is in initial mode, command mode, or command interrupt mode.

Run Command (/RUN)

The /RUN command causes an application program to resume reading input data from a terminal after the terminal has issued a data mode escape command. This command causes the terminal to return to data mode from command interrupt mode.

Sign-Off Command (/OFF)

The /OFF command causes the terminal to be returned to initial mode or to be placed offline. The /OFF command may be accompanied by either the word HOLD or the word DROP:

HOLD – The terminal is returned to initial mode. The CCP will accept an /ON command or a /MESSAGE command from the terminal, but no other command will be accepted.

DROP – The terminal is placed offline; that is, the CCP will accept no commands from the terminal until it is again placed online by the system operator (see index entry *change the status of a terminal*). At that time, it enters initial mode.

If neither HOLD nor DROP is entered, the CCP assumes the option which was selected during the CCP assignment stage (see index entry *TERMINAL statement*). All /OFF commands are logged on the system operator's console with an indication of whether HOLD or DROP was used as the /OFF option.

The /OFF command may be entered any time after the /ON command, while the terminal is in command mode. It may not be entered following a data mode escape command unless the /CANCEL command has been issued.

The /OFF command clears all file specification entries in effect for this terminal (see *File Specification Command*).

PLANNING CONSIDERATIONS

Since the terminal operator will probably be calling programs he did not write himself, he must have access to information about the programs, perhaps by means of a "program run sheet" for each program he can call. This sheet will be unique to each installation, but will probably contain at least the following information:

- What, from the terminal operator's point of view, is the function of the program?
- Should the program request be queued if it cannot be honored immediately, or should it be rejected (should the terminal operator enter a /Q or /NOQ command)?
- Does the program have to be told which files to use? (Should the terminal operator enter one or more /FILE commands?)
- Does the program expect to be requested by this terminal (must the terminal operator enter a /NAME command)?
- Should the terminal operator enter input data for the program at the same time he makes the program request?
- What kind of input (and in what format) does the program expect from the terminal operator?
- Are there any exceptional conditions or special situations which can arise from the program (must the terminal operator notify the system operator in certain situations? Does the program diagnose error conditions, and what are they and what must the terminal operator do?)

In addition to a "program run sheet" for each program he can call, the terminal operator will also require some current information about the system, such as:

- If the password security option is in effect, what is the current password?
- If the installation has its own unique sign-on procedure, what is it?
- What is the current set of terminal name assignments for the terminal?
- What are the data mode escape characters for the system?
- Should the terminal operator specify HOLD or DROP when he signs off – what is the current default for the terminal? – perhaps he should not sign off until the end of the day.

The terminal operator may also require information about the schedule of work he is expected to complete.

- Are there certain programs he must call each day?
- Must he call programs in any particular order?
- Is his use of the terminal limited in any way? – to certain hours or to a certain amount of time per use, for example?

The system operator has an important role in the operation of a communication-based system under control of the CCP. Through the 5471 printer keyboard, the system operator initiates the activity of the CCP, monitors its operation throughout the run, and shuts it down at the end of the run. In addition to the operational control of the CCP, the system operator may take part in the generation and assignment stages of the CCP. See *Chapter 4: Generating the CCP* and *Chapter 5: Making an Assignment Run* for the required procedures and statements.

INITIATING THE CCP

The system operator starts the CCP by entering OCL statements after Disk System IPL (initial program load). The OCL statements cause the CCP to be loaded and supply FILE OCL statements for all disk files used by the programs to be run under the CCP.

After the startup routine has been loaded, it asks questions of the system operator that allow him to exercise several options. He can:

- Select the assignment set to be used for this run.
- Change the maximum number of concurrent user programs allowed by the set (unless the maximum for the system is one).
- Change the password assigned in the set.
- Suppress access to certain disk data files that are normally accessible to programs in this set.
- Suppress access to certain application programs normally available in this set.
- Suppress the use of certain communication lines in the set.
- Suppress the use of certain terminals in the set.
- Change the main storage allocation for communications buffer area and user program area assigned in the set.

Changes specified to a set during startup do not permanently alter the assignments in the set; the changes apply only to the current CCP execution. The system operator may respond to the first question by entering a special command to allow the CCP to begin execution using the assignment set used in the last CCP execution (along with the changes or suppressions specified in the startup procedure for that execution). When this facility is specified, the CCP assumes that *no* changes have been made to the assignment set or to any programs, files, lines, or terminals since the last CCP execution. The purpose of this facility is to minimize the time required to start the CCP.

