IBM System/3
Model 10 Disk System
Communications Control Program
System Reference

Feature 6033
Program Number 5702–SC1

GC21-7588-0
S3-36

Preface

This publication describes the Communications Control Program (CCP) feature of the IBM System/3 Model 10 Disk System and provides information to aid the system installation manager, IBM Systems Engineer, and application programmers in generating the CCP system.

The generation of the CCP system is accomplished by using the facilities of the:

- IBM System/3 Model 10 Disk System Management (5702-SC1)
- Macros Feature (feature codes 6020, 6021)
- Overlay Linkage Editor Feature (feature codes 6026, 6027)
- Programming support for the desired communication adapters: Multiline/Multipoint Feature (feature codes 6030, 6031); Multiple Line Terminal Adapter Feature (PSHRPQ number 5799-WAU)

Prerequisite Knowledge

You should be an experienced System/3 Model 10 Disk System user familiar with the basic concepts of teleprocessing.

Devices and Programs Supported and Required

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

First Edition (September 1973)

This manual and GC21-7579 obsoletes GC21-7579-0 and Technical Newsletter GN21-7680.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM Systems, refer to the latest IBM System/3 Newsletter, GN20-2228, for the editions that are applicable and current.

The Communications Control Program feature will operate with version 08, modification 00 of the IBM System/3 Model 10 Disk System, Program Number 5702-SC1, and with all subsequent versions and modifications until otherwise indicated.

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.

Related Publications

- *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference*, GC21-7579.
- *IBM System/3 Disk System Communication Control Program System Operator Guide*, GC21-7581.
- *IBM System/3 Disk System Communication Control Program Terminal Operator's Guide*, GC21-7580.
- *IBM System/3 Disk Operator's Guide*, GC21-7508.
- *IBM System/3 Model 10 Disk System Control Programming Reference Manual*, GC21-7512.
- *IBM System/3 Disk System Control Programming Macros Reference Manual*, GC21-7562.
- *IBM System/3 Overlay Linkage Editor Reference Manual*, GC21-7561.
- *IBM System/3 Disk Systems System Control Program Logic Manual*, SY21-0502.
- *IBM System/3 Disk Systems Data Management and Input/Output Supervisor Logic Manual*, SY21-0512.
- *IBM System/3 Model 10 Disk System Multiline/Multipoint Binary Synchronous Communications Reference Manual*, GC21-7573.
- *IBM System/3 Model 10 Disk System Multiple Line Terminal Adapter RPQ Program Reference and Component Description Manual*, GC21-7560.
- *IBM 1050 Data Communication System Principles of Operation*, GA24-3474.
- *IBM 2740 Communications Terminal Models 1 and 2 Component Description*, GA24-3403.
- *IBM 2741 Communication Terminal*, GA24-3415.
- *IBM System/7 Binary Synchronous Communications Module Programming Guide and Reference Manual*, SC34-1510.
- *IBM System/7 Teleprocessing Multiplexor "TPMM" Programming Guide and Reference Manual Supporting RPQ D08011*, SC34-1506.
- *IBM 3270 Information Display System Component Description*, GA27-2749.
- *IBM 3735 Programmer's Guide*, GC30-3001.
- *IBM System/3 Model 10 Disk System 3735 Application Package Coding Manual*, GC21-5096.

Additional publications related to the Disk System can be found by referring to the *IBM System/3 Bibliography*, GC21-8080.

Contents

Chapter 1. Introduction

The Communications Control Program (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: If you are not acquainted with the terms used in this introduction, you can find them explained either in *Appendix E: Glossary,* or in the *Data Processing Glossary,* GC20-1699.

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 of 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.

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 (console). 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.
- Change the status of terminals on the system.
- 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.
- Initiate an orderly shutdown of the system, or cause the immediate termination of the CCP.

The system operator can also perform some of the functions of a terminal operator. He can request programs and communicate with those programs much as a terminal operator would. However, programs requested from the console should not require the system operator to enter a significant amount of data, since his primary responsibility is controlling the system for the sake of the operators of terminals.

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.

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 of familiar functions, such as access to data files, is not changed under the CCP. Further, the types of 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: macro instructions are provided, which can be 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 to address a different terminal.

Other facilities offered by the CCP to the programmer are:

- Access to the name of the terminal which requested the program
- 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

DEVICES AND PROGRAMS SUPPORTED

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

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 and the facilities that should be included in that system.
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 assignment 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 will 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 by the system operator. During startup, the CCP routines open disk files, adapters, and communication lines and complete various tables and control blocks. During **operation**, the CCP performs the functions requested by terminal operators and the system operator, executing application programs as directed by those operators. 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.

Chapter 2. Using the CCP from a Terminal

To the operator of a terminal, the system is a resource to help him accomplish his tasks as those tasks arise. The terminal operator may think of the system as belonging to him, alone, unless the combined demands upon the system are great enough to cause some delay in its responsiveness to him. All contention among terminal operators for use of the system is managed by the communications control program.

The CCP distinguishes two types of terminals:

Command terminals, which can request services of the CCP including the running of application programs.

Data terminals, which cannot request services, and are used only as directed by the application programs.

In order to call for application programs, the command 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 the 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. The sequence of operations at the terminal, and the format of data sent to and from it, are entirely directed by the application program. The terminal operation continues to be directed by the application program until that program releases it. However, the operator while sending data from his terminal to the application program, can reestablish communication with the CCP in order to:

- Send a message to the system operator and then resume sending data to the program.
- Release his terminal from control of 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.

HOW THE OPERATOR REQUESTS A PROGRAM

When a command terminal is not in use, it is continually monitored by the control program for the presentation of a request. The operator of a terminal calls for an application program to perform a specific function by simply entering the program name at the terminal. The CCP then attempts to load and execute the program and to put that program in communication with the terminal operator. From then until the completion of the program's execution, the interaction between the terminal operator and the system is dictated by that program.

When the application program has completed execution, it yields control of the terminal to the communications control program. The CCP once again monitors the terminal for a program request. The next request may be for the same, or for a different, function.

Each terminal operator must be trained in the functions he can call upon and in the procedures for interacting with the application programs that perform those functions.

HOW THE OPERATOR REQUESTS SYSTEM SERVICES

While monitoring for program requests, the communications control program can also respond to commands to perform services for the terminal.

The operator can condition the system's response when it is unable to comply immediately with his request for a program. The system may be temporarily too busy with requests from other operators. By command, he may choose one of the following when this condition occurs:

- The system should deny his request and allow him to make some other request.
- The system should hold his request and honor it at the earliest possible moment.

Once the operator of a terminal has specified one of these system responses, the CCP handles all program requests from that terminal accordingly until the operator gives the other specification.

A File command allows the operator to specify the disk data files that are to be accessed by programs he requests, if those programs are written to accept the specification. Within the information system, there may be several files containing similar data in the same format. A school system, for example, might have a separate student records file for each school. An application program requested by the operator might have been written to access any of these files, but the program must be told which file to use on a particular run. A File command issued by the terminal operator applies to all programs requested at that terminal until a contradictory command is issued.

Another command permits the terminal operator to send a message to the system operator, requesting him to take some action.

The commands discussed so far are issued to the communications control program while it is monitoring a terminal for requests. Once a terminal is in interaction with an application program, however, the input from that terminal is meaningful only to the application program, with one exception. The CCP checks each message from a terminal to a program for the appearance of a certain string of characters, specified to be significant in your system. When it detects the presence of these characters in a message, the CCP interprets the message as an attempt by the terminal operator to escape from control of the application program and communicate directly with the CCP. At this point, the communications control program accepts a request from the operator to release the terminal from the control of the application program, or to send a message to the system operator. If the request was to send a message, the terminal operator can ask the CCP to resume execution of the application program after the message is sent.

TERMINAL MODES

There are two classes of terminals that were defined previously, based on whether or not the terminals are capable of entering commands to the CCP, **command terminals** and **data terminals** (see *Appendix E: Glossary*). Data terminals are capable only of transmitting or receiving data under control of an application program; they are not capable of commanding CCP services. When data 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 data 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 or other commands. 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 providing additional information 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, it remains in command mode 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, at that time the terminal is again placed in command mode and is able 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 escape from data mode and 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 release his terminal completely from the control of the program (at which point the terminal is again in command mode).

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 how 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 Command*).
- He can tell the CCP by what name, of a set of names defined as valid, his terminal should be known to the program he is requesting (see *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 (by using the Message command).
- Release his terminal from control of the application program (see *Release Command*).
- Resume execution of the program (see *Run Command*).

Sign-On Command (/ON)

The Sign-on command notifies the CCP that the terminal operator wishes to begin making requests of the system. If the system uses a security feature, the Sign-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 Queue or No-queue command indicates how the CCP is to handle program requests from this terminal which cannot be honored immediately:

Queue — The operator will wait for the program to start. The CCP places the request on a queue and honors it as soon as possible.

No-queue — The operator will not wait if the program cannot start immediately. The CCP rejects the command if it cannot be honored immediately and allows the operator to enter another request.

A Queue or No-queue command remains in effect until the terminal operator enters a different Queue or No-queue command or until he signs off. If neither a Queue nor a No-queue command is entered at the terminal, the CCP assumes the No-queue option.

Note: Once a terminal has a program request queued, the request cannot be removed from the queue.

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 2-1).

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.

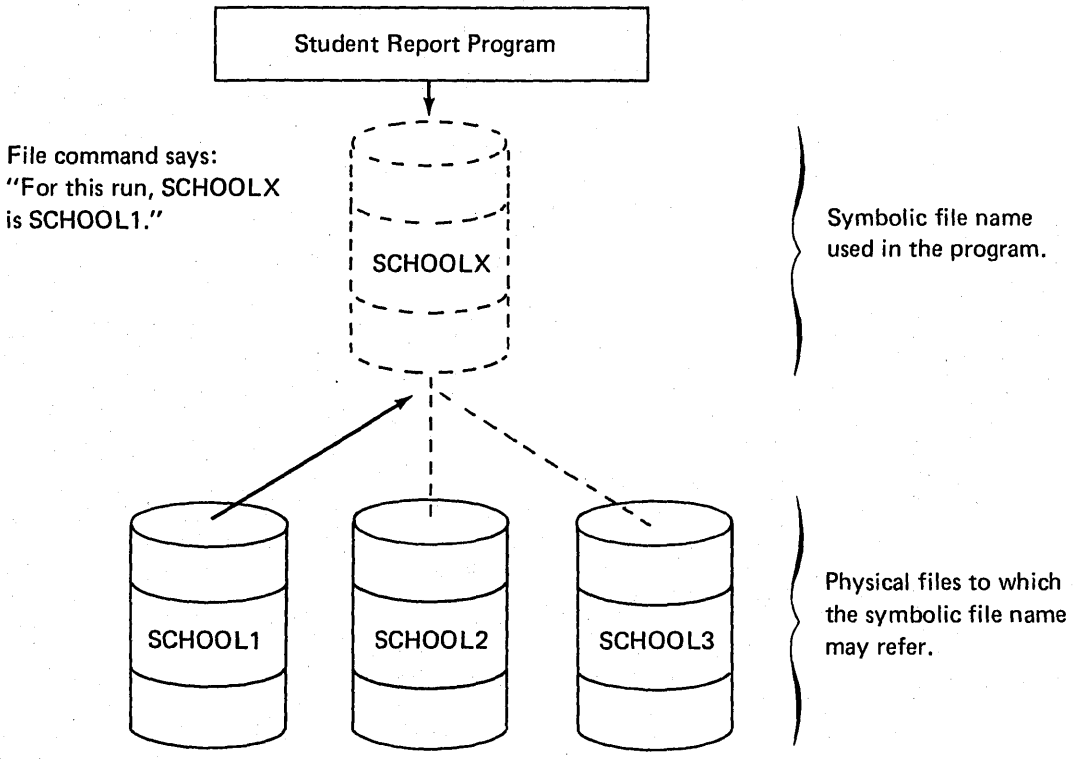


Figure 2-1. Illustration of File Command

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 this terminal requests on any operation directed to the requesting terminal (see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579).

The CCP maintains a list of symbolic names eligible for each terminal. These names are associated with the terminal during the CCP assignment and the terminal to which they apply may be changed by the system operator. The primary purpose of the Name command is to allow a terminal to assume (with the consent of the system operator) 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 two 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, input data can be entered with the program request.

Before the operator enters a program request, he should consider whether Queue or No-queue, File, and 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.

Data Mode Escape Command

The terminal operator can interrupt an application program he requested (when the terminal is in data mode, under control of the program) by entering a predetermined set of six characters at any time the program attempts to get input from the terminal. 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 requesting 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*).
- Release his terminal completely from the application program he interrupted (see *Release Command*).

Release Command (/RELEASE)

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

The Release command causes the terminal to be released from the control of the application program. The program is informed of this action at the terminal. The terminal operator is then free to enter another program request or other commands.

Message Command (/MSG)

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 and, possibly sent one or more messages to the system operator. This command causes the terminal to return to data mode from command interrupt mode.

Sign-Off Command (/OFF)

The Sign-off command causes the terminal to be returned to initial mode or to be placed offline. The Sign-off command can be accompanied by either the word HOLD or the word DROP:

HOLD – The terminal is returned to initial mode. The CCP will accept a Sign-on command or a Message command from the terminal, but no other command will be accepted.

DROP – If the terminal is connected by a switched line, the line is disconnected; the terminal can reestablish the connection only by redialing the system. If the terminal is connected by a nonswitched line, 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 entries *MLTATERM statement* and *BSCATERM statement*). All Sign-off commands are logged on the system operator's console with an indication of whether HOLD or DROP was used as the Sign-off option.

The Sign-off command may be entered any time after the Sign-on command, while the terminal is in command mode. It may not be entered following a data mode escape command unless the Release command has been issued.

The Sign-off command clears all file specification entries in effect for this terminal (see *File Command*) and restores the original name of the terminal (see *Name 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 Queue or No-queue 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?

Chapter 3. System Operator Control of the CCP

The system operator has the important role of exercising final control over the communications-based system. He initiates its activity by loading and running the communications control program. He determines when the system should refuse to accept new requests from terminal operators. While the system is in operation, he may initiate certain system actions, determine the system's status, and alter the set of terminals permitted to access the system. He must also make decisions when exception situations (such as error conditions) are detected by the communications control program or a program running under its control.

Once the CCP has been started, all communication between the control program and the system operator is through the 5471 printer/keyboard (console). Messages from the system are printed on that device, some requiring responses from the operator. His response also is keyed on that device. At any point during the execution of the communications control program, the operator may wish to command a system action; he does so by pressing the REQ key on the console, then keying his command.

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 6: Generation Stage* and *Chapter 7: Assignment Stage* for the required procedures and statements.

INITIATING THE CCP

The system operator starts the CCP by entering OCL statements at some point after a Disk System IPL (initial program load). The OCL statements cause the CCP to be loaded and supply OCL FILE 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.

SYSTEM OPERATOR COMMANDS

During operation of the communications control program, the system operator can command the CCP to:

- Disable or enable access to the system by any terminal.
- Change the terminal to which a specific terminal name should reference.
- Communicate with command terminals.
- Initiate the online test of a terminal to determine whether it is operating correctly.
- Suspend, and later resume the acceptance of terminal commands, the initiation of user programs, or the execution of one or more programs.
- Control allocation of unit record devices in a DPF system.
- Cancel the execution of an application program currently running.
- Display information about the system's status.

- Begin an orderly shutdown of the system by allowing no new program requests but permitting currently scheduled programs to complete their execution.
- Cease all activity of the system immediately and terminate execution 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 console. All commands consist of an operation code and most require one or more operands. The operation codes can be used in either their full-length or in abbreviated versions. Specific operation codes and operands are given in the *IBM System/3 Model 10 Disk System Communication Control Program System Operator's Guide,* GC21-7581.

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, is not connected, is not a command terminal, 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 application programs 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 (see index entry *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

For each file the task is using:

- Filename
- 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 or program name when attempting 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.
- Not allow additional commands from command 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 all commands from command terminals were previously suspended, allow command terminals to resume entering commands.

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 rejected when:

- The symbolic terminal name is currently being used by a program running under the CCP.
- The symbolic terminal name is currently the only name assigned to an online terminal which is in command or initial mode or has a queued program request.

- The name is currently assigned to a different type of device.

- 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 under which a command terminal operates the terminal operator must also enter a command (see index entry *Name Command*).

Start and Stop Online Terminal Test

The Test 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 Test command with a STOP operand stops a looping 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.

An 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. However, if an error occurs, an error message will be printed on the console. Certain terminals may have restrictions as to what online tests can be executed from the central system or requested by the terminal operator.

Allocation of Unit Record Equipment

The Allocate command allows the system operator, on a system equipped with the Dual Programming feature, to control the program level to which each unit record device is allocated. In this way the system operator can control individually whether the printer and card device is available to the CCP application program or to the program in the other level.

USING THE SYSTEM OPERATOR'S CONSOLE AS A REQUESTING TERMINAL

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

- Queue

- No-queue

- 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 the program begins execution, 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 Shutdown command. 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; but any queued program requests are executed, and each of these programs is informed of the pending shutdown condition.

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

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*).

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 (see *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581.

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 reference ID 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.

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 is 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.

Chapter 4. CCP Services for the Application Programmer

The communications control program aids the programmer in two primary ways:

- By relieving him of many programming concerns inherent in an event-driven system, it lets him concentrate on application programs that do the processing he requires.
- It permits him to write application programs that include communications input/output in a high-level language.

Programs that run under the communication control program can be written in any of four languages:

- RPG II
- COBOL
- FORTRAN IV
- Basic Assembler

By writing in RPG II, COBOL, or FORTRAN, the programmer can avoid the strict rules required when using Basic Assembler Language.