SYSTEM OPERATOR COMMANDS

After start-up of the CCP, the system operator can interact with the system in various ways; he can:

- Communicate with a remote terminal by telling the CCP to send a message to the terminal and by receiving messages from the terminal
- Inquire into the status of the CCP at any time
- Modify the status of the CCP
- Function as a remote terminal by entering program requests (see *Using the System Operator's Console as a Requesting Terminal*)

The system operator sends messages, makes inquiries, and modifies the status of the CCP by entering commands using the 5471 printer/keyboard. All commands consist of an operation code and one or more operands. The operation codes can be used in either their full-length versions or in abbreviated versions. In some cases, operands can also be abbreviated. Specific operation codes and operands will be given in a later publication.

Message Command

The message command is used by the system operator to send a message to a terminal. The operator designates which terminal is to receive the message by including either the symbolic terminal name or the CCP's terminal reference identification with the message text.

The CCP notifies the system operator when the message cannot be transmitted because the terminal is either offline, has no output capability, is under control of an application program, or does not exist in the system.

Display Outstanding Reply Requests

This command causes the CCP to identify the tasks which are currently awaiting a reply to a message they have sent to the system operator. The CCP replies by writing a list of task identifications on the system operator's output device. If there are no outstanding reply requests, the CCP informs the system operator.

Display Queued Program Requests

This command causes the CCP to list the symbolic name of each terminal for which a program request has been queued, along with the name of the program it requested. If there are no queued requests, the CCP informs the system operator.

Display Terminal Status

This command causes the CCP to print the following information about a specific terminal or about all terminals defined to the CCP:

- Symbolic terminal name
- Identification of the task using the terminal
- Encoded attributes and status of the terminal. This includes information about the terminal such as whether it is online or offline, signed on, awaiting some event, or what operating mode it is in.

Display Terminal Assignments

This command causes the CCP to print the symbolic names and reference identification of a specific terminal or of all terminals in the system. The system operator can use this command to display terminal assignments when he wants to enter a command to change the terminal which is actually addressed by a particular symbolic name (see *Assign a Symbolic Name to a Terminal*).

Display System Status

This command displays information about either an individual user task named in the operand or for all user tasks currently in main storage:

- Task identification (a single character which identifies the task)
- Status information for the task (such as, is the task running or suspended?)
- Program name
- Amount of main storage occupied by the task
- Number of terminals used by the task
- Number of files used by the task

If an individual task is named in the operand, the CCP also prints out the following information for the task:

For each terminal the task is using:

- Symbolic terminal name
- CCP terminal reference identification assigned to the terminal during CCP assignment
- Attributes and status of the terminal

For each file the task is using:

- Filename
- Whether the file has been allocated and opened
- Information about the file, such as file type and file organization

Cancel a User Program or the CCP

This command can be used by the system operator to cause a particular user program to cease processing or to immediately stop the CCP. The CCP informs the system operator if he has entered an invalid task identification and program name to cancel a user program.

Suspend Requests/Execution/Initiation of User Programs

This command allows the system operator to suspend user activity in any of the following ways:

- Suspend the execution of all user programs currently running under control of the CCP
- Suspend the execution of a particular user program running under the CCP
- Stop the initiation of user programs under the CCP
- Do not allow additional program requests from requesting terminals

The CCP informs the system operator if he has entered an invalid task identification and program name or if suspension is already in effect for the programs he specified.

A program can be cancelled after its operation has been suspended.

Resume Requests/Execution/Initiation of User Programs

The resume command allows the system operator to cause user program activity to be resumed after a previous suspension, as follows:

- When execution of all user programs running under the CCP was previously suspended, cause the execution of all programs to be resumed
- Cause execution of a previously-suspended program to be resumed
- Allow the CCP to resume initiation of pending program requests
- When program requests from command terminals were previously suspended, allow command terminals to resume program requests

The CCP informs the system operator if he has specified an invalid task identification and program name or if the resumed task was never suspended.

Change the Status of a Terminal

This command allows the system operator to change the status of a terminal to online or offline. The command is not allowed if the terminal is currently allocated to a program or awaiting a queued program request.

The CCP informs the system operator of the status of the terminal following the command.