In whatever language he writes, the programmer can ignore the problems that arise from his program contending with others for system resources. Those problems are managed by the CCP. The programmer is assured that all required resources are available to his program each time it is executed. If necessary, the CCP defers the execution of his program until those resources are available. If his program shares access to a disk data file with another concurrently executing program, the CCP manages the contention and does not permit the two programs to cause an erroneous record update through conflicting reads and writes.

Programs that execute under the control of the communications control program are written much the same as programs in the same language for a system without telecommunications; that is, the statements used to process data and the handling of data files are identical. The standard disk data management methods are supported by the control program. Only two elements are likely to differ significantly:

- The overall logic of the program.
- The means of communicating with terminals or with the system operator.

Except for RPG II, the high-level languages do not offer any statements for accomplishing terminal input/output. Terminals cannot be treated as data files. Furthermore, the MLTA IOCS and BSCA IOCS do not permit access to their facilities directly from a high-level programming language. The communications control program incorporates MLTA IOCS or BSCA IOCS (or both, if you have both line types in your system) and offers the application programmer a method of using these IOCS facilities to interact with terminals.

Since there are no statements in the languages to specify terminal actions, the application program indicates those actions to the communications control program by means of one of the following statements:

- In COBOL or FORTRAN – CALL statement
- In RPG II – EXIT statement, or use of a SPECIAL file
- In Basic Assembler – a supplied macro instruction

Each of these statements is accompanied by parameters to indicate the specifics for the operation. See *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

Most communication by programs running under the CCP is with the requesting terminal. But the ability also exists to address other terminals. A terminal is addressed in the program by a name that is chosen by the user; the name normally applies to one particular terminal. However, should a certain terminal become unavailable during a run of the communications control program, the system operator can reassign the name to another terminal. Any program addressing a terminal by the reassigned name addresses the new terminal. For the most part, the application program need not be concerned with the type of terminal with which it is communicating.

An application program communicates with the system operator by addressing the 5471 printer/keyboard (console) as another terminal. A certain constant name is always used to address the console. The only operations available here are:

- Write a message.
- Write a message and wait for a reply.

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.

The CCP performs four types of control program services:

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

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 (maximum eight) 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
- 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.
- Line scheduling.
- Data code translation.
- Terminal testing.
- Other services which allow the application programmer to be largely 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 *BSCALINE statement* and *MLTALINE statement*). This list gives the order in which the terminals on a line are to be polled (interrogated) for data and commands when none of the terminals on the line are busy.

Selection is the specific addressing of a terminal by communications management in order to transmit output data to the terminal.

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.

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 by which a command terminal operates, 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).

Unit Record I/O Requests: During CCP operation, the CCP file management function intercepts all requests from user application programs for unit record I/O operations. This function is performed by a CCP service subroutine, link-edited with the user program. 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, deallocating of system resources, purging programs from the system, and maintaining a record of the requests for 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 requesting terminal programs, however, are honored before other requests.

Note: Once a terminal has a program request queued, the request cannot be removed from the queue.

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 are available to it.

Requests for program which can support multiple requesting terminals 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 Request 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*).

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 operations.

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, 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 shutdown, 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:

The CCP incorporates the appropriate MLMP 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.

Chapter 5. Designing Your Communications-Based System

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 if you can devise an 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 processed 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 remain 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 *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

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.

ESTABLISHING THE SYSTEM

The CCP is designed to suit diverse data processing environments involving online applications. Telecommunications line and terminal configurations, and the configurations of central processors that host them, will vary greatly among the users of the CCP. Different users will have different requirements in functional and performance characteristics of their telecommunications subsystem. The CCP, as distributed, consists of a set of code which can be tailored to the needs of the user.

The CCP is established in the following stages:

- Generation
- Assignment
- Operational System Startup

The generation of the CCP by the user is the first stage in tailoring the distributed code to his needs. This stage requires the processing of a number of generation control statements, a series of linkage edits, and a set of disk utility operations. During this stage a set of the CCP system code is created which defines the functions that this version of the CCP can perform.

Once generated, the CCP is still not bound to a specific set of user programs, data files, or terminal assignments. The set of programs, data files, and terminals of any CCP system will probably vary throughout the life of that version of the CCP, even as the required functional abilities remain constant. Therefore a procedure separate from generation is provided for the establishment and modification of these definitions. Like generation, the assignment stage is performed as a normal System/3 operation: its specific function is the creation, modification, or replacement of a set of control tables used by the CCP to manipulate user programs, files, and terminal resources. But unlike the generation process, which tends to be lengthy, the assignment stage is rather brief, requiring only the reading, interpretation, and encoding of straightforward user specifications, and the writing of these to a disk data file.

The assignment stage identifies programs, terminals, disk files, unit record devices, and symbolic terminal names to be used in a particular execution set for the CCP. As with generation, the programs necessary to perform the assignment stage are supplied as part of the distributed CCP modules. The user may run the assignment stage many times in the development of his telecommunications applications. For example, he may have just developed a new application program which he now wishes to incorporate under the CCP. He may also have different assignment sets that can be used for different runs of the execution of the CCP.

After he has performed the generation and assignment stages, the user is ready to operate his CCP system. The system operator now loads the CCP, specifying the assignment set the system will execute under. Certain additional elements for only the current run of the CCP can be specified during the operational system startup.

OPERATING THE SYSTEM

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 during 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.

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:

- Executing programs are notified to go to end-of-job
- Communication lines and adapters are closed
- Disk file DTFs in main storage are restored
- Disk data files are closed
- If the program request 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
- The CCP sends a closing message to the system operator
- The CCP exits to the system end-of-job routine

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. For detailed storage estimates for the CCP, see *Appendix F. Storage Estimates.*
- 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?

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

DUAL PROGRAMMING FEATURE (DPF) CONSIDERATIONS

The CCP does not require the dual programming feature; but it does not prevent its use during the execution of the CCP.

The CCP itself can only operate in a single program level partition at any time. Because certain batch functions are available in program level 1 only, it is recommended that the CCP be run in program level 2. Assuming main storage is available, and subject to the standard constraints imposed by Disk System Management with regard to DPF, user programs not controlled by the CCP can operate in the other program level concurrent with the execution of the CCP.

If DPF is to be used while the CCP is in operation, it is strongly recommended that the disk configuration of the system include more than a single drive, in order to reduce degradation resulting from contention for disk access.

A program running in the non-CCP level can use a teleprocessing adapter (MLTA or BSCA) concurrently with the execution of the CCP. This adapter must not be supported by the particular assignment set currently in use for the CCP. The programming support for the non-CCP adapter can be:

- RPG II Telecommunications Feature
- Multiline/Multipoint (MLMP) Feature
- Multiple Line Terminal Adapter (MLTA) Feature

For the description of the valid uses of disk files, unit record devices, and the console in DPF under the CCP, see *Disk System Management (DSM) Considerations.*

SERIAL SIO CHANNEL (SIOC) CONSIDERATIONS

The SIOC should not be used when using MLTA terminals under the CCP. MLTA interrupts cannot be processed fast enough via the MLTA IOCS if SIOC interrupts are occurring concurrently.

DISK SYSTEM MANAGEMENT (DSM) CONSIDERATIONS

The CCP operates in conjunction with DSM and uses DSM facilities whenever possible, including the I/O Supervisor for disk and unit record I/O devices.

Certain constraints are placed on the existing DSM programming support.

1. Special considerations must be given to a program not written to be run under the CCP. In order to run under the CCP, the following considerations apply:

   - The program must be defined to the CCP within an assignment set.

   - If using unit record devices the program must be re-link edited to include the CCP intermediate data management modules for the MFCU, 1442, 5203, or 1403.

     The CCP allows the unit record devices to be used by the non-CCP level in a DPF system. The system operator can make the device available to the other level if the device is not currently in use or there are no program requests in allocation which require it.

     The CCP allocation insures that a unit record device is allocated to a program prior to initiation.

     When the user program allocates and opens a unit record device, the CCP processes the request. Necessary buffers and IOBs are built, input data priming occurs, the address of the unit record error recovery routines are set into the DTF, and the DTF is marked as belonging to the specific user program.

     When the device DTF is to be closed, the CCP insures that any pending output is completed. The device remains associated with the program until the program terminates.

   - The program must not use the console through normal device or file definition. Communication must only be done through a CCP communications operation or by an output only Halt/Syslog request.

   - The program must not use telecommunication devices except through CCP operations.

   - The program must not use multivolume disk files.

   - If the program is requested from a terminal, that terminal is allocated to the program until released by the program.

   - System program with names beginning with $ are not permitted to run under the CCP.

   - The program must not issue unconditional halts, either stand alone or through Halt/Syslog.

2. The CCP cannot be loaded while the other level is active on a DPF system. After the operational stage of the CCP is running, the other level is available to load and execute programs.

3. The OCL input for the non-CCP level on a DPF system must be from a card read device, not from the console. Procedures on disk are permitted for starting the CCP or for use by the non-CCP level.

4. The CCP level logs all messages to the console regardless of where the system log device has been assigned. In a DPF system, the non-CCP level cannot log to the console. Therefore, if the system log is assigned to the console during startup, the CCP automatically turns the log off. If the system log is not assigned or is assigned to the printer, the CCP will take no actions as regards the system log assignment.

5. The INTERRUPT button on a DPF system must not be used for the CCP level. It is the user's responsibility to avoid using this button in the CCP level.

6. The following considerations apply to the use of disk files under the CCP:

   - All disk files to be processed during a CCP run must be online and described via the // FILE OCL statements following the // LOAD $CCP,xx statement.

   - All disk files are actually opened at startup of the CCP and closed at the CCP shutdown. Because of this, no index file key sort occurs until CCP shutdown. Programs wishing to access added records to an index file after the adding program has successfully reached EOJ, must have included additional data management.

   - The rules for sharing disk files between program levels of a DPF system are:

CCP Program Level	Other Program Level
Sequential or indexed file*	Cannot process file
Direct file input	Can retrieve and update
Direct file update	Can retrieve from file only

   * The CCP treats all sequential and indexed files as 'add' files.

   - Once an index sequential add or index file load is done by a program running under the CCP, that file cannot be accessed during that CCP run.

   - Creation of direct disk files is not permitted by programs running under the CCP.

   - The CCP permits concurrently running programs to update the same file, by protecting the block of sectors containing the record, until the program releases the block of sectors. The block is released when the program reads another block. Users wishing to use this capability must be aware of a possible lockout condition if the opening program neither releases the block nor goes to EOJ.

   - Multivolume disk data files are not supported by the CCP.

7. The following considerations apply to the use of unit record devices under the CCP:

   - A unit record device cannot be shared by concurrent programs running under the CCP. Once a program terminates, the device is available to another program running under the CCP (or, in DPF, in the other level).

   - If unit record devices are supported by the CCP, the device is allocated to either the CCP level or the non-CCP level while the appropriate program is executing.

8. The following considerations apply to use of the console:

   - All programs running under the CCP must issue operations to the console as CCP communications operations or via Halt/Syslog. The console is not supported by the CCP as a file.

   - Programs running in the non-CCP level on a DPF system must not use the console.

   - The CCP use of the console prohibits all but stand-alone halts from being displayed.

   - When a program has been requested from the console, the system operator cannot request another program or enter any terminal-type commands until that program has begun executing.

   - All console messages are identified by a prefix indicating to the issuer of the message (task ID) whether or not a response is required, and a message type number. All responses to these messages are given through the console, using a task ID to identify the response.

9. Checkpoint/Restart is not supported under the CCP. On a DPF system, Checkpoint/Restart can be used in the non-CCP level.

10. Inquiry (Rollout/Rollin) is not supported under the CCP or in the other level.

UNIT RECORD I/O CONTROL CONSIDERATIONS

The unit record control routines of the CCP prevent the CCP program level from being idle while waiting on a non-ready unit record device (I/O attention light for CCP level).

The programming required to control unit record I/O under the CCP is contained in two places:

1. The CCP resident control program incorporates the general control logic.

2. Additional, and specific device-oriented logic is incorporated into the user program module. This logic is included via the linkage edit process at the direction of the user when building his program.

Any user program unit record I/O request first passes through the specific device logic in the user program module. This routine determines if the device is ready. If it is, the routine branches to the DSM unit record data management module to complete the request. However, if the device is not ready, the link edited CCP routine does not allow the operation to be initiated at this time. Instead it branches to the CCP control routines to place that program into a wait for the device. When there are no other programs in the CCP system ready to run, the CCP control routines give control back to the module to synchronously test the device status again. Once the device is found to be ready, the operation is allowed to take place.

The first time that a device is found to be not ready for an operation when called upon by the user, a CCP message will be written to the system operator to inform him that the specific device is not ready and that some action needs to be taken.

If, in a DPF system, the non-CCP level causes an I/O attention to a unit record device, the CCP level is unaffected and can continue to run.

3270 DISPLAY FORMAT FACILITY (DFF) CONSIDERATIONS

The 3270 Display Format Facility (DFF) is a facility of the CCP that is selected separately during the CCP generation. The purpose of the facility is to allow programs written in RPG II, COBOL, FORTRAN IV, and Basic Assembler to control the display format for the 3270 Information Display System. The DFF makes it possible to control the display format and perform operations involving data fields in the display directly from the application program in a manner similar to performing operations with any other terminal supported under the CCP.

The DFF is composed of the Display Format Generator routine $CCPDF (DFGR) and the Display Format Control routine (DFCR). The DFGR, which is executed prior to the CCP startup, processes special DFF specifications, builds display formats, and stores the display formats in an object library. The DFCR processes requests for DFF services issued by application programs running under the CCP.

The DFGR operates in either program level but not concurrently in both program levels. The main storage required for execution is always 14K. When the other level is using the same system input device, the DFGR cannot be initiated until that device is available for use. Both levels may use Halt/Syslog at the same time. Logged information from both levels is interspersed. The DFGR cannot place new display formats in a library if the other DPF level is using temporary entries or doing library functions.

The DFGR operates under control of the System/3 Model 10 Disk System Management. For additional discussion of the DFGR and DFCR, see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

Components within the 3270 System that are supported are as follows:

Component	Environment	Model Numbers
IBM 3271	For remote applications	Model 1 and Model 2
IBM 3277	Attached to the 3271 Control Unit	Model 1 and Model 2
IBM 3284	Attached to the 3271 Control Unit	Model 1 and Model 2
IBM 3286	Attached to the 3271 Control Unit	Model 1 and Model 2
IBM 3275	Stand Alone display station	Model 1 and Model 2
IBM 3284	Attached to the 3275 Display Station	Model 3

Special features of the 3270 System supported are: the selector pen, the audible alarm, and the operator identification card reader.

The 3270 is supported as a remote attachment to the System/3 Model 10. Communications between the System/3 and the 3270 are maintained using the Binary Synchronous Communications multipoint data link mode of operation. All operations that can be performed with the 3270 in the remote operation are supported except for the read-type and general poll commands. Polling sequences are used for remote read operations.

TERMINAL SECURITY CONSIDERATIONS

If you have no terminal security feature built into your system, each command terminal can issue program requests and other commands to the system after the Sign-on command has been entered at that terminal.

If you are concerned with security of access to the system, you can include a password feature in the CCP at generation (see index entries *$ESEC — terminal sign-on security* and *SYSTEM statement*).

In a system with password protection, no requests are accepted from a terminal until its operator presents the current password and the CCP verifies it. Once an operator signs on with the password, he can make any number of requests without repeating it. If the operator is not always at his terminal, he can sign off the terminal anytime he leaves it. Once he has signed off, the CCP requires any further use of the terminal to be accompanied by the password. Thus if other, possibly unauthorized, persons gain access to that terminal, they cannot gain access to the system.

The valid password for the current run is established in an assignment set, but can be changed by the system operator at startup of the CCP (see *Startup Procedure* in the *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581). The password can change at startup on every run, or can be retained for days or even weeks. In any case, only terminal operators who know the password are permitted access to the system.

If the user wishes to write his own terminal sign-on security routines rather than use the CCP password facility, there are stringent requirements his routines must observe with regard to the interface with the CCP (see index entry *$ESEC — terminal sign-on security*).

All sign-on attempts are logged on the console.

Chapter 6. Generation Stage

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 physical size (main storage space required) and its maximum capability. Further selection is done at assign

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 select the 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 and initializes an assignment file ($CCPFILE), whose contents – the specifications of an actual terminal configuration, disk files to be accessed, and programs to be used – are filled in at assignment time.

PROCEDURE FOR GENERATION

The key step for the user in the generation procedure is where he describes the kind of CCP he wants by modifying the sample generation statements provided to him. In order to modify the statements, the user must know certain facts about his system and about the capabilities of the CCP he wants to generate. These facts are determined during the CCP system design. Design considerations are given in *Chapter 5. Designing Your Communication-Based System,* earlier in this publication. The following is a checklist of the facts a user must know before performing generation:

Generation Checklist

_____ Type of card device attached to the system (MFCU and/or 1442)

_____ Printer configuration of the system (5203 and 1403)

_____ Disk configuration of the system (5444 and 5445)

_____ Maximum number of concurrently executing user programs

_____ Whether or not the system has DPF

_____ Whether or not the data mode escape feature is to be used, and if so the six user-specified data mode escape characters

_____ Is a program request count to be kept

_____ Is disk file sharing to be allowed

_____ Will any programs to be run under the CCP use the symbolic file reference facility or will all programs be written to reference specific files

_____ Is Display Format Facility (DFF) to be supported

_____ Programming language(s) to be supported by the CCP

_____ Which 5444 units you will use to mount the packs during generation which will later be used for preparing programs to be run by the CCP

_____ What type of sign-on security will be used, if any

_____ Length of your security comparison information, if you use your own sign-on security checking routine

_____ Anticipated number of assignment sets to be placed into $CCPFILE

_____ Maximum number of programs and files in an assignment set

_____ Maximum number of terminals planned

_____ Space to be reserved in $CCPFILE for dynamic main storage dumps

_____ Main storage size of processing unit

_____ Number of tracks to be reserved in $CCPFILE for CCP trace entries

_____ Disk unit and pack name on which the pack to contain $CCPFILE will be mounted during generation

_____ Beginning track location for $CCPFILE

_____ Number of MLTA lines to be supported

_____ Whether or not MLTA input and output will always be translated to and from EBCDIC

_____ MLTA terminal devices to be supported

_____ MLTA line transmission codes

_____ BSCA lines, line features, and BSC control logic to be included in the CCP support

_____ BSCA line transmission codes

_____ BSCA terminal devices to be supported

_____ Disk unit on which disk system management resides (F1 or R1)

_____ Disk unit onto which the CCP will be generated

_____ Disk unit(s) and pack name(s) where work file space can be found during generation

_____ Disk unit on which the distribution CCP modules reside

Generation is the key step toward obtaining a usable communications control program, because the user includes those features which the system will make use of and excludes features which are not needed, thus creating the smallest possible CCP which contains all the facilities required. The user should carefully check each generation option and specification to determine if his system will require it. Some features will be required for his system, some will be completely unnecessary, and others may be simply desirable. It should be kept in mind that most features chosen increase the size of the control program, lessening that portion of storage in which application programs can be executed (see *Appendix F. Storage Estimates*).

The basic procedure for the CCP generation is:

1. A sample control statement deck is punched from the source library of the distribution pack.

2. The sample deck is modified by the user to his specifications and entered as input to the next step of generation.

3. A full job stream to accomplish the necessary functions is punched for the user.

4. The job stream is used to generate your version of the CCP and the CCP assignment file ($CCPFILE) is ready for the user's initial assignment run.

The function of the generation stage of the CCP is to:

- Generate modules that require modifying at the source level.

- Link edit the generated modules and certain other relocatable modules to create a load module that will be the resident control program during the CCP operations, and a load module that will initialize $CCPFILE.

- Copy these and other load modules to the designated CCP production pack. The production pack is the pack from which the CCP is loaded for execution. The CCP production pack can be any pack other than the distribution pack. It may or may not be a DSM system pack and might be the current system pack during the CCP generation.

- Copy additional selected relocatable modules to the 'program preparation' packs (packs that will be used for compilations and linkage edits of user written application programs that will execute under the CCP).

- Allocate and initialize (but not fully enter information into) the CCP assignment file ($CCPFILE) on the designated pack.

The generation stage assumes:

- That the DSM is properly generated on the system pack, including the appropriate MLTA and/or BSCA I/O macros and subroutines.

  MLTA — The MLTA microcode deck (obtained from IBM Field Engineering) must be loaded into the object library under the name $MLMC1 on the system pack.

  The MLTA error statistics file (MLTERFIL) must be created and initialized on F1. To initialize MLTERFIL, the MLTA feature provides, in the object library, module $$MLFI. The OCL statements required to initialize MLTERFIL are:

```
// LOAD $$MLFI,XX
// FILE NAME-MLTERFIL,UNIT-F1,
//      RETAIN-P,RECORDS-9,
//      PACK-pppppp,LOCATION-nnn
// RUN
```

  BSCA — The BSCA must provide a file on F1 for logging control station terminal statistics. To initialize MLTERFIL, the MLMP (Multiline/Multipoint) feature provides (in the object library) module $$BSFI. The OCL statements required to initialize MLTERFIL are:

```
// LOAD $$BSFI,XX
// FILE NAME-MLTERFIL,UNIT-F1,
//      PACK-pppppp,TRACKS-1,
//      LOCATION-nnn,RETAIN-P
// RUN
```

  Note: MLTERFIL need be initialized only once to accommodate both BSCA and MLTA statistics. Part of MLTERFIL comprises the BSCA terminal log area and is used for logging the control station terminal statistics. Another part of MLTERFIL is used for logging MLTA statistics if MLTA is present. Do not initialize the file twice if you use both BSCA and MLTA.

- That during generation a copy of the Macro Processor ($MPXDV and all its subsequent load modules) and the Overlay Linkage Editor exist on the system pack used during the CCP generation.

- That an appropriate sized object library and source library have been allocated on the production pack (5444 only).

- That the object library on the production pack has been reorganized and no modules have been deleted since reorganization. All the CCP modules should be contiguous on the production pack after the CCP generation. (The CCP has no control over where DSM may place modules if modules have been deleted.)

- That on any program preparation pack there already exist the unit record data management routines that will be used by programs compiled using that pack.

Notes:

1. **The printed output resulting from generation must be saved in case of required maintenance by IBM Field Engineering personnel. This paper is the only documentation of the user's unique system and the precise sequence of events during this particular CCP generation.**

2. The user should consider back-up procedures (of his own design) in case the CCP distribution pack or the one or more packs generated are inadvertently destroyed.

3. Generation can be accomplished on a System/3 other than the system which will use the CCP in teleprocessing.

Operational Procedures For Generation

The CCP is distributed on a pack separate from the distribution of the other components of the System/3 Disk System Management. If you are generating both the basic DSM and the CCP, you must generate the basic DSM first, following the procedures described in *IBM System/3 Model 10 Disk System Operator's Guide,* GC21-7508.

To generate the CCP, you must mount the CCP distribution pack on a unit separate from the system pack. You must have performed an IPL (initial program load) from the system pack at some previous point. Unlike the generation of DSM, you may direct the output from the CCP generation to more than one pack. The output consists of:

- Those modules required for performing the assignment stage and for executing the operational CCP; these modules are directed to the CCP production pack.

- The subroutines to be used on compiling and link editing application programs to be run under the CCP (macros in the case of Basic Assembler Language programming); these subroutines are directed to one or more program preparation packs.

Each of the above might be a different pack, or could be the same pack. You specify the disk unit on which each of these packs is mounted.

The CCP generation procedure is dependent on certain modules which must be present on the system pack from which you performed the IPL:

- The Macro Processor

- The Overlay Linkage Editor

- The macros and subroutines for BSCA and/or MLTA, as appropriate for the terminal devices to be supported.

Figure 6-1 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.

Generating the CCP is divided into six steps:

Step 1 (User):

The user enters from the system input device (whether that be the console or the card device) statements of the following form:

```
// &
// LOAD $MAINT,dsunit
// RUN
// COPY FROM-diunit,TO-PRTPCH,
//      LIBRARY-S,NAME-$CGSMP
// END
```

where *dsunit* on the // LOAD statement is the unit on which the DSM system pack resides. The *diunit* in the // COPY statement is the unit on which the CCP distribution pack is mounted.

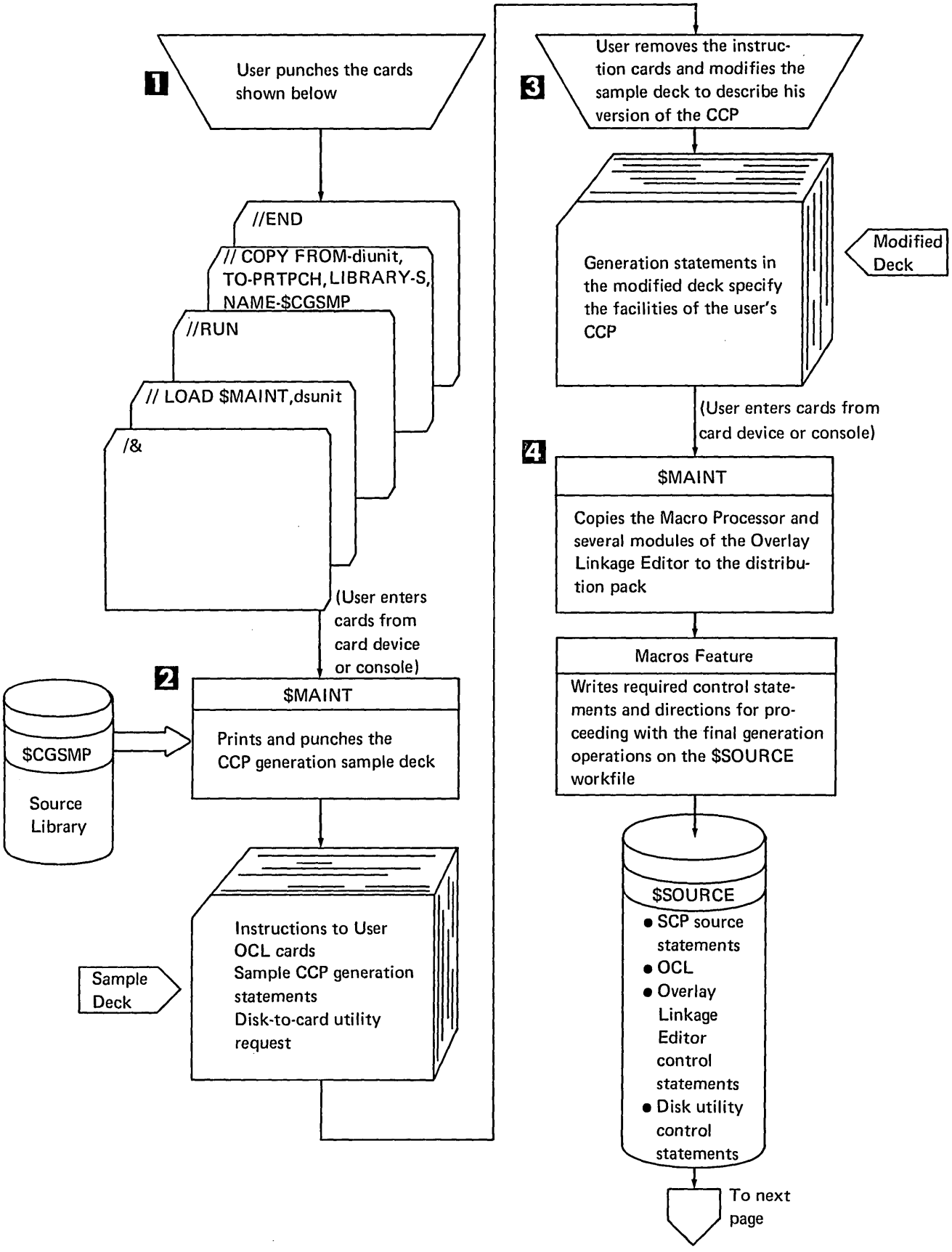


Figure 6-1 (Part 1 of 3). CCP Generation Procedure

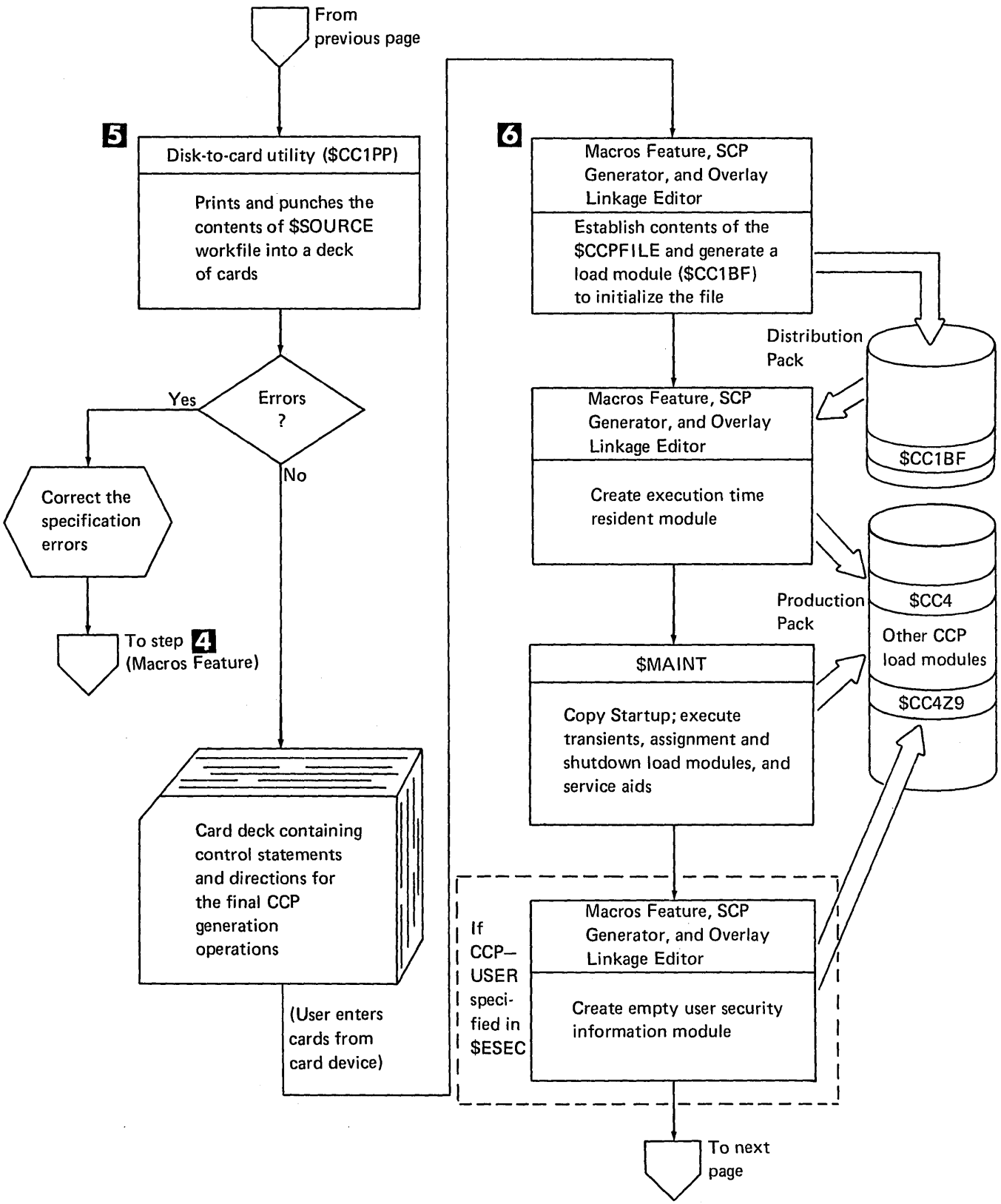


Figure 6-1 (Part 2 of 3). CCP Generation Procedure

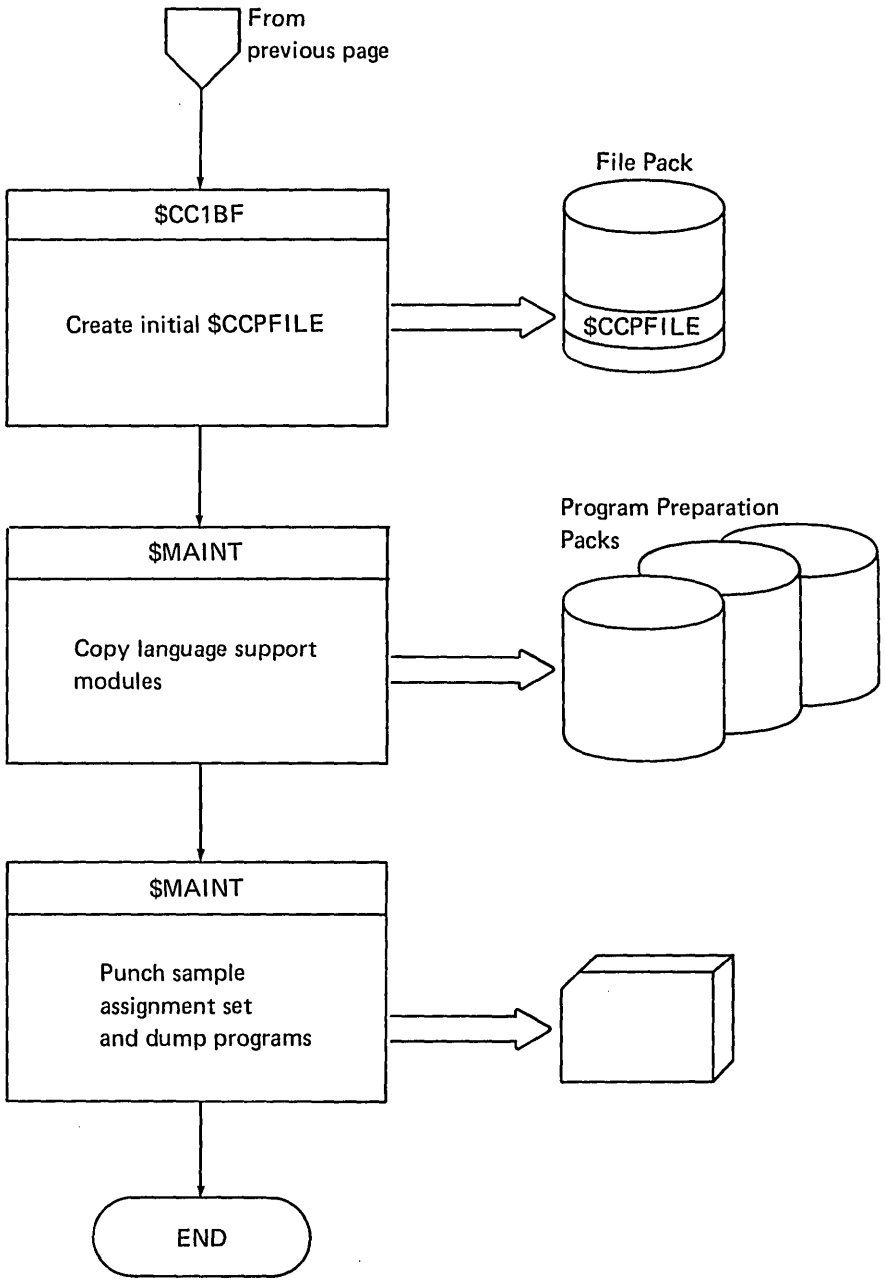


Figure 6-1 (Part 3 of 3).. CCP Generation Procedure

Step 2 (System):

The system retrieves from the source library on the CCP distribution pack the member named $CGSMP (CCP sample generation deck), prints the entire member on the printer, and punches the entire member into cards.