Assign a Symbolic Terminal Name to a Terminal

This command allows the system operator to assign a symbolic terminal name to a specific terminal. This command is used, for example, to assign an alternate terminal when a particular terminal is inoperative. This command is invalid when:

- The symbolic terminal name is currently being used by a program running under the CCP
- The symbolic terminal name is the only name assigned to an online terminal which is in command or initial mode or has a queued program request
- Changing the assignment would cause a terminal unit allocation conflict with a current request for a never-ending program

In order to change the name of a command terminal, the terminal operator must also enter a command (see index entry */NAME command*).

Start and Stop Online Terminal Test

The START command allows the system operator to initiate the online test facility for terminals that are supported by either the MLTA or BSCA IOCS (see index entry *online terminal test*). The STOP command stops the online test of MLTA terminals. The operator can specify that either a single or multiple online tests be run for MLTA devices. The CCP informs the system operator either when the command is invalid for any reason or when the test has been started and stopped successfully.

Online terminal test can also be initiated directly by the terminal operator if he suspects that his terminal is not operating correctly or if he wishes to test it before he transmits data. The terminal must be capable of input and output. The CCP or the application program are not informed of a test that is initiated by the terminal operator. Certain terminals may have restrictions as to what online tests can be executed from the central system or requested by the terminal operator.

USING THE SYSTEM OPERATOR'S CONSOLE AS A REQUESTING TERMINAL

The CCP allows the system operator to request programs using the 5471 printer/keyboard in the same way a terminal operator requests programs. The following commands are available to the system operator:

- /Q
- /NOQ
- /FILE
- Program Request

The effect of these commands is the same as described for the terminal operator (see index entry *Terminal Operator Commands*).

Certain restrictions and assumptions apply to using the operator's console to request programs:

- The operator's console is always considered to be signed on.
- The operator's console cannot be in data mode or command interrupt mode. All requests by the user program for input data from the console must be made by a Put-Then-Get operation issued to the console.
- After the console has requested a program and before it is released by the program or the program terminates, the console cannot enter terminal commands (that is, other program requests), but can perform other system operations.

SHUTTING DOWN THE CCP

When it is time for the system operator to stop the CCP, he issues the command, SHUTDOWN. The SHUTDOWN command allows all programs that are currently running under the CCP to continue until they terminate themselves. Following a SHUTDOWN command, the CCP informs the user program via a return code that the system operator wants the program to terminate as soon as possible. It is the responsibility of each user program executing at that time to recognize the return code and perform the necessary termination action. No new program requests are accepted.

The system operator can shut down the CCP without allowing all programs to continue to completion by issuing a Cancel command (see *Cancel a User Program or the CCP*).

The CCP will not go to end-of-job until all executing programs have gone to end-of-job or have been cancelled.

SYSTEM OPERATOR MESSAGES

The system operator can receive messages from several sources while he is monitoring the CCP operation. Some of the messages require a response; some do not.

CCP Responses

One type of message the system operator can receive is a CCP response to a command he has issued which inquires into or modifies the status of the CCP. These responses can be either confirmations from the CCP that it has carried out a command or error messages when the CCP encounters an error in the command. These messages contain the identification of the task that issued the message.

Messages from Terminal Operators

The system operator can also receive messages sent by a terminal operator using the /MESSAGE command. These messages are accompanied by the symbolic name of the terminal that issued the message. These messages may or may not ask for the system operator to return a message. If a response is required, the system operator also issues a /MESSAGE command, accompanied by the symbolic name or CCP reference identification of the terminal that is to receive the message.

Messages from User Programs

A user program that requires input data from the system operator (whether the program was requested by the system operator or by a terminal operator) must issue a Put-Then-Get operation to the system operator describing the input it requires. The message is prefixed by an asterisk (*) to indicate to the system operator that a response is required. The CCP prefixes the message with the task identification and program name of the message's origin. This kind of a message does not force an immediate response from the system operator; he must press the REQ key to indicate he is ready to respond.

In order to respond to a message from a user program, the system operator enters the task identification given in the output message and the appropriate text.

Disk System Management Halt Codes

Halt codes issued by programs running under the CCP are printed as messages to the system operator rather than being displayed in the message display unit and halting the entire program level. This enables the CCP to halt the execution of only one program, allowing other programs to continue running. The system operator also enters his response to the halt message on the operator's console, instead of replying via the system console data switches as under disk system management.

PLANNING CONSIDERATIONS

The CCP system operator requires a deeper understanding of the system than the operator of a batch system. He must make decisions on his own in a variety of situations. Many of these decisions will require a thorough understanding of the method of operation of the CCP. The system operator has the ability to display and modify the current status of the CCP, so he must thoroughly understand the effect of his actions on the CCP and on the information processing system as a whole.

The system operator should be involved as early as possible in planning for installation of the CCP. Prior to operating the system, he should become acquainted with the functions of the application programs in the system and with the files used by each program. He must be familiar with the configuration of the system and with the current status of the system and the current system assignments.

To keep the communication-based system running smoothly when the system operator is absent, a backup system operator should be available. This may be the system manager or one of the terminal operators or programmers.

Operating Aids

The system operator must have certain current information about the system available at all times, including descriptions of the programs available for use under the CCP, the current system assignments, and other current system operating information.

A "program description sheet" for each application program should include at least the following information:

- Symbolic name of the program
- Function of the program (including how it affects the files it uses)
- System resources used by the program. This should list the files used by the program, the terminals used, how the terminals are used, and what names the program uses for the terminals. The main storage requirement of the program and its typical operating time should also be given.
- Does the program require one or more FILE statements?
- What kind of input does the program expect – can input data be entered at the same time the program request is made?
- Are there any special considerations? Is the use of the program restricted in any way? Are there potential problems involved in suspending or cancelling the program before it has finished its run?

In addition to program definitions, the system operator needs other information about the system. He should have a copy of the current CCP assignments to provide him with information about the terminals attached to the system, the lines available on the system, the files available, and the programs used under the CCP.

The system operator also needs current system information like:

- What is the current password (or other security information, if the installation has its own security procedures)
- What are the current data mode escape characters for the system?
- What should be the current default for sign-off from each terminal – HOLD, or DROP?
- Is there a certain schedule of work to be completed? This can be a composite schedule of the work to be performed by each terminal in addition to work to be performed by the system operator, to inform the system operator of the total work schedule, since he is in a position to answer questions from the terminal operators.

Appendix A: Devices and Programs Supported and Required

TERMINALS AND FEATURES SUPPORTED

The following terminals may be used with the communications control program (this list of terminals and features supersedes the list given in the *General Information Manual,* GC21-7578-0).

Through the multiple line terminal adapter:

- 1050 Data Communication System
  - Multipoint switched
  - Multipoint nonswitched
- 2740 Communication Terminal Model 1
  - Basic
  - Basic with checking
  - Dial
  - Dial with checking
  - Dial with transmit control
  - Dial with transmit control and checking
  - Station control
  - Station control with checking
- 2740 Communication Terminal Model 2
  - Basic
  - Buffer receive
  - Checking
  - Buffer receive and checking
- 2741 Communication Terminal
  - Basic
  - Switched
- Communicating Magnetic Card SELECTRIC® Typewriter
  - Point-to-point switched
- System/7
  - Appears identical to a 2740 Model 1

With the binary synchronous communications adapter:

- 3270 Information Display System
  - Multipoint nonswitched
- 3735 Programmable Terminal
  - Point-to-point switched
  - Multipoint nonswitched
- System/3
  - Point-to-point switched
  - Point-to-point nonswitched
  - Multipoint with the CCP as control station
  - Multipoint with the CCP as a tributary
- System/7
  - Point-to-point switched
  - Point-to-point nonswitched
  - Multipoint with the CCP as control station
- System/360, System/370
  - Point-to-point switched
  - Point-to-point nonswitched
  - Multipoint with the CCP as control station
  - Multipoint with the CCP as tributary

Terminals that are equivalent to those explicitly supported *may* also function satisfactorily. The customer is responsible for establishing equivalency. IBM assumes no responsibility for the impact that any changes to the IBM-supplied products or programs may have on such terminals.

SYSTEM DEVICE AND PROGRAM REQUIREMENTS

Device Requirements

The following is the minimum device configuration necessary for a communications-based system using the CCP:

- 5410 Model A15 Processing Unit with 24,576 bytes of main storage
- One 5444 Model 2 Disk Storage Drive
- 5471 Printer-Keyboard
- 5424 Multi-Function Card Unit (MFCU) or 1442 Card Read/Punch (not required during CCP operation)
- 5203 or 1403 Printer (not required during CCP operation)
- Multiple-Line Terminal Adapter RPQ (RPQ numbers S40028, S40033) or one Binary Synchronous Communications Adapter
- At least one communications terminal of a type listed under terminals and features supported

With the above configuration, no more than one application program may be executing at a time. The minimum main storage size in which concurrent execution of more than one program is supported is 32,768 bytes (5410 Model A16).