There are four sections to the information printed and punched (in the following order):

- Instructions to the user for modifying the punched deck.
- OCL and COPY control statements to copy the Macro Processor and parts of the Overlay Linkage Editor to the distribution pack.
- OCL to call the Macro Processor, followed by sample $E generation control statements to generate a CCP system. One card is provided for each parameter of a generation control statement. The card generally contains the default value, and provides a comment indicating alternative values for the parameter.
- OCL to call the CCP disk-to-print/punch utility program ($CC1PP).

The following is an example of the printout that accompanies the sample generation deck:

```
*****************************************************************************
*              I N P U T   T O   C C P   G E N E R A T I O N                *
*****************************************************************************
*                                                                           *
*                                                                           *
* THESE CARDS -- AFTER YOU HAVE MODIFIED THEM TO YOUR SPECIFICATIONS --     *
*   WILL BE THE INPUT TO THE FIRST PASS OF CCP GENERATION                   *
*                                                                           *
*                                                                           *
* DISCARD THE LEADING CARDS OF THIS DECK, FROM THE // COPY CARD AT THE      *
*   BEGINNING THROUGH THE CARD THAT MARKS *** END OF INSTRUCTIONS ***       *
*                                                                           *
* DISCARD ALSO THE // CEND CARD -- THE LAST CARD OF THE DECK                *
*                                                                           *
*                                                                           *
* YOU MUST MAKE THREE KINDS OF MODIFICATIONS TO THE REMAINING CARDS --      *
*                                                                           *
*   1. IN THOSE CARDS THAT ARE MARKED  ++  IN COLUMNS 74-75, REPLACE        *
*      ANY  $$  OR  ##  IN THE CARD WITH THE IDENTIFICATION OF A DISK       *
*      UNIT --                                                              *
*                                                                           *
*      --  $$  WITH THE UNIT ON WHICH YOUR SYSTEM PACK IS LOCATED           *
*                                                                           *
*      --  ##  WITH THE UNIT ON WHICH THE CCP DISTRIBUTION PACK IS          *
*              LOCATED                                                      *
*                                                                           *
*   2. REPLACE CARD NUMBER 00299 WITH A CARD PUNCHED  /*  IN COLUMNS 1-2    *
*                                                                           *
*   3. MODIFY THE GENERATION CONTROL STATEMENTS -- THOSE CARDS              *
*      FOLLOWING CARD NUMBER 00205 -- TO SPECIFY THE REQUIREMENTS OF        *
*      THE CCP YOU WISH TO GENERATE                                         *
*                                                                           *
* IF YOU DO NOT WISH TO USE THE DISTRIBUTION PACK FOR THE REQUIRED          *
*   $SOURCE FILE, SPECIFY THE LOCATION OF THAT FILE BY CHANGING THE UNIT    *
*   AND PACK PARAMETERS IN CARDS 00204 AND 00306                            *
*                                                                           *
*                                                                           *
* WHEN YOU HAVE MADE THESE CHANGES, PLACE THE MODIFIED DECK IN THE          *
*   HOPPER OF THE SYSTEM INPUT DEVICE AND BEGIN PASS 1 OF CCP GENERATION    *
*                                                                           *
****** END OF INSTRUCTIONS ******************** END OF INSTRUCTIONS ******
*                                                                            00100
***  COPY MACRO PROCESSOR AND REQUIRED MODULES OF OVERLAY LINKAGE            00101
***  EDITOR FROM SYSTEM PACK TO DISTRIBUTION PACK                            00102
*                                                                            00103
// LOAD  $MAINT,$$                                                        ++00104
// RUN                                                                     00105
// COPY  FROM-$$,TO-##,LIBRARY-O,RETAIN-R,NAME-$MPX.ALL                   ++00106
// COPY  FROM-$$,TO-##,LIBRARY-O,RETAIN-R,NAME-$OLYNX                     ++00107
// COPY  FROM-$$,TO-##,LIBRARY-O,RETAIN-R,NAME-$OLBO                      ++00108
// END                                                                     00109
*                                                                           00200
***  PROCESS SPECIFICATIONS FOR THE CCP TO BE GENERATED                     00201
*                                                                           00202
// LOAD  $MPXDV,##                                                        ++00203
// FILE  NAME-$SOURCE,RETAIN-T,UNIT-##,PACK-PID002,TRACKS-50              ++00204
// RUN                                                                     00205
       $EIOD CARD-NO,            -- MFCU / 1442 / 'MFCU,1442' --      X   IOD00
             PRINTR-NO,          -- 5203 / 1403 --                    X   IOD01
             DISKS-NO,           -- R2 / 'R2,F2' --                   X   IOD02
```

```
        D5445-NO            -- D1 / 'D1,D2' --                          IOD03
$EFAC   MAXEUP-1,           -- 2 / 3 / 4 / 5 / 6 / 7 / 8 --         X   FAC00
        DPF-NO,             -- YES --                               X   FAC01
        ESCAPE-NO,          -- 'CCCCCC' / X'XXXXXXXXXXX' --         X   FAC02
        PGMCNT-NO,          -- YES --                               X   FAC03
        FSHARE-NO,          -- YES --                               X   FAC04
        SYMFIL-NO,          -- YES --                               X   FAC05
        FORMAT-NO           -- YES --                                   FAC06
$EPLG   LANG-     ,         -- COBOL / RPGII / FORTRAN / ASSEM --   X   PLG00
        PPUNIT-             -- R1 / F1 / R2 / F2 --                     PLG01
$ESEC   SECURE-NO,          -- CCP / USER --                        X   SEC00
        LUSI-0              -- 1 - 4096 IF SECURE-USER --               SEC01
$EFIL   SETS-1,             -- 2 - 25 --                            X   FIL00
        PROGS-10,           -- 1 - 255 --                           X   FIL01
        DFILES-5,           -- 1 - 50 --                            X   FIL02
        TERMS-1,            -- 2 - 254 --                           X   FIL03
        DUMPS-1,            -- 2 - 9 --                             X   FIL04
        CORE-24K,           -- 32K / 48K / 64K --                   X   FIL05
        TRACE-1,            -- 2 - 20 --                            X   FIL06
        FLUNIT-  ,          -- R1 / F1 / R2 / F2 --                 X   FIL07
        TRKLOC-   ,         -- VALID TRACK NUMBER / OMITTED --      X   FIL08
        FLPACK-             -- NAME OF PACK --                          FIL09
$EMLA   LINES-              -- 0 - 8 --                             X   MLA00
        XLATE-YES           -- NO --                                    MLA01
$EMLD   TYPE-     ,         -- SEE SYSTEM REFERENCE MANUAL --       X   MLD00
        XMCODE-             -- SEE SYSTEM REFERENCE MANUAL --           MLD01
$EBSC   BSCA- ,             -- 0 - 2 --                             X   BSC00
        DIAL-NO,            -- YES --                               X   BSC01
        PP-NO,              -- YES --                               X   BSC02
        MP-NO,              -- YES --                               X   BSC03
        CS-NO,              -- YES --                               X   BSC04
        GETMSG-NO,          -- YES --                               X   BSC05
        ITB-NO,             -- YES --                               X   BSC06
        RECSEP-1E,          -- TWO HEX DIGITS --                    X   BSC07
        ASCII-NO,           -- YES --                               X   BSC08
        EBCDIC-YES,         -- NO --                                X   BSC09
        XPRNCY-NO,          -- YES --                               X   BSC10
        RESPOL-NO,          -- YES --                               X   BSC11
        AUTORS-NO           -- YES --                                   BSC12
$EBSD   TYPE-               -- SEE SYSTEM REFERENCE MANUAL --           BSD00
$EGEN   DSUNIT-$$,          -- R1 / F1 --                           X ++GEN00
        CCUNIT-  ,          -- R1 / F1 / R2 / F2 --                 X   GEN01
        WKUNIT-  ,          -- UNIT / 'UNIT,UNIT,UNIT' --           X   GEN02
        WKPACK-     ,       -- PACK / 'PACK,PACK,PACK' --           X   GEN03
        DIUNIT-##,          -- R1 / F1 / R2 / F2 --                 X ++GEN04
        MINRES-NO           -- YES --                                   GEN05
*/*    -- REPLACE THIS CARD WITH /* IN COLUMNS 1-2 --                    00299
*                                                                       00300
***  PRINT RESULTS OF CCP GENERATION PASS 1                             00301
*                                                                       00302
***  IF NO ERRORS, PUNCH INPUT TO CCP GENERATION PASS 2                 00303
*                                                                       00304
// LOAD $CCIPP,##                                                     ++00305
// FILE NAME-$SOURCE,RETAIN-S,UNIT-##,PACK-PID002                     ++00306
// RUN                                                                  00307
```

Step 3 (User):

The user modifies the sample deck to reflect the requirements of his system. Specifically the user can modify:

- The *unit* on the LOAD $MPXDV,unit card to indicate the unit on which the CCP distribution pack resides.
- The *unit* and *pack* on the // FILE card of the Macro Processor OCL to indicate the unit and pack on which the work file $SOURCE should be allocated during step 4.
- The CCP $E generation control statements and the operands thereon to indicate the requirements of his system.
- The */* card following the $E control statements. This card is replaced with a /* card.
- The *unit* on the // LOAD $CC1PP,unit card to indicate the unit on which the CCP distribution pack resides ($CC1PP is the CCP supplied disk-to-print/punch utility).
- The *unit* and *pack* parameters on the // FILE statement of the disk-to-print/punch utility OCL to indicate the unit and pack on which the work file $SOURCE is to be allocated during step 4.

The user should modify the punched control statements, without inserting any additional cards in the deck, except where he requires more than one:

- $EPLG statement — additional $EPLG statements, with both the LANG and PPUNIT parameters present, are required if more than one programming language is to be supported.
- $EMLD statement — additional $EMLD statements, with both the TYPE and XMCODE parameters present, are required if more than one type of MLTA terminal is to be supported.
- $EBSD statement — additional $EBSD statements, with the parameter TYPE present, are required if more than one type of BSC terminal is to be supported.

Note: If any errors are detected in the user's specifications, the utility program $CC1PP will print them, and will not permit the user to proceed to the next step until those errors are corrected, and the Macro Processor step repeated.

The modified deck, with the initial instruction cards removed by the user, must be placed in the system input device, and the actual generation process begun.

Step 4 (System):

The Library Maintenance program ($MAINT) first copies the Macro Processor and parts of the Overlay Linkage Editor to the distribution pack. Then the system calls the Macro Processor to analyze and expand the CCP generation $E control statements.

The process causes records to be created in the file $SOURCE. If any errors are detected in the user's specifications, only diagnostic messages from the CCP generation are written to the file. If there are no specification errors, the CCP generation writes to $SOURCE the records necessary to create the specified version of the CCP.

Step 5 (System):

The program $CC1PP reads the file $SOURCE and prints what has been generated to that file. If there are specification errors, only the user's original statements and the error diagnostic messages are printed; in order to proceed further, the user must correct those errors and perform the Macro Processor step again.

If there are no errors, the user's original statements are printed. Then the records which will be input to step 6, to create the user's version of the CCP, are printed, and are punched on the system punch device.

Step 6 (System):

The user places the punched output from the previous step, without modification, into the system input device, and begins the major step of the CCP generation. During this step the user's CCP is created. The sequence of events in this step is:

1. Creation of a load module which contains the initial contents of $CCPFILE and the instruction code to initialize that file ($CC1BF).
2. Source generation and link edit of the module $CC4, which is the resident control program during CCP operation.
3. Copying of supporting load modules for the operational stage, including startup and shutdown and for the assignment stage.
4. Creation of an initialized module $CC4Z9, if SECURE-USER was specified in the $ESEC statement, later to contain the user's security information.

5. Initialization of $CCPFILE, later to be filled with user specifications by an assignment run.

6. Copying of the subroutines and macros used in the compilation and link edit of application programs to run under the CCP.

7. Punching the following cards:

   - Stand-Alone core dump programs (see *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581).

   - OCL and Overlay Linkage Editor control statements to link edit the installation verification program.

   - OCL and sample control statements for an assignment build run, necessary in order to execute the installation verification program. The following is an example of the sample assignment input and optional link edit printout:

```
********SAMPLE ASSIGNMENT,OPTIONAL LINK EDIT AND SAMPLE START-UP DECK*****************************
*********SAMPLE ASSIGNMENT SET ***************************************
*
*****FILL IN UNIT
// LOAD $CCPAS,
*****FILL IN PACK AND UNIT
// FILE NAME-$CCPFILE,RETAIN-P,UNIT-  ,PACK-
*****FILL IN PACK AND UNIT
// FILE NAME-$CCPWORK,RETAIN-S,TRACKS-3,UNIT-  ,PACK-
// RUN
*********THE FOLLOWING STATEMENTS CAN BE MODIFIED FOR YOUR CONFIGURATION
*********BUT SOME MUST BE KEPT TO RUN CCPIVP.  SEE THE COMMENTS IN
*********THIS DECK.
*
// SET ID-@,ACTION-CREATE,DFLTEXEC-YES
// SYSTEM MINUPA-21.00K,MINTPBUF-2840,MAXEUP-2,
// PASSWORD-FECD,
// COMMANDL-50,TRACEBLK-2,SQB-2,FSB-2,DFFPACK-PROGRAM,PGMREQL-15
*
*
// TERMATTR ATTRID-1,TRANSLAT-NO,BLKL-512,DATAFORM-MESSAGE,
// VERIFYID-NO,DFF3270-YES
*
*
*********THIS STMNT TYPE REQD FOR CCPIVP OR MLTALINE STMNT
// BSCALINE TYPE-CS,LINENUM-1,POLLLIST-'00,01,10,11'
// BSCATERM TERMID-00,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
// ADDRCHAR-*60604040*,POLLCHAR-*40404040*
// BSCATERM TERMID-01,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
// ADDRCHAR-*6060C1C1*,POLLCHAR-*4040C1C1*
// BSCATERM TERMID-10,TYPE-3277M2,ATTRID-1,COMMAND-NO,
// ADDRCHAR-*61614040*,POLLCHAR-*C1C14040*
// BSCATERM TERMID-11,TYPE-3277M2,ATTRID-1,COMMAND-NO,
// ADDRCHAR-*6161C1C1*,POLLCHAR-*C1C1C1C1*
*
// TERMNAME NAME-CU0DV0,TERMID-00
// TERMNAME NAME-CU0DV1,TERMID-01
// TERMNAME NAME-CU1DV0,TERMID-10
// TERMNAME NAME-CU1DV1,TERMID-11
*
*********THIS STMNT TYPE REQD FOR CCPIVP
// DISKFILE NAME-CGIVFIL1,DEVICE-5444,ORG-C,RECL-16
*
*
*********THIS STMNT TYPE REQD FOR CCPIVP
// DISKFILE NAME-CGIVFIL2,DEVICE-5444,ORG-C,RECL-16
*********NOTE THAT ONE DISKFILE STATEMENT -CGIVFILE- WOULD BE NEEDED
*********IF SYMBOLIC FILES ARE NOT BEING USED.
*
*
// DISKFILE NAME-DUMMY1,DEVICE-5444,ORG-D,RECL-256
// DISKFILE NAME-DUMMY2,DEVICE-5444,ORG-I,RECL-64,KEYL-8,KEYPOS-1,
// MSTRINDX-YES
*********THIS STMNT TYPE REQD FOR CCPIVP IF SYMBOLIC FILES ARE USED.
// SYMFILE NAME-CGIVFILE,DISKFILE-'CGIVFIL1,CGIVFIL2'
*
*
*************THIS STMNT NECESSARY FOR CCPIVP,PACK AND PRINTER VALUES
*************CAN BE CHANGED FOR YOUR CONFIG.
// PROGRAM NAME-CCPIVP,LANGUAGE-ASSEM,PGMDATA-YES,
```

```
// FILES-'CGIVFILE/CO/NOSHR',
// PACK-PROGRAM,PRINTER-YES
*
*
*********NOTE THAT CCPIVP MUST BE ON CORRECT PACK AT STARTUP OF CCP.
*********IF THE PRINTER IS TO BE USED CCPIVR MUST BE LINK EDITED FIRST
*********AS CCPIVP.  CCPIVP MUST BE ON CORRECT PACK AT STARTUP.*******
*
// PROGRAM NAME-DUMMY1,LANGUAGE-RPGII,MRTMAX-2,PGMDATA-YES,
// FILES-'DUMMY1/DU/SHR,DUMMY2/IRUA/SHR',PACK-SYSTEM,DFFMTERM-4,
// DFFNDF-2,DFFSFDT-1006
// PROGRAM NAME-DUMMY2,LANGUAGE-COBOL,MRTMAX-2,PGMDATA-YES,
// FILES-'DUMMY1/DU/SHR,DUMMY2/IRANA/SHR',PACK-SYSTEM,DFFMTERM-2,
// DFFNDF-1,DFFSFDT-396
*/*  REPLACE WITH /*
*********END OF SAMPLE ASSIGNMENT DECK*********************************
*
*********OPTIONAL LINK EDIT DECK FOR CCPIVP IF THE PRINTER IS SUPPORTED
*********BY THIS GENERATION OF CCP*************************************
*
// LOAD $OLINK,***PACK CONTAINING OVERLAY LINKAGE EDITOR
// FILE NAME-$WORK,TRACKS-10,RETAIN-S,UNIT-  ,***FILL IN UNIT AND PACK
// FILE NAME-$SOURCE,TRACKS-10,RETAIN-S,UNIT-  ,***FILL IN UNIT AND PACK
// RUN
*
#######  FOLLOWING  L. E. STMNTS FOR NON-RPGII PACK BASED ON $EPLG GENERATION STATEMENT
// PHASE NAME-CCPIVP,RETAIN-R,UNIT-***PRODUCTION PACK
// OPTIONS MAP-XREF
*********FOLLOWING STATEMENT MUST BE FIRST INCLUDE STATEMENT
// INCLUDE NAME-CCPIVR,UNIT-***PRODUCTION PACK*******
// EQUATE OLDNAME-$$LPRT,NEWNAME-$NLPRT
// EQUATE OLDNAME-@@LPRT,NEWNAME-$$LPRT
// INCLUDE NAME-$NLPRT,UNIT-****PACK CONTAINING UNIT RECORD MODULE BASED ON $EPLG CCP
***GENERATION STATEMENT
// INCLUDE NAME-$$LPRT,UNIT-***PACK CONTAINING UNIT RECORD MODULE BASED ON $EPLG CCP
***GENERATION STATEMENT
// INCLUDE NAME-$$CSIP,UNIT-***PACK CONTAINING DSM SCP MODULES
// INCLUDE NAME-$$CSOP,UNIT-***PACK CONTAINING DSM SCP MODULES
// END
#######  END OF NON-RPGII L. E. STMNTS
*
#######  FOLLOWING L. E. STMNTS FOR RPGII PACK BASED ON $EPLG GENERATION STATEMENT
// PHASE NAME-CCPIVP,RETAIN-R,UNIT-***PRODUCTION PACK
// OPTIONS MAP-XREF
*********FOLLOWING STATEMENT MUST BE FIRST INCLUDE STATEMENT
// INCLUDE NAME-CCPIVR,UNIT-***PRODUCTION PACK*******
// INCLUDE NAME-$$LPRT,UNIT-***CCP RPGII COMPILATION PACK,BASED ON $EPLG GENERATION STATEMENT
// INCLUDE NAME-$$UPRT,UNIT-***CCP RPGII COMPILATION PACK,BASED ON $EPLG GENERATION STATEMENT
// INCLUDE NAME-$$CSIP,UNIT-***CCP RPGII COMPILATION PACK BASED ON $EPLG GENERATION STATEMENT
// INCLUDE NAME-$$CSOP,UNIT-***CCP RPGII COMPILATION PACK BASED ON $EPLG GENERATION STATEMENT
// END
#######END OF RPGII L. E. STMNTS
*
*********END OF OPTIONAL LINK EDIT STATEMENTS**************************
*
*********SAMPLE START-UP OCL FOR CCPIVP********************************
*
// LOAD $CCP,***PRODUCTION PACK
***FOLLOWING TWO // FILE STATEMENTS CORRESPOND TO SAMPLE ASSIGNMENT DECK WITH SYMBOLIC FILES****
// FILE NAME-CGIVFIL1,RETAIN-T,TRACKS-1,UNIT-  ,PACK-        ***ANY 5444
// FILE NAME-CGIVFIL2,RETAIN-T,TRACKS-1,UNIT-  ,PACK-        ***ANY 5444
***IF SYMBOLIC FILES ARE NOT USED REPLACE THE PRECEEDING TWO STATEMENTS WITH THE FOLLOWING
***SINGLE // FILE STATEMENT*************************************************
// FILE NAME-CGIVFILE,RETAIN-T,TRACKS-1,UNIT-  ,PACK-        ***ANY 5444
*
// RUN
*********END OF SAMPLE ASSIGNMENT,OPTIONAL LINK EDIT AND SAMPLE START-UP OCL FOR CCPIVP*********
/*
```

Step 7 (User):

In order to verify that an operational CCP system has been generated, the user should:

1. Modify the unit parameters in the OCL and Overlay Linkage Editor control statements, and link edit the installation verification program.

2. Modify the parameters necessary in the OCL and sample assignment statements, and perform an assignment run which specifies the necessary environmental information to execute the installation verification program (optional).

3. Start the CCP and execute the installation verification program.

GENERATION CONTROL STATEMENTS

Each CCP generation requires a set of generation control statements. If the required statements are not specified, no generation will take place. Some generation statements are always required for CCP support, and others are required only if the user desires a certain option.

Those control statements always required are:

$EIOD	(I/O Devices)
$EFAC	(CCP Facilities)
$EPLG	(Programming Languages)
$EFIL	($CCPFILE Allocation)
$EGEN	(CCP Generation Stream)

The optional control statements are:

$ESEC	(Terminal Sign-on Security)
$EMLA	(MLTA Support)
$EMLD	(MLTA Devices)
$EBSC	(BSC Support)
$EBSD	(BSC Devices)

Optional statements must be present to include support for either MLTA or BSCA (or both at the user's choice); that is, the $EMLA statement and at least one $EMLD statement must be present for MLTA support, and the $EBSC statement and at least one $EBSD statement must be present for BSCA support.

All generation control statements (if present) must be in the following order. The $EGEN statement must be last.

$EIOD
$EFAC
$EPLG
$ESEC
$EFIL
$EMLA
$EMLD
$EBSC
$EBSD
$EGEN

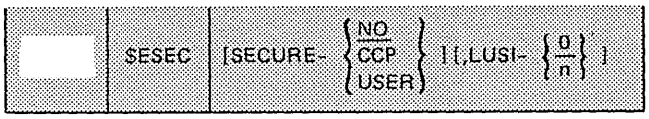
Writing Generation Control Statements

You code generation control statements as follows:

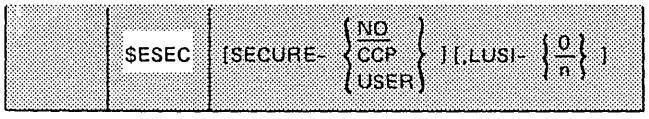
Starting Column
1 8 14 72

Name	Operation	Operands	Continuation
Symbol or blank	Statement name	No operands or one or more separated by commas	Any nonblank character if continuation is being used

The name field may contain any valid assembly language symbolic name beginning in column 1. The name is assigned to the first byte of generated code. Since the name is optional, it is not shown.

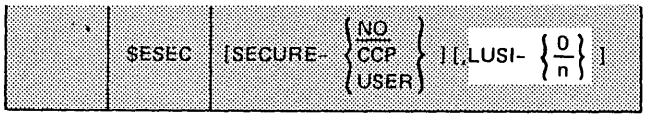


The desired mnemonic operation code (control statement name) must appear as specified in the control statement description. The operation code must start in column 8.

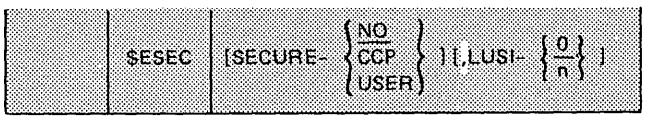


The operands specify available services and options. The operands must start in column 14 and are written as follows:

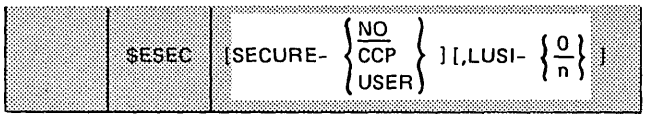
1. Each operand consists of a keyword followed by a dash and a parameter.



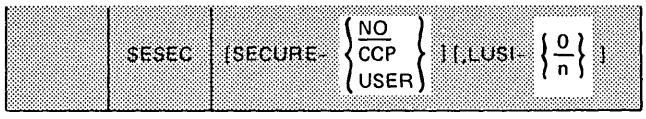
2. No blanks should be left between operands.



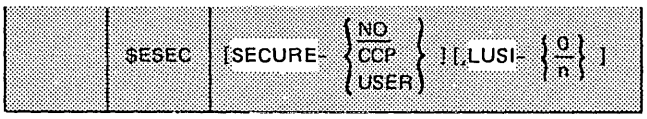
3. Commas precede all but the first operand.



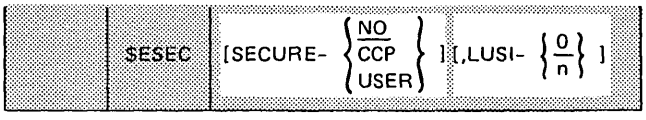
4. The parameter part of the operand must immediately follow the dash.



5. The keyword part of each operand must correspond to one of the keywords in the control statement description.

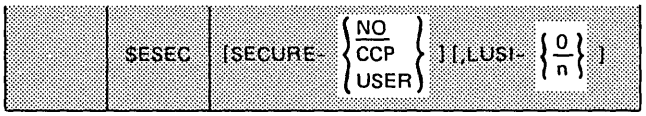


6. Some operands are not required. These optional operands are indicated by enclosing the operand within brackets [KEYWORD-parameter]. The operand enclosed in the brackets may or may not be coded, depending on whether or not the associated option is desired. The symbols [ ] are used to help define the control statements. These symbols are not to be coded; they are only used to indicate how a control statement may be written.

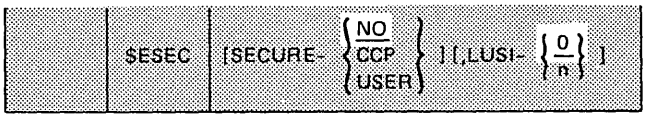


7. An option list for a keyword parameter is specified as follows:

Braces { } indicate that a choice must be made. One of the parameters from the vertical stack within braces must be coded, depending on which of the associated services is desired. The symbols { } are used to help define the control statements. These symbols are not coded; they are only used to indicate how a control statement may be written.



8. The operands may be written in any order. If a keyword is not specified, the default value is used. A default value is selected for optional keywords that are omitted. The default value is indicated in the macro instruction description by a line under the default option. For example, KEY- { A / B / C } indicates the option A is the default value.



9. No operands may be specified beyond column 71. If continuation is required, column 72 must contain a nonblank character and the last operand must be followed by a comma. An operand cannot be divided and continued on the next line. The operands of the continued field must begin in column 14. For an example of continuation coding, see Figure 6-2.

10. Comments must be separated from the operand or comma by at least one blank space. Comments cannot be inserted between operands on a one-line control statement. Figure 6-3 shows examples of comments used with control statements. On the assembly listing, all comments on the generated code are justified by the macro processor to begin in column 40. Any comments too long to be contained in columns 40 through 71 are truncated from the right.

```
1   4    8   12   16   20   24   28   32   36   40   44   48   52   56   60   64   68   72
NAME1    $MCRO  OPERAND1,OPERAND2,OPERAND3,OPERAND4,OPERAND5,OPERAND6,                    X
                OPERAND7,OPERAND8

NAME2    $MACR  OPERAND1,OPERAND2,                                                        X
                OPERAND3,                                                              X
                OPERAND4,                                                              X
                OPERAND5
```

Figure 6-2. Continuation Coding Examples

```
1   4    8   12   16   20   24   28   32   36   40   44   48   52   56   60   64   68   72
COMNT1   $MCRO  OPERAND1,OPERAND2          THIS INST. HAS TWO OPERANDS

COMNT2   $MACR  OPERAND1,                  THIS INSTRUCTION AND THIS COM-             X
                OPERAND2                   MENT ARE CONTINUED.
*                                          THIS COMMENT IS QUITE LENGTHY
*                                          AND IS ENTERED BEFORE THE IN-
*                                          STRUCTION. OTHERWISE, IT WOULD
*                                          FOLLOW THE MACRO EXPANSION IN
COMNT3   $ACRO  OPERAND1                   THE LISTING
```

Figure 6-3. Comments on Macro Instructions

Most, but not all, operands have default values. The user may elect to have a control statement present but have all operands absent if he accepts the default values.

In all cases where YES is appropriate, the single letter Y may be used. In all cases where NO is appropriate, the single letter N may be used.

In the following generation control statement descriptions the term **object system** refers to the system which will be executing the CCP after generation.

$EIOD – I/O Devices

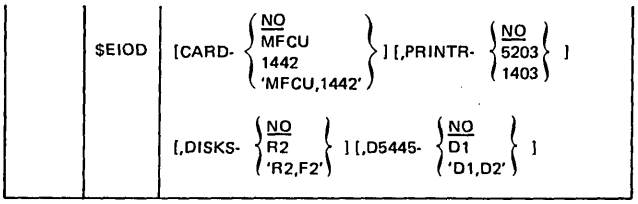
What You Must Know

_____ What card and print devices on the system are to be used by the application programs which will execute under CCP control

_____ Disk configuration of the system (5444 and 5445)

The $EIOD generation control statement indicates the unit record devices which can be used by user programs under the CCP and the disk drives attached to the object system beyond the minimum requirements.

This statement is always required and must be the first of the generation control statements. Only one $EIOD statement can be entered. All operands of this statement are optional.



CARD- { NO | MFCU | 1442 | 'MFCU,1442' }

The CARD operand indicates the card devices which can be used by the user programs under CCP. The default is NO, meaning no user program will read or punch cards. A single value (MFCU or 1442) may be specified. If both are used, 'MFCU,1442' must be specified. The value of this operand causes the appropriate card reader/punch support to be included in the CCP, and the appropriate subroutines to be copied to the user's program preparation packs.

PRINTR- { NO / 5203 / 1403 }

The PRINTR operand specifies whether or not the object system line printer can be used by the user programs, and if so, which line printer. The default is NO, meaning no user programs under the CCP will use line printer output. The value 5203 or 1403 causes appropriate line printer support to be included in the CCP, and the appropriate subroutines to be copied to the user's program preparation packs.

DISKS- { NO / R2 / 'R2,F2' }

The DISKS operand indicates which 5444 disks other than F1,R1 are available and used by the object system. The default is NO, meaning only F1 and R1 are attached to the system. R2 means F1, R1, and R2 are attached. 'R2,F2' means F1, R1, F2, and R2 are attached. The value of the operand causes appropriate 5444 disk support to be included in the CCP.

D5445- { NO / D1 / 'D1,D2' }

The D5445 operand indicates the 5445 disk drives attached to the object system. The default is NO, meaning no 5445 drives are attached and no support is included. D1 means a 5445 Model 1 (one drive) is attached. 'D1,D2' means a 5445 Model 2 (two drives) is attached. The specification of D1 or 'D1,D2' causes the appropriate 5445 support to be included in the CCP.

$EFAC – CCP Facilities

What You Must Know

_____ Maximum number of concurrently executing user programs

_____ Whether or not the system has DPF

_____ Whether or not the data mode escape facility is to be generated, and if so the six user-specified data mode escape characters

_____ Is a program request count to be kept

_____ Is disk file sharing to be allowed

_____ Will any application programs use the symbolic file name reference facility

_____ Is Display Format Facility (DFF) to be supported

The $EFAC generation control statement indicates options which determine the CCP facilities to be included during generation.

This statement is always required and must immediately follow the $EIOD statement. Only one $EFAC statement can be entered. All operands of this statement are optional.

```
        $EFAC  [MAXEUP- {1/n}] [,DPF- {YES/NO}]
               [,ESCAPE- {NO/'cccccc'/X'xxxxxxxxxxxx'}]
               [,PGMCNT- {YES/NO}]
               [,FSHARE- {YES/NO}] [,SYMFIL- {YES/NO}]
               [,FORMAT- {YES/NO}]
```

MAXEUP- { 1 / n }

The MAXEUP operand specifies the maximum number of concurrently executing user programs the CCP is capable of supporting. The default parameter value is 1 (the minimum) and the maximum is 8. A parameter value greater than 1 includes functional support to handle the concurrent execution of multiple programs. If the value 1 is specified, no multiprogram support is generated. A value of 1 requires that operand FSHARE have a value of NO. For a minimum system this value is 1.

DPF- {YES / NO}

The DPF operand specifies whether or not the CCP is to include the functional logic and control blocks to support the presence of another program level and interface with a DPF DSM supervisor. YES includes DPF support by the CCP and NO excludes this support. The default is NO. If the DSM that will execute the CCP is generated for DPF, this operand must be YES. If the DSM is generated for non-DPF, this operand must be NO. For a minimum system this value is NO.

ESCAPE- {NO / 'cccccc' / X'xxxxxxxxxxxx'}

The ESCAPE operand specifies whether or not data mode escape is supported and if so, the six data mode escape characters to be used. The data mode escape function is used by the terminal operator to instruct the CCP to accept the next input as a command to the CCP and not as data for the program with which the terminal had been communicating. This is the only method by which a terminal may interrupt the normal course of a program it requested. The CCP checks all data input from a requesting terminal for this string of characters as the first six bytes.

Care should be taken that this string is not a sequence of bytes which may inadvertently be entered as data. It is suggested that the string be made up of six special characters such as slash (/) or some other special character(s).

EBCDIC characters or hexadecimal digits can be used for this value. If EBCDIC characters are used, this value must be made up of exactly six characters. It is suggested the value be enclosed in apostrophes as shown, because certain characters which may be chosen for this string might be considered operand delimiters by the Macro Processor.

If the character apostrophe(') is to be one of the characters of the string, then each such apostrophe must be coded as four successive apostrophes, and the parameter must be bounded by apostrophes.

If the hexadecimal form is used, the parameter must be coded as:

- The letter X
- A single apostrophe (')
- Exactly 12 hexadecimal digits
- A single apostrophe (')

The default value is NO, indicating no terminal can interrupt a program and communicate directly with the CCP. For a minimum system this value is NO.

PGMCNT- {YES / NO}

The PGMCNT operand specifies whether or not a count should be kept of the number of times a user program was requested. The count is a request count, not an execution count. YES indicates these counts are to be accumulated during the execution of the CCP, and are to be added to previous counts in $CCPFILE during the CCP shutdown. The default is NO, indicating program request counts are not to be kept.

These counts may be of use to the user in designing or redesigning user programs to take advantage of certain CCP features (such as reusability or multiple requesting terminals) affecting program request response time.

For a minimum system the value is NO.

FSHARE- {YES / NO}

The FSHARE operand specifies whether the control program logic and control blocks are to be included (value YES) to support shared disk file update by concurrent user programs. The default is NO which excludes this support.

The specification of any particular program's willingness to share specific files with other concurrently running programs is made at assignment time, but no sharing of files in update mode can actually take place unless YES is indicated here.

This operand value must be NO if MAXEUP-1 was specified on this statement. No sharing can logically take place if only one user program is to be executing at a time.