Additional Devices Supported

The following device facilities are supported by the communications control program:

- Up to 65,536 bytes of main storage
- Up to two 5445 Disk Drives (for data files only)
- An additional 5444 Model 2 or 3 Disk Storage Drive
- Both 5424 MFCU and 1442 Card Read/Punch
- 120 or 132 print positions on the 5203
- As many as two Binary Synchronous Communication Adapters, and one Multiple Line Terminal Adapter with up to eight lines
- Dual Program Feature (see note)

Note: The communications control program does not require the dual program feature to allow more than one program to be executed at a time. Use of the dual program feature is not prohibited during execution of the communications control program, but any program executed in the other program level does not run under control of the communications control program. The communications control program can not be run in both program levels concurrently.

System Programs Required

Execution of the communications control program requires IBM System/3 Model 10 Disk System Management, including all transient modules for the appropriate IOCS.

Generation of the communications control program requires IBM System/3 Model 10 Disk System Management including:

- Macro Processor (feature codes 6020, 6021)
- Overlay Linkage Editor (feature codes 6026, 6027)
- The appropriate communications IOCS (program number 5799-WAU for MLTA, and/or program number 5702-SC1, feature codes 6030, 6031 for the Model 10 Disk System BSCA).

No special programming systems requirements exist for the running of system assignments.

For the preparation of application programs, an applicable compiler or assembler is required.

Appendix B: Glossary

This glossary contains only terms that have a special meaning related to the CCP. Other communications and data processing terms used in this publication are defined in *IBM Data Processing Glossary*, GC20-1699.

command interrupt mode: The operating mode of a terminal following data mode escape until the program execution is resumed by a RUN command (the terminal reenters data mode) or until the program is cancelled by a CANCEL command (terminal enters command mode).

command mode: The operating mode of a terminal following a successful sign-on, up to and including the program request. Following program termination, a terminal returns to command mode until another program request is made or until sign-off.

command terminal: A terminal that is capable of commanding CCP services related to requesting a program. Terminals are designated command or non-command terminals at assignment time.

communications management: A major function of the CCP that controls terminal input-output.

communications service subroutine: A relocatable subroutine provided by the CCP that is link-edited to user programs written in RPG II, COBOL, or FORTRAN IV. The subroutine is called by the user program whenever the program requires a communications service, enabling programmers to request communications services in these languages. A separate subroutine is provided for COBOL, FORTRAN IV, and RPG II; a macro is provided for Basic Assembler.

data entry application: A communications-based system application in which terminals are in continuous operation (as opposed to the typical inquiry application). Data entry applications include document preparation (such as invoice writing) and entering data directly into data files from a terminal, such as in creating files.

data mode: The operating mode of a terminal when it is under control of a user program, until the program terminates, the terminal is released by the program, or the data mode escape characters are entered. While in data mode, a terminal is not in direct communication with the CCP.

data mode escape: A special CCP command, consisting of a unique string of six characters entered at a terminal while the terminal is in data mode. The data mode escape command interrupts the execution of the application program and places the terminal in command interrupt mode.

disk system management: The group of system programs which control the operation of the IBM System/3 Model 10 Disk System. Disk System management performs scheduling, input/output control, storage assignment, data management, and related services.

file management: A major function of the CCP that controls the use of data files by programs running under the CCP.

initial mode: The operating mode of a command terminal before a sign-on at the terminal has been accepted by the CCP.

inquiry: A communications-based system application in which a request for information is entered from a terminal and a response is returned to the terminal.

inquiry-with-update: A communications-based system application in which records of transactions entered from terminals are used to interrogate and update one or more master files maintained by the system (synonymous with *inquiry and transaction processing*).

multiple requester program: A type of application program under the CCP that can process requests from more than one requester terminal concurrently.

never-ending program: A user application program which, after it has been requested, remains in main storage until the CCP is shut down or the program is terminated by the system operator.

non-command terminal: A terminal that is not capable of commanding CCP services. A non-command terminal is always either in stand-by mode (not polled for input by the CCP) or in data mode (under control of an application program).

order entry application: A form of data entry application in which transactions (such as sales orders) are entered into a data file from remote terminals.

password security option: A optional CCP feature, selected during CCP generation, which requires a terminal operator to enter a predetermined password before the CCP will accept input from the terminal.

physical file: See *symbolic file.*

program management: The major function of the CCP that fetches programs, allocates system resources to programs, purges programs from main storage, and optionally maintains a count of the number of times each application program is executed.