To summarize the values of operands when file sharing support is to be included:

- FSHARE-YES must be specified
- MAXEUP must be greater than 1

When file sharing support is to be excluded:

- FSHARE-NO must be specified
- MAXEUP may be any valid value

For a minimum system the value is NO.

SYMFIL- {YES / NO}

The SYMFIL operand specifies whether or not symbolic disk file reference support is to be included in the CCP. With this support, a symbolic file name may be used to reference any of several disk files. For a discussion of symbolic files, see index entries *File Command (/FILE), DISKFILE statement,* and *SYMFILE statement.*

A value of YES includes symbolic file support in the CCP. The default value is NO, indicating support for symbolic files is not to be included in the CCP. If any symbolic files will be used by programs which will execute under the CCP, this value must be YES. For a minimum system this value is NO.

FORMAT- {YES / NO}

The FORMAT operand specifies whether or not the 3270 Display Format Facility (DFF) is to be included in the CCP. A value of YES includes DFF support in the CCP. The default value is NO indicating a CCP system without DFF support is to be generated.

$EPLG – Programming Languages

What You Must Know

_____ Programming languages to be supported by the CCP

_____ 5444 unit on which the pack that will be used for preparation of programs written in that language will be mounted during generation

Each $EPLG generation control statement indicates a programming language the user wishes to use for program preparation under CCP. The following rules apply to this statement:

- At least one $EPLG statement is required.
- One $EPLG statement is required for each programming language to be supported by the CCP. Only one $EPLG statement may be entered for any one language.
- The first $EPLG statement must immediately follow a $EFAC statement. Multiple $EPLG statements may come in any order after the first.
- All operands on this statement are required.

The user should note that depending on the languages chosen, the pack used for preparing programs for use under the CCP may not be valid for use in preparing programs to use directly under DSM. This condition may exist only if the CARD and PRINTR operands of the $EIOD statement indicated that one or more programs to be run under the CCP make use of a unit record device. Due to the renaming of certain modules (unit record), all four languages can be used on a pack used for preparing programs for the CCP only; but, if the pack contains RPG II, that same pack cannot be used for preparing programs for use directly under the DSM. If RPG II is not involved, a single pack containing COBOL and/or FORTRAN and/or the Assembler (any or all) can be used for program preparation for the CCP or for use directly under DSM, but the link edit step is slightly different.

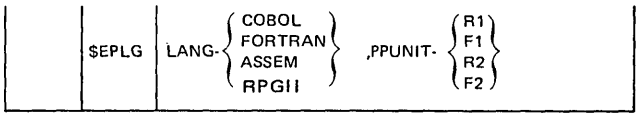
The following chart may help clarify the point:

	Under CCP	Under Both CCP and DSM
RPG II	Valid	Invalid
COBOL FORTRAN ASSEM	Valid	Valid
RPG II COBOL FORTRAN ASSEM	Valid	Invalid

Any combination involving RPG II under the CCP will cause the standard DSM unit record data management module names to be renamed and the CCP substitute intermediary modules to take the names of the standard DSM unit record data management modules. See *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579 for the program preparation considerations.

The CCP unit record intermediary routines are copied to the user's pack only if, in the $EIOD statement, unit record support was selected. The DSM unit record data management routine names are modified only if RPG II support is selected in a $EPLG statement.

In view of this, it would benefit the user to save an original version of the DSM unit record data management modules for use in case of the need to re-link edit any of them (see Figure 6-4).



LANG-{COBOL, FORTRAN, ASSEM, RPGII}

The LANG operand specifies a programming language which is to be supported by the CCP. There is no default. The value of the operand indicates that support for the language that is to be included in the CCP. ASSEM is an abbreviation for Basic Assembler Language. At least one of the languages is required.

PPUNIT-{R1, F1, R2, F2}

The PPUNIT operand specifies the disk unit containing a pack which will be used for compiling application programs for the CCP. For example, this is the pack on which the RPG II compiler would reside. Use this operand to specify the disk unit on which this pack will reside during generation. There is no default. Any 5444 unit is valid, except the unit on which the distribution pack is mounted.

This is the unit onto which generation is to copy the CCP unit record intermediary routines, renaming the DSM routines if RPG II is to be supported by the CCP.

The Assembler user not only gets data management routines, he also gets the CCP $N macros copied into his source library on this pack.

The Assembler user designates the PPUNIT as a unit on which the Macro Processor resides. The CCP $N macros must be on the same pack as the Macro Processor or the DSM pack from which the user loaded the program (IPL).

A // PAUSE card containing comments on directions for proceeding will be generated into the input stream for copying and/or renaming data management modules. This will allow the user to ensure that the correct pack is mounted on the correct unit. The routines copied to the pack on this unit are:

LANG Parameter	What is Copied
COBOL	Communications service subroutine CCPCIO
FORTRAN	Communications service subroutine CCPFIO
ASSEM	Communications service macros $NCOM, $NPLO, $NOPV, $NRTV, $NPL, and $NCIO
RPG II	Communications service subroutines SUBR90, SUBR91, SUBR92, and SUBR93

If card or print devices are to be referenced by any of the user's programs that will run under the CCP (either the CARD or PRINTR operands are specified as other than NO in the $EIOD statement), then the required unit record intermediary subroutines are copied, as well, to the pack on this unit. The routines copied depend on the devices to be supported and the language specified by the LANG operand (see Figure 6-4).

		Language Specified (LANG Operand of $EPLG Statement)					
		COBOL FORTRAN OR ASSEM		RPG II			
				On the PPUNIT		To the PPUNIT	
				The DSM routine:	Is renamed to:	An Intermediary routine named:	Is copied and given the name:
Device to be Supported ($EIOD Statement)	CARD-MFCU	$NMFFF		$$NFFF	$$UFFF	$$UFFF	$$MFFF
		$NMFRD		$$NFRD	$$UFRD	$$UFRD	$$MFRD
		$NMFPR		$$MFPR	$$UFPR	$$UFPR	$$MFPR
		$NMFPU		$$MFPU	$$UFPU	$$UFPU	$$MFPU
		$NMFRP		$$MFRP	$$UFRP	$$UFRP	$$MFRP
		$NMFRU		$$MFRU	$$UFRU	$$UFRU	$$MFRU
		$NMFPP		$$MFPP	$$UFPP	$$URPP	$$MFPP
	CARD-1442	$NARFF		$$ARFF	$$URFF	$$URFF	$$ARFF
	PRINTR- 5203 1403	$NLPRT		$$LPRT	$$UPRT	$$UPRT	$$LPRT

Figure 6-4. Required Unit Record Intermediary Subroutines Copied by Generation

$ESEC – Terminal Sign-On Security

What You Must Know

_____ What type of sign-on security will be used, if any

_____ Length of the user's security information, if a user written sign-on security routine is to be used

The $ESEC generation control statement indicates the type of terminal sign-on security to be used (if any).

This statement is optional. It is included only if terminal sign-on security is desired. If included, this statement must immediately follow a $EPLG statement. Only one $ESEC statement may be entered. All operands of this statement are optional.

	$ESEC	[SECURE- {NO / CCP / USER}][,LUSI- {0 / n}]

SECURE- {NO / CCP / USER}

The SECURE operand specifies the inclusion or exclusion of terminal sign-on security support.

The default is NO indicating no terminal sign-on security. Any command terminal may sign on to the user's system without presenting any validation information.

CCP indicates that the CCP password security routines will be included and used. Each command terminal user must then give the proper 1-6 character password as the operand of his Sign-on command before he can address the system to use further facilities. The actual password is specified in an assignment run.

USER indicates that the user desires to include his own sign-on security routines instead of using those of the CCP. These must follow the rules specified for such routines (see *IBM System/3 Model 10 Disk System Communications Control Program Programmer's Reference,* GC21-7579). In addition, at startup the user's security information must exist in a load module named $CC4Z9. It may have been placed there by the user's own method or by the CCP supplied utility $CCPAU (considered part of the assignment stage).

For a minimum system the value is NO.

LUSI- {0 / n}

The LUSI operand specifies the length (number of bytes) of the user's security information. It should be omitted unless SECURE-USER is also specified. The default value is 0 and the maximum is 4096.

The value of this operand must not be 0 if SECURE-USER was specified.

If the user specifies SECURE-USER, at startup this number of bytes is moved from load module $CC4Z9 to the user's security work area reserved in the resident CCP supervisor during the CCP generation.

For a minimum system the value is 0.

$EFIL – $CCPFILE Allocation

What You Must Know

_____ Anticipated number of assignment sets to be placed into $CCPFILE

_____ Anticipated number of programs and files in an assignment set

_____ Anticipated number of terminals in your configuration

_____ Number of main storage dumps to be retained in error situations

_____ Main storage size of processing unit

_____ Number of tracks to be reserved in $CCPFILE for CCP trace entries

_____ Disk unit and pack name on which the file is to be created

_____ Beginning track location for $CCPFILE, if you wish to position the file on the pack

The $EFIL generation control statement provides information which will affect the allocated size and location of $CCPFILE.

This statement is always required. Only one $EFIL statement may be entered. This statement must immediately follow the $ESEC statement (if present) or the $EFAC statement (if $ESEC is not present).

Except for the FLUNIT, FLPACK, and TRKLOC operands, which specify the location of the file, all other operands are used by the CCP generation only to estimate the disk space required for $CCPFILE.

You need not be overly concerned about the possibility of allocating too little space for $CCPFILE. If at some later time you find that the space is too small for your requirements, you can enlarge the space for the file by either:

- Using the $COPY program to copy the file to a larger space on disk.
- Using the program $CC1BF, which will be present on your CCP production pack, and specifying a // FILE OCL statement to indicate the number of tracks required. Run $CC1BF, then rerun assignment to reestablish the contents of $CCPFILE.

	$EFIL	[SETS- {1 / n}] [,PROGS- {10 / n}] [,DFILES- {5 / n}] [,TERMS- {1 / n}] [,DUMPS- {1 / n}] [,CORE- {24K / 32K / 48K / 64K}] [,TRACE- {1 / n}] ,FLUNIT- {R1 / F1 / R2 / F2} ,FLPACK-pack [,TRKLOC-n]

SETS- {1 / n}

The SETS operand indicates the maximum number of assignment sets which the user anticipates placing into $CCPFILE.

Only one assignment set need be defined. If space is available at assignment, it is possible to place more sets into $CCPFILE than the value given here. This value simply serves as a guideline in allocating space for the file.

The default value is 1. The maximum is 25. For a minimum $CCPFILE the value is 1.

PROGS- {10 / n}

The PROGS operand indicates the number of user programs anticipated for each assignment set in $CCPFILE. Minimum is 1 and the maximum is 255. The default value is 10.

The value given here does not place any actual restriction on the number of user programs in an assignment set, but simply serves as a guideline to the CCP generation in allocating space for $CCPFILE.

For a minimum $CCPFILE the value may be 1.

DFILES- {5 / n}

The DFILES operand indicates the average number of disk files anticipated for each assignment set in $CCPFILE. Minimum may be 0 if the user expects to use no disk files in any user programs. The maximum is 50. The default is 5.

The value given here does not place any actual restriction on the number of disk files in an assignment set, but simply serves as a guideline to the CCP generation in allocating space for $CCPFILE.

Another factor to be considered in determining this operand value is the number of // FILE cards which may be entered at the CCP startup (the maximum number of files that can be used in any CCP run). The DSM allowable maximum is 40. If more than 40 files are defined for an assignment set when starting up the CCP, the excess must be suppressed.

For a minimum $CCPFILE, a value of 0 can be specified.

TERMS-{1 / n}

This optional operand indicates the number of terminals anticipated for each assignment set in $CCPFILE. Minimum is 1 (the default value) and the maximum is 254.

The value given here does not place any actual restriction on the number of disk files in an assignment set, but simply serves as a guideline to the CCP generation in allocating space for $CCPFILE.

For a minimum $CCPFILE this value is 1.

DUMPS-{1 / n}

The DUMPS operand indicates that space is to be reserved in $CCPFILE for this number of dynamic main storage dumps. The default (and minimum) value is 1. The maximum is 9.

When a user program is forcibly terminated by the CCP and space is available in $CCPFILE for dumps, the entire contents of core will be written out to disk. If space for a complete dump of core is not available, no dump will be taken. For a discussion of the dump to disk from forcible termination, see *IBM System/3 Model 10 Disk System Communications Control Program Programmer's Reference,* GC21-7579.

The user must be aware that moving his CCP system from a smaller CPU to a larger CPU without reallocating $CCPFILE will reduce the number of dumps which can be taken, because each dump will require more space than on a smaller CPU. If, at startup, space is not available for at least one dump, the CCP is not allowed to continue execution.

CORE-{24K / 32K / 48K / 64K}

The CORE operand indicates object CPU size. The default is 24K. This operand value indicates the total main storage size, not the size of the anticipated CCP program level partition.

$CCPFILE dynamic main storage dump space allocation is based on this value. At startup, the DSM configured size of the CPU running the CCP is used to determine how many core dumps can be taken during the run.

TRACE-{1 / n}

The TRACE operand indicates the number of tracks which are to be reserved in $CCPFILE for CCP trace entries. The default is 1 (the minimum) and the maximum is 20.

At the system operator's option, the CCP trace entries, used as a service aid, may be written from the internal trace table to disk. Each track holds 384 entries. For a discussion of the operator command to write trace entries to disk, see *IBM System/3 Model 10 Disk System Communications Control Program System Operator's Guide,* GC21-7581.

FLUNIT-{R1 / F1 / R2 / F2}

The FLUNIT operand specifies the unit on which, during generation, the pack where $CCPFILE is to be allocated. Any of the 5444 units is valid. There is no default parameter.

The value specified here need not be the CCP production pack.

During the second pass of the CCP generation, when the space for $CCPFILE is about to be allocated and its contents initialized, a // PAUSE statement which is provided in the input will permit the system operator to verify that the correct pack is mounted on this unit and, if not, to mount it.

FLPACK-pack

The FLPACK operand specifies the 5444 pack name upon which $CCPFILE is to be allocated. There is no default value.

TRKLOC-n

The TRKLOC operand can be used to specify the beginning track location for the allocation of $CCPFILE.

If this operand is omitted, $CCPFILE will be allocated in the location determined by the DSM (the highest track locations available).

$EMLA – MLTA Support

What You Must Know

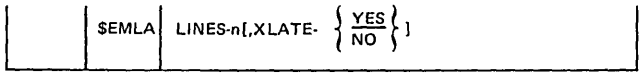
_____ Number of MLTA lines to be supported

_____ If MLTA is to be supported, whether all programs will deal with translated (EBCDIC) data, or whether some programs will operate upon data in line transmission code

The $EMLA generation control statement indicates whether or not the Multiple Line Terminal Adapter (MLTA) is to be used by the CCP and, if so, the number of MLTA lines.

This statement is optional. It need be included only if the MLTA is to be supported. Either the MLTA statements ($EMLA and $EMLD) or the Binary Synchronous Communication Adapter (BSCA) statements ($EBSC and $EBSD) must be included. Both may be specified. Only one $EMLA statement may be entered. If included:

- This statement must immediately follow the $EFIL statement.
- At least one $EMLD statement is required to immediately follow this statement if the value of operand LINES is not zero.



LINES-n

The LINES operand indicates the number of teleprocessing lines on the user's MLTA. The minimum is 0, and the maximum is 8.

By implication, 0 indicates no MLTA support is to be generated into the CCP and thus no $EMLD control statements are allowed.

If this value is not 0, at least one $EMLD statement is required.

XLATE- {YES / NO}

The XLATE operand specifies whether or not, in the user program communications operations, programs **always** will require translation of line transmission code to EBCDIC on input and translation from EBCDIC to line transmission code on output.

The default is YES indicating that translation is always used in MLTA communications operations. NO indicates that translation may sometimes be suppressed, via a specification in the assignment stage TERMATTR statement.

Regardless of the value of this operand, the forcing of upper case translation of input data is always allowed in the TERMATTR statement.

For a minimum system, the value is YES.

$EMLD – MLTA Devices

What You Must Know

_____ MLTA terminal devices to be supported

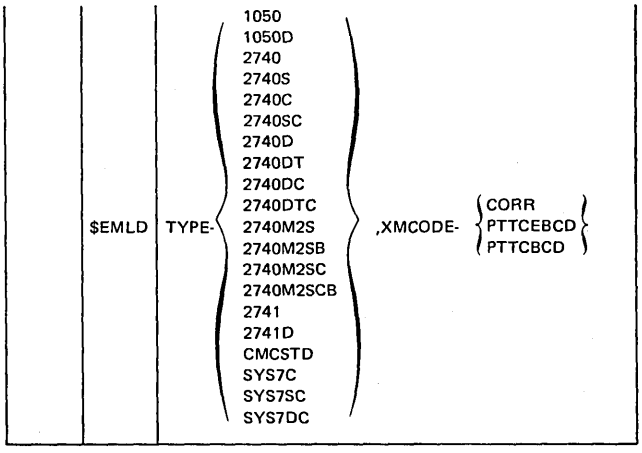
_____ MLTA line transmission code for each terminal type

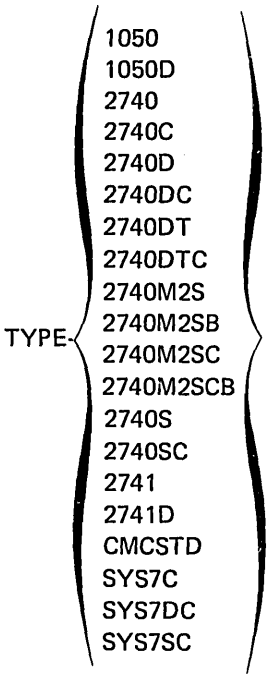
The $EMLD generation control statement indicates an MLTA terminal type to be supported with its features, the type of line the terminal is on, and the transmission code required on the line.

This statement is optional. It need be included only if the MLTA is to be supported.