program request: A command, consisting of a program name entered at a terminal or the system operator's console, that causes the CCP to initiate execution of an application program.

program usage count: The optional CCP program management function of maintaining a count of the number of times each application program is executed.

program-selected terminal: From the point of view of the application program, a terminal that is selected by an application program for input/output, as opposed to a terminal that requested the program (see *requester terminal*). Program-selected terminals can be either *required* (must be allocated to the program before the program can run) or *acquired* (allocated dynamically to the program as it is running).

requester terminal: From the point of view of the application program, a terminal that requested the program, as opposed to a terminal that is selected by the program (see *program-selected terminal*). Requester terminals are always command terminals.

sign-on: The procedure performed at a terminal while it is in initial mode. This procedure may include entering only the /ON command, or entering the /ON command with a password or other user-specified security data.

single requester program: A type of application program under the CCP that can process a request from only one requester terminal at a time.

symbolic file: A symbolic name in an application program which can, on separate executions of a program, refer to different files, known as *physical files.* A symbolic file is related by the terminal operator to a specific physical file by means of a /FILE command.

system task: A unit of work for the processing unit from the standpoint of the CCP, consisting of a CCP function (as opposed to a user application, or *user task*) that must be performed by the CCP, such as communications management.

task identification: An identifying character associated with a task which differentiates between that task and other tasks running concurrently under the CCP. System tasks are identified by an alphabetic character; user tasks are identified by a numeric character.

task management: A major CCP function that manages the concurrent execution of two or more programs, including intercepting and routing requests from those programs for input/output.

terminal reference identification: A unique two-character identifier, assigned to each terminal during the CCP assignment stage, that is used by the CCP and the system operator to refer to a specific terminal. Any of the 64 EBCDIC characters may be used.

user task: A unit of work for the processing unit from the standpoint of the CCP, consisting of a user program (as opposed to a system function, or *system task*) that must be executed by the CCP.

Appendix C: Bibliography

The following publications contain information that readers may require to gain more detailed knowledge of the System/3 Model 10 Disk System, teleprocessing, telecommunications equipment, and System/3 programming languages that can be used with System/3 and the CCP:

CCP

- *IBM System/3 Model 10 Disk System Communications Control Program General Information Manual,* GC21-7578

General System/3

- *IBM System/3 Model 10 Components Reference Manual,* GA21-9103

MLTA and MLTA Terminals

- *IBM System/3 Model 10 Disk System Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual,* GC21-7560
- *IBM 2740 Communications Terminal Models 1 and 2 Component Description,* GA24-2403
- *IBM 2741 Communication Terminal,* GA24-3415
- *IBM 1050 Data Communication System Principles of Operation,* GA24-3474

BSC and BSCA Terminals

- *General Information: Binary Synchronous Communications,* GA27-3004
- *IBM 3270 Information Display System Component Description,* GA27-2749
- *IBM 3735 Programmer's Guide,* GC30-3001
- *IBM System/7 Binary Synchronous Communications Module Programming Guide and Reference Manual,* SC34-1510
- *IBM System/3 Model 10 Disk System Multiline/Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573

Programming Language Manuals

- *IBM System/3 Disk System RPG II Reference Manual,* SC21-7504
- *IBM System/3 Subset American National Standard COBOL,* GC28-6452
- *IBM System/3 Subset American National Standard COBOL Compiler and Library Programmer's Guide,* SC28-6459
- *IBM System/3 Disk FORTRAN IV Reference Manual,* SC28-6874
- *IBM System/3 Disk System Basic Assembler Program Reference Manual,* SC21-7509
- *IBM System/3 Model 10 Disk System Control Programming Macros Reference Manual,* GC21-7562

General Telecommunications

- *IBM Data Communications Primer,* C20-1668
- *IBM System/360 Introduction to Teleprocessing,* C30-2007

READER'S COMMENT FORM

IBM System/3
Model 10 Disk System
Communications Control Program
Planning Manual

GC21-7579-0

YOUR COMMENTS, PLEASE . . .

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM branch office serving your locality.

Corrections or clarifications needed:

Page *Comment*

Please include your name and address in the space below if you wish a reply.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

GC21-7579-0

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Systems Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Publications, Dept. 245

Fold

Fold

IBM System/3 Printed in USA GC21-7579-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

GC21-7579-0

IBM System/3 Printed in USA GC21-7579-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)