If the MLTA is to be supported:

- At least one $EMLD statement is required.
- One $EMLD statement is required for each unique terminal-type/transmission-code combination (not one for each terminal).
- The first $EMLD statement must immediately follow a $EMLA statement. Other $EMLD statements must follow the first.
- Both operands on this statement are required.





The TYPE operand indicates the type of terminal with its features on a dial (switched) line or non-dial (nonswitched) line.

The specification of a particular terminal type in a $EMLD statement causes the control logic support for that terminal type to be included in the CCP. Omission of a terminal type among the user's $EMLD statements indicates support for that type is not desired.

The following table explains the meaning of each terminal type code:

Terminal Type	Description
1050	1050 on a nonswitched line
1050D	1050 on a dial (switched) line
2740	2740 Model 1 without extra features on a nonswitched line
2740C	2740 Model 1 with longitudinal redundancy checking (LRC) feature on a nonswitched line
2740D	2740 Model 1 without extra features on a dial (switched) line
2740DC	2740 Model 1 with longitudinal redundancy checking (LRC) feature on a dial (switched) line
2740DT	2740 Model 1 with transmit control feature on a dial (switched) line
2740DTC	2740 Model 1 with transmit control and longitudinal redundancy checking (LRC) features on a dial (switched) line
2740M2S	2740 Model 2 with station control feature on a nonswitched line
2740M2SB	2740 Model 2 with station control and buffer receive features on a nonswitched line
2740M2SC	2740 Model 2 with station control and longitudinal redundancy checking (LRC) features on a nonswitched line
2740M2SCB	2740 Model 2 with station control, longitudinal redundancy checking (LRC), and buffer receive features on a nonswitched line
2740S	2740 Model 1 with station control feature on a nonswitched line
2740SC	2740 Model 1 with station control and longitudinal redundancy checking (LRC) features on a nonswitched line
2741	2741 on a nonswitched line
2741D	2741 on a dial (switched) line
CMCSTD	Communicating Magnetic Card SELECTRIC ® Typewriter on a dial (switched) line. The CMCST is supported to the extent that it functions identically to a 2741D
SYS7C	System/7 functioning as a 2740 with longitudinal redundancy checking (LCR) feature on a nonswitched line. The System/7 is supported to the extent that it functions identically to a 2740C

Terminal Type	Description
SYS7DC	System/7 functioning as a 2740 with longitudinal redundancy checking (LRC) feature on a dial (switched) line. The System/7 is supported to the extent that it functions identically to a 2740DC
SYS7SC	System/7 functioning as a 2740 with station control and longitudinal redundancy checking (LRC) features on a nonswitched line. The System/7 is supported to the extent that it functions identically to a 2740SC

XMCODE- {CORR / PTTCEBCD / PTTCBCD}

The XMCODE operand indicates the MLTA line transmission code which the terminal in this $EMLD statement will use.

CORR — Correspondence Code

PTTCEBCD — Paper Tape Transmission Code EBCDIC

PTTCBCD — Paper Tape Transmission Code BCD (Binary Coded Decimal)

Specification of a particular code causes generation to include the line-code-to-EBCDIC (upper and lower case) and EBCDIC-to-line-code translation modules when copying modules to the production pack.

The following table indicates the valid values of XMCODE for each terminal type with its features:

Terminal	Valid Transmission Code(s)
1050 1050D	PTTCEBCD
2740 2740C 2740D 2740DC 2740DT 2740DTC 2740M2S 2740M2SB 2740M2SC 2740M2SCB 2740S 2740SC	CORR PTTCEBCD PTTCBCD
2741 2741D	CORR PTTCEBCD PTTCBCD
CMCSTD	CORR
SYS7C SYS7DC SYS7SC	PTTCEBCD

$EBSC – BSC Support

What You Must Know

_____ BSCA lines, line features, and BSC control logic to be included in the CCP support

_____ BSCA line transmission code for each terminal type

The $EBSC generation control statement indicates general specifications concerning Binary Synchronous Communications (BSC) support.

This statement is optional. It need be included only if the Binary Synchronous Communications Adapter (BSCA) is to be supported. Either the BSCA statements ($EBSC and $EBSD) or the MLTA statements ($EMLA and $EMLD) must be included. Both may be specified. Only one $EBSC statement may be entered. If included:

- This $EBSC statement follows the last $EMLD statement if MLTA support is included. If MLTA support is not included, this statement immediately follows the $EFIL statement.
- At least one $EBSD statement is required to immediately follow this statement if the value of the BSCA operand is other than zero.

	$EBSC	[BSCA-{0/1/2}][,DIAL-{YES/NO}][,PP-{YES/NO}] [,MP-{YES/NO}][,CS-{YES/NO}][,GETMSG-{YES/NO}] [,ITB-{YES/NO}][,RECSEP-{1E/XX}] [,ASCII-{YES/NO}][,EBCDIC-{YES/NO}] [,RESPOL-{YES/NO}][,AUTORS-{YES/NO}] [,XPRNCY-{YES/NO}]

BSCA-{0/1/2}

The BSCA operand indicates the number of BSC adapters (lines) the CCP is to support. The minimum (and default) is 0. Maximum is 2.

A value of 0 indicates that BSC support is not to be included, in which case no other operands should be specified (if specified, they must not have other than default values). Also no $EBSD statements are allowed.

A value of 1 or 2 indicates that BSC control logic support is to be included in the CCP.

DIAL-{YES/NO}

The DIAL operand indicates whether a BSCA dial (switched line) network is to be supported by the CCP. The default is NO.

DIAL-YES is valid only if the value of operand BSCA was 1 or 2, and the 3735 or CPU is to be supported.

YES includes the support and NO excludes it.

PP-{YES/NO}

The PP operand indicates whether a BSCA point-to-point (leased or private) network between CPUs is to be supported by the CCP.

PP-YES is valid only if BSCA-1 or BSCA-2 was specified.

YES includes the support and NO excludes the support.

MP-{YES/NO}

The MP operand indicates whether a BSCA multipoint tributary (leased or private) network is to be supported under the CCP. This means the System/3 with the CCP is polled and addressed by another computer.

MP-YES is valid only if BSCA-1 or BSCA-2 was specified. YES includes the support and NO excludes the support.

The default is NO.

CS-{YES / NO}

The CS operand indicates whether a BSCA control station (leased or private) network is to be supported under the CCP. This means the System/3 with the CCP polls and addresses other terminals.

CS-YES is valid only if BSCA-1 or BSCA-2 was specified. YES includes the support and NO excludes the support.

CS-YES must be specified if 3270 type terminals are to be supported under the CCP.

The default is NO.

GETMSG-{YES / NO}

The GETMSG operand indicates whether the BSC control logic to *get a message* should be included in the CCP (see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579).

YES includes the functional capability to read (in one user requested operation) from start-of-message to EOT. For example, one could read from many separated fields on the screen of the 3277 and receive all the data as one continuous string. GETMSG-YES is required if FORMAT-YES is specified on the $EFAC statement.

The default is NO and excludes the support.

ITB-{YES / NO}

The ITB operand indicates whether or not BSC intermediate-text-block characters will be used in communicating with BSC terminals.

YES on this operand is valid only if BSCA-1 or BSCA-2 was specified. YES includes the control logic to handle fixed length block records with intermediate text-block characters.

The default is NO and excludes the support.

RECSEP-{1E / xx}

The RECSEP operand specifies the hexadecimal record separator byte for BSC transmission. A CCP generated to support BSC terminals always includes code for handling variable length records.

The user can specify a record separator byte as any two hexadecimal digits. When separator bytes are used, the most commonly used byte is hexadecimal 1E which is the default value. The record separator byte specified here is the character to be used in every instance of variable length block records.

ASCII-{YES / NO}

The ASCII operand specifies whether any CCP supported BSC adapter uses ASCII transmission code.

YES includes the capability of translating between ASCII transmission code and EBCDIC code used by programs in the system. If BSCA is not 0, either ASCII-YES must be specified, or EBCDIC-NO must not be specified.

The default is NO and excludes the translation for ASCII code.

EBCDIC-{YES / NO}

The EBCDIC operand specifies whether any CCP supported BSC adpter uses EBCDIC transmission code.

When no ASCII or EBCDIC operand is specified and BSCA is not 0, EBCDIC-YES is assumed.

The default if YES.

RESPOL-{YES / NO}

The RESPOL operand specified whether BSC polling modules are to be core resident rather than executed as transients.

YES makes the polling routines resident. NO specifies polling is to be done by transients. There is no difference in function in the two methods. Core resident polling is used only for improved response time. The user may wish to use this feature (core resident polling) when his programs may be interactive with many terminals on one line.

YES is valid only if BSCA-1 or BSCA-2 was specified in this same control statement. If RESPOL-YES is specified, CS-YES must be in effect.

The default is NO.

AUTORS- {YES / NO}

The AUTORS operand specifies whether the BSC multipoint tributary (non-control station) support to automatically send a negative response to polling and addressing sequences is to be included in the CCP. This will be in effect only after the first data transfer has completed and will remain in effect until a subsequent BSCA I/O operation or a MLMP CLOSE operation is issued to the line. This allows the CCP to respond faster and avoid possible timeout problems on the CPU that is polling or addressing your system.

YES causes the proper module to be core resident. NO means that the module has to be loaded each time it is required, and timeouts may occur while the module is being loaded. The user should specify YES if he expects serious timeout problems.

If AUTORS-YES is specified, MP-YES must be in effect. The default is NO.

XPRNCY- {YES / NO}

The XPRNCY operand specifies whether or not the text transparency feature is to be used by application programs that will run under control of the CCP.

YES includes support for text-transparency. NO excludes the support.

If YES is specified, EBCDIC-NO must not be specified in this $EBSC statement.

The default is NO.

$EBSD — BSC Devices

What You Must Know

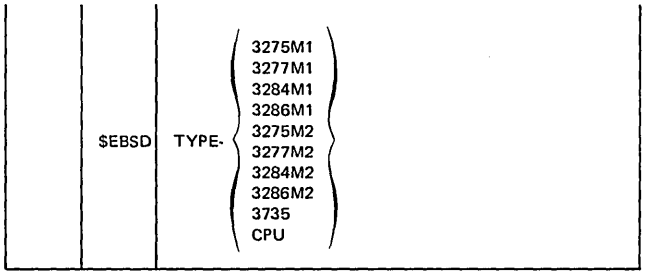
_____ BSCA terminal devices to be supported

The $EBSD generation control statement indicates a BSC device type which the CCP is to support.

This statement is optional. It need be included only if BSC is to be supported.

If BSC is to be supported:

- At least one $EBSD statement is required.
- One $EBSD statement is required for each unique terminal type.
- The first $EBSD statement must immediately follow a $EBSC statement. Other $EBSD statements immediately follow the first. Multiple $EBSD statements may be in any order.
- The operand on this statement is required.



TYPE- { 3275M1 | 3277M1 | 3284M1 | 3286M1 | 3275M2 | 3277M2 | 3284M2 | 3286M2 | 3735 | CPU }

The TYPE operand specifies a particular BSC terminal type to be supported. Specification of a terminal type causes the necessary control logic to be included into the CCP to support that type. On individual $EBSD statements, any or all of the above terminal types may be specified.

The 3272 is not supported by the CCP.

M1 indicates a 480-byte buffer terminal and M2 indicates a 1920-byte buffer terminal of the 3270 System series.

3735 indicates the 3735 Programable Terminal. For a description on how to create and transmit forms descriptor program (FDP), see *IBM 3735 Programmer's Guide,* GC30-3001; and *IBM System/3 Model 10 Disk System 3735 Application Package Coding Manual,* GC21-5096.

CPU indicates support for all CPUs capable of receiving or transmitting in BSC (see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579).

$EGEN — CCP Generation Stream

What You Must Know

_____ Disk unit and pack name on which disk system management resides (F1 or R1)

_____ 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

_____ Disk unit on which the Macro Processor and the distribution CCP modules reside

The $EGEN generation control statement indicates where various unit and pack names are located during the CCP generation, and permits the user to specify the minimum size of resident code.

This statement is always required. It must be the last of all $E generation control statements. Only one $EGEN statement can be entered.

The disk units used in a CCP generation are not fixed, as in a generation of DSM. Any available unit may be used to hold any pack required, so long as:

- The distribution pack remains mounted on the same unit throughout generation.
- The system pack remains mounted on the same unit throughout generation.
- The distribution pack is not used as the receiving pack for any relocatable or load modules produced by generation.

```
$EGEN  DSUNIT- {F1|R1}
       CCUNIT- {F1|R1|F2|R2}
       WKUNIT- {unit|'unit,unit,unit'}
       WKPACK- {pack|'pack,pack,pack'}  ,DIUNIT- {F1|R1|F2|R2}
       [MINRES- {YES|NO}]
```

DSUNIT- {F1 / R1}

The DSUNIT operand specifies which 5444 disk unit (F1 or R1) contains the DSM during the CCP generation. This is the unit from which the user loads the DSM (performs IPL). Either of the values is valid. There is no default.

CCUNIT- {F1 / R1 / F2 / R2}

The CCUNIT operand specifies the production 5444 disk unit upon which the generated CCP modules, except for those which directly support application program preparation (specified in the $EPLG statement), are to be generated. Any of the values is valid. There is no default. This unit can be the same as that specified for DSUNIT.

WKUNIT- {unit / 'unit,unit,unit'}

The WKUNIT operand specifies the 5444 disk unit(s) upon which work files are to be allocated during the CCP generation. The value for unit may be R1, F1, R2, or F2. There is no default value.

Either the single unit form or the triple unit form must be used. The same form must be used for the WKPACK operand. If the single unit form is used, $SOURCE, $WORK, and $WORK2 will all be allocated on the same (specified) unit whenever they are required.

If the triple unit form is used, the first unit indicates where $SOURCE is to be allocated whenever it is required. The second unit indicates where $WORK is to be allocated whenever it is required. The third unit indicates where $WORK2 is to be allocated whenever it is required.

The work space required is as follows:

- For $SOURCE: 120 tracks
- For $WORK: 40 tracks
- For $WORK2: 40 tracks

There is sufficient space for all of these work files on the distribution pack, PID002 (full capacity pack only). However, the time required for generation can be significantly reduced if these files (particularly $SOURCE) can be allocated on another pack, especially if that pack can be on a unit served by a separate disk arm.

WKPACK- {pack / 'pack,pack,pack'}

The WKPACK operand specifies the pack name(s) upon which the work files are to be allocated during the CCP generation. The value for pack is the appropriate user's pack name. There is no default value.

Either the single pack form or the triple pack form must be used. The same form must be used as for the WKUNIT operand. If the single unit form was used for WKUNIT, then the single pack form must be used here. If the triple unit form was used for WKUNIT, then the triple pack form must be used here. If the single pack form is used, $SOURCE, $WORK, and $WORK2 will all be allocated on the same (specified) pack whenever they are required.

If the triple pack form is used, the first pack indicates the pack name for allocation of $SOURCE whenever it is required. The second pack indicates the pack name for allocation of $WORK whenever it is required. The third pack indicates the pack name for allocation of $WORK2 whenever it is required.

DIUNIT- {F1 / R1 / F2 / R2}

The DIUNIT operand specifies the 5444 disk unit which contains the Macro Processor and the distribution CCP modules as shipped from the IBM Program Information Department (PID). The value must not specify a unit which is the same as that specified for CCUNIT, nor the same as any specified for PPUNIT in a $EPLG statement.

MINRES- {YES / NO}

The MINRES operand specifies whether or not the size of the resident CCP control routine is to be made as small as possible by making certain communications handling routines transient rather than resident, as well as by deleting code that provides an incrementing halt code during the operational stage. For a description of the incrementing halt code, see *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581.

YES indicates that the resident control routine is to be made as small as possible by the above method. NO, the default, indicates that these functions are to remain resident.

MINRES-YES makes available more space in main storage for user programs during a CCP run at the cost of a certain degree of response time of the system. The difference in main storage space is identified in *Appendix F. Storage Estimates.*

CCP PROGRAMS USED IN GENERATION

Print/Punch Utility ($CC1PP)

The disk-to-print/punch program ($CC1PP) is used in step 5 of the CCP generation to print any error messages, or if there are no errors, to print and punch the input stream for the second pass of generation.

The output from $CC1PP can be punched on either the MFCU or the 1442, whichever is the standard punch device for the system on which generation occurred.

SCP Generator ($CGxxx)

The System Control Program Generator ($CGDRV and related $CGxxx phases) is used in the second pass of generation (step 6) to generate:

- The initialization data for $CCPFILE.
- The execution time resident load module ($CC4).
- A null user security information module if a user security routine is to be used.

Initialize Assignment File Build ($CC1BF)

The initialize Assignment File Build program ($CC1BF) allocates and initializes $CCPFILE in step 6 of the CCP generation to prepare $CCPFILE for the assignment stage.

Reexecution of $CC1BF (after proper scratching of $CCPFILE) will reinitialize $CCPFILE to the state it was before the first assignment set was entered.

By changing the unit and pack name on the // FILE card of $CC1BF, $CCPFILE may be initialized on another unit and pack. By changing the TRACKS parameter of the // FILE card, the size of the new $CCPFILE can be changed. Multiple $CCPFILEs can exist so long as each is on a separate pack. Selection of the unit for the appropriate $CCPFILE can be made at startup.

SOURCE MODULES USED IN GENERATION

In addition to the source library modules used for generating the execution time resident code or supporting transients, the following source library members are present on the distribution pack:

$CGSMP — CCP sample generation deck

$CGEND — Instructions to the user at the end of generation

$CGSET — Sample assignment input deck for the Installation Verification program (CCPIVP)

CCPDAN — Core dump program to be loaded from a 5424; prints the dump on a 5203 or 1403 with AN2 or LC2 print chain

CCPDHN — Core dump program to be loaded from a 5424; prints the dump on a 5203 or 1403 with HN2 print chain

CCPDPN — Core dump program to be loaded from a 5424; prints the dump on a 5203 or 1403 with PN2 print chain

CCPDTN — Core dump program to be loaded from a 5242; prints the dump on a 5203 or 1403 with TN5 print chain

For additional information about the core dump programs, see *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581.

Chapter 7. Assignment Stage

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

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 6. Generation Stage*). Each time the CCP is run, it operates under one of the assignment sets; that is, the CCP has access to a particular group of terminals, files, and programs. The user can restrict the resources defined by a set during operation 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.

While the range of function specified during the CCP generation will not vary by assignment set, the specific operating environment of the CCP execution may vary. For example, at generation the user may have limited the number of user programs which are permitted to run concurrently to 4; but the actual number which will be allowed to run may vary from one assignment set to another (up to four user programs).

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 the system environment changes, the contents of the assignment file can be modified by repeating the assignment run, without regenerating the CCP.

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 each time certain other aspects of the CCP configuration change. The assignment stage of the CCP can be run frequently or infrequently, depending upon how often the CCP environment changes.

If the CCP is to 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 can 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 weekend, 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 actual 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:

- 18K 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, $CCPFILE, and one for a work file ($CCPWORK).

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 syntax errors.

The Assignment List program, $CCPAL, analyzes the LIST statement for errors.

Error messages are written to the system logging devices.

ASSIGNMENT FILE ($CCPFILE)

During an assignment run, the user defines a specific CCP run environment in a set of tables contained in the CCP assignment file on disk. This file, allocated and initialized during the CCP generation, always has the name, $CCPFILE. When the user starts his CCP in actual operation, the startup routines of the CCP will, using the information in $CCPFILE, initialize the CCP for this run so that only a specific set of terminals, files, and programs indicated by the user may be used.

A set of specifications in the assignment file, once set by the user in an assignment run, hold for any number of operational runs of the user's CCP. The assignment file need be changed only when one or more sets of specifications needs to be changed.

Assignment allows the user to have more than one operational environment in which to run the CCP. He might, for example, have one set of terminals, files, and programs with which he works during the day, and a different (though possible overlapping) set with which he must deal at night. His requirements might even be for more than two such environments. The assignment file may, if the user desires, contain several different set of specifics.

The user may have more than one $CCPFILE but only one per pack. The CCP system operator may, upon any startup of an operational run of the CCP, identify the unit containing the appropriate $CCPFILE and the set of specifics that shall apply to the current run.

Note: The maximum number of sets allowed in one assignment file is 25.

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 Build program analyzes the statements to ensure that they are valid. Following successful validation, the specifications are encoded by $CCPAS and written into the assignment file as an assignment set (for example, a group of tables in the assignment file, $CCPFILE, which defines one CCP operating environment). 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.

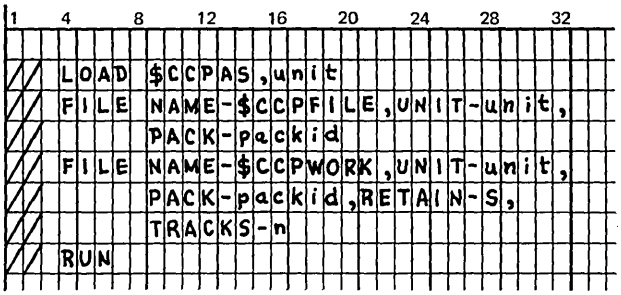
A set of assignment statements begins with a statement identifying the assignment set and whether it is to be modified, created, deleted, or replaced. Only this statement is required to delete a set. Subsequent statements provide the specific information to be placed in the assignment set. When an item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

$CCPAS may be run in any of four execution modes:

1. Create a new assignment set.
2. Delete an existing assignment set.
3. Replace an existing assignment set.
4. Modify the system environment portion of an existing assignment set.

The input for $CCPAS is as follows:

- The assignment file $CCPFILE.
- The OCL FILE statements provided by the user. The source statements must be entered from the system input device. The following OCL statements are needed:



Notes:

1. The unit parameter for $CCPAS specifies the location of the Assignment Build program.
2. The unit parameter for $CCPFILE specifies the location of the assignment file to be operated on.
3. $CCPWORK should have RETAIN-S specified to allow deletion after use and can be on any 5444 unit the user chooses.
4. The keyword RECORDS should not be used to define the size of $CCPWORK; instead the keyword TRACKS should be used. Additionally, the size must be large enough to contain the largest set to be processed (see *Appendix F. Storage Estimates*).

- The assignment control statements instructing $CCPAS of the function(s) it is to perform. $CCPAS reads the control statements from the system input device. Each group of control statements, grouped together by the user to define an operating environment, is called an *assignment set*. The input to $CCPAS can consist of one or more sets of control statements.

  The following control statements define an assignment set and must be given in the order shown:

// SET	This required statement identifies the output assignment set operated on and the type of operation to be performed. If multiple input assignment sets are provided as input to $CCPAS, the // SET statement defines the beginning of each assignment set.
// SYSTEM	This required statement defines certain operational environment options not directly related to terminals, files, or programs when this set is used for execution of the CCP.
// TERMATTR	Each of these statements defines the variable attributes of a terminal. Each terminal statement (BSCATERM and MLTATERM) must reference a TERMATTR statement. *Note:* Any BSCALINE statement with corresponding BSCATERM statements (as a group) can appear before or after any MLTALINE/MLTATERM group. The order of input lines determines the priority of service during execution of the CCP. For example, the order may be: MLTALINE MLTATERM (one or more) BSCALINE BSCATERM (one or more) MLTALINE MLTATERM (one or more)
// BSCALINE	Each of these optional statements identifies a BSCA line, its characteristics, and the order in which the terminals on that line (if there is more than one terminal) shall be polled.
// BSCATERM	At least one of these statements must follow a BSCALINE statement. Each BSCATERM statement identifies a terminal on the line, its operating characteristics, and whether or not it can issue commands to the CCP. *Note:* The BSCATERM statement must follow the BSCALINE statement for which the terminal is defined.
// MLTALINE	Each of these optional statements identifies a MLTA line, similar to the BSCALINE statement.

// MLTATERM	At least one of these statements must follow a MLTALINE statement. Each MLTATERM statement identifies a terminal on the line, its operating characteristics, and whether or not it can issue commands to the CCP. *Note:* The MLTATERM statement must follow the MLTALINE statement for which the terminal is defined.
// TERMNAME	Each of these required statements defines a symbolic terminal name by which a terminal may be referenced in an application program. The symbolic terminal name can be assigned to a physical terminal or can be left unassigned.
// DISKFILE	Each of these optional statements defines a user disk data file and the physical attributes of the file.
// SYMFILE	Each of these optional statements defines a symbolic name that can be used to reference one or more disk files specified on // DISKFILE statements.
// PROGRAM	Each of these required statements identifies a user program which is permitted to execute during a CCP run which uses this set. The statement defines the program resource requirements and operational characteristics.

When creating or replacing an assignment set, at least one BSCALINE and one BSCATERM statement, or one MLTALINE and one MLTATERM statement must be the control statements.

- A /* statement must be given at the end of all input control statements.

The output for the assignment build program is as follows:

- A listing of the assignment control statements on the printer.
- Appropriate diagnostics on the printer or log device and an error halt, at the end of the run, if necessary.
- The $CCPFILE containing created, replaced, or modified assignment sets, and no longer containing any sets specified to be deleted. The $CCPFILE is ready for use in operational startup.

An assignment set will be created or replaced in the assignment file only if the complete set, as read by the Assignment Build program, is free of errors.

In a DPF system, $CCPAS cannot run concurrently in both levels nor run concurrently with the CCP in the other level. ($CCPFILE cannot be validly altered from both levels concurrently or while the CCP is running.)

ASSIGNMENT CONTROL STATEMENTS

Each control statement contains an identifier and parameters. The identifier is a term that defines the type of control statement and is always the first field of the statement following the //. The parameters are the control information being supplied to the program. Each parameter consists of a keyword to identify the parameter, a hyphen, and the appropriate control information value. Parameters within any single control statement can be entered in any order.

Writing Assignment Control Statements

The rules for constructing assignment control statements are as follows:

Statement Identifier: // in columns 1 and 2 followed by at least one blank preceding the statement identifier.

Blanks: One or more blanks is required between the // and the statement identifier, and between the statement identifier and the first keyword. The first blank following a keyword value indicates the end of the statement. A blank following a comma (which follows a value given for a keyword parameter) indicates that the statement is continued on the following input record (which must have a //b in the first three columns).

Keyword Parameters: These are separated by commas. A hyphen (-) separates each keyword from the corresponding value. Blanks are not allowed within or between parameters.

Sublist: A Sublist is a series of values given for a single keyword. Each value is related (such as a series of filenames). The first value of a sublist always has an apostrophe (') given before the value, succeeding values are separated by commas, and the last value has an apostrophe after it. For example: EXAMPLE-'value,value,value'.

Split-Value: A split-value is a group of values which may be given for a keyword parameter; however, each has a separate and distinct relationship to the keyword. These split-values are separated by a slash (/). If a series of split-values is given, the series is called a split-value sublist. If a split-value is not given as a sublist, then apostrophes need not be given in front of and behind the complete split-value.

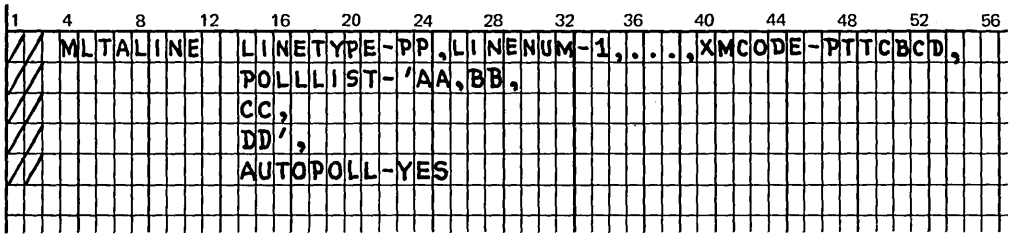
- Split-value example: KEYWORD-value1/value2/value3
- Split-value Sublist example: KEYWORD-'value1/value2, value1/value2'

Special Characters: Slashes (/), blanks, commas (,), hyphens (-), and apostrophes (') must not appear within the bounding apostrophes if a keyword is defined as either sublist or split-value capable. Any of these characters can appear within the apostrophes if the keyword is not defined as either sublist or split-value capable. In this case, a single apostrophe (') must be represented by two successive apostrophes ('').

Statement Length: Positions 1 through 71 of a record may be used. The first blank encountered following a keyword parameter, without indication of continuation to the next record, delimits the statement.

Continuation: Control statements can be continued on subsequent input records by placing a comma immediately after a keyword parameter value on a statement. The comma must be followed by one or more blanks prior to or including column 72. (Thus the comma can appear in column 71 and a blank in column 72.) A continuation line must start with two slashes (//) followed by one or more blanks, followed by the remaining values (if part of a sublist) or the remaining keyword parameters. Any number of continuation records can be used.

For example:



Comments: The user can include, in addition to control statements, special statements that contain only comments. A comment statement is identified by an asterisk (*) in column 1 and may contain any combination of characters. Comments are limited only by the length of the physical input record. Any comments too long to be contained in 88 columns are truncated from the right.

In addition, comments can be included on any record of a control statement following the last parameter on that record. One or more blanks must separate the last parameter from the comments.

Special Meaning of Capital Letters, Numbers, and Special Characters: Capitalized words and letters, numbers, and special characters have special meanings in control statement descriptions:

- In control statements, capitalized words and letters must be written as they appear in the statement description. If numbers appear with the capitalized information, they must also be written as shown.
- Lower case terms indicate that a value must be substituted for each term.

SET Statement

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

```
// SET    [ID- {1|c} ] [,ACTION- {CREATE|REPLACE|DELETE|SYSMOD} ]
          [,DFLTEXEC- {YES|NO} ]
```

ID- {1 | c}

The ID parameter specifies which assignment set is to be affected. Each assignment set in $CCPFILE is identified by a different single character.

If the parameter is omitted, the default value is the number 1. Any single extended alphameric character (0-9, A-Z, #, $, @) can be used to identify the assignment set.

ACTION- {CREATE | REPLACE | DELETE | SYSMOD}

The ACTION parameter specifies what operation is to occur. If it is omitted, the default value is CREATE.

CREATE. If CREATE is used and the set ID does not exist in the file, the new set is added to the file. If the created set cannot fit, either because there is not enough room in the file for the tables or not enough room for the entry in the directory, an error message is given and the request ignored. A maximum of 25 sets are allowed in the control file.

If a set exists already in the file with the same ID character, an error diagnostic is given and all succeeding control statements to the next SET or /* statement are checked for syntax errors. A diagnostic message is issued and the request to create this set is ignored.

REPLACE. If REPLACE is used and the set exists in the file, the set in the file is entirely replaced with the information on the statements following the SET statement. If the value is REPLACE and the set cannot be found in $CCPFILE, a warning message is given and the action defaults to CREATE.

DELETE. If DELETE is used and the set ID does exist in the $CCPFILE, the set and its table values are deleted from the file. If the value is DELETE and the specified set does not exist in the file, a warning diagnostic is given.

SYSMOD. The SYSMOD action indicates that only the SYSTEM statement, or the execution default set identification, or both are to be modified in a set. The set must already be in the file (for further definition, see index entry *SYSTEM Statement*).

DFLTEXEC-{YES / NO}

The DFLTEXEC parameter specifies whether or not this control set ID is to be used as a default if no set ID is given at the start of execution of the CCP. If the parameter is not given, the default value is NO. If the value YES is given, then this set is used as the default at startup. The last processed SET statement which has this parameter and the value YES specified overrides any previous values.

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 initially allocated 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 dynamic buffer area for communicating with terminals
- Maximum length of commands and program requests
- Number of entries to be allowed in tables for shared files and symbolic file reference
- Amount of core space to be allowed for system trace
- Pack on which the user's DFF formats reside

It is possible to modify only the SYSTEM statement for a set which already exists in the $CCPFILE. If ACTION-SYSMOD was specified on the SET statement, the SYSTEM statement must be the only other statement given for this set. In this case, only those parameters given on this statement are used to modify the parameters that were given when the set was created, replaced, or last modified.

```
// SYSTEM    MINUPA-nn.nnK,MINTPBUF-n [,MAXEUP-{1/n}]
              [,PASSWORD-password] [,PRINTER-{NO/YES}]
              [,MFCU-{NO/YES}] [,RP1442-{NO/YES}]
              [,PGMREQL-{6/n}] [,COMMANDL-{20/n}]
              [,TRACEBLK-{0/1/2/3/4/6/8/12/24}] [,SQB-n] [,FSB-n]
              [,DFFPACK-{SYSTEM/PROGRAM}]
```

MINUPA-nn.nnK

The MINUPA parameter specifies the minimum size of user memory area for executing programs in an operational run of the CCP for this set. The size of user memory must be large enough to include the sum of:

1. The size of the largest group of application programs that must be executed concurrently (if MAXEUP-1 is specified, then simply the size of the largest application program to be executed).
2. If the Display Format Facility (DFF) is used, then while any program which uses DFF is executing:
   a. The size of the DFF control routine.
   b. The size of the DFF output hold area for each line (largest BLKL parameter specified in the TERMATTR statement to be used on each line with DFF3270-YES specified).
   c. For each program which uses DFF, the size of the work space added to a DFF program. This size is determined by the DFFSFDT, DFFMTERM, and DFFNDF parameters in the PROGRAM statement.

The sizes associated with the DFF can be determined from *Appendix F. /Storage Estimates.*

The value for this parameter must end with the letter K. K represents 1024 bytes. Any fractional portion of the value must be one of the following:

.0
.00
.25
.5
.50
.75

Examples of values which may be specified are:

26.25K
26.5K
26.75K
26K
8.0K
8.00K
8.50K

The minimum value allowed is 5K.

MINTPBUF-n

The MINTPBUF parameter specifies the number of bytes which must be set aside for buffer work area for teleprocessing communications.

This is a common work area and is separate from and does not include the buffer for each communications line. The work area is also separate from the record area in the user program. For an example of determining the size of the TP buffer, see index entry *sample assignment set: calculation of core sizes.*

MAXEUP- {1 / n}

The MAXEUP parameter defines the maximum number of concurrently executing user programs which are allowed while operating with this set. The default value if this parameter is omitted is 1; only one user program can execute at a time. The maximum number which can be specified is 8.

However, if a number greater than 1 is given, the validity of that number is checked to ensure that the number of programs given on the SYSTEM statement is less than or equal to the number given in the MAXEUP operand on the $EFAC control statement at the generation of the CCP.

Note: When deciding on this value, ensure that the number is reasonable with respect to the MINUPA and MINTPBUF parameters.

PASSWORD-password

The PASSWORD parameter specifies the system password. This parameter must be specified if the SECURE-CCP option was selected on the $ESEC control statement at generation, and must not be specified if that option was not selected. If the user chooses to use his own security system, then this parameter must be omitted (the security information he checks a sign-on against is set by using the program $CCPAU).

If this parameter is given, the value must contain one to six characters specifying the system password. The password may be any combination of the 64-character set with no embedded blanks.

If the value given for this parameter is a special character, it must be enclosed in apostrophes. (Special characters are any characters other than the 36 alphameric characters.) In order to code an apostrophe as a character in the password, two successive apostrophes must be coded.

PRINTER- {NO / YES}

The PRINTER parameter declares whether or not the printer is allocated to the program level in which the CCP is executing at the beginning of a CCP run. This parameter has meaning only in a DPF system. If the parameter is not given, the default is NO, which means that any program which uses the printer cannot be requested successfully until the system operator allocates the printer to the CCP level. If the YES option is chosen, the printer is available to the CCP program level at the beginning of the CCP run.

MFCU- {NO / YES}

The MFCU parameter declares whether or not the MFCU (Multi-Function Card Unit) is allocated to the program level in which the CCP is executing at the beginning of a CCP run. This parameter has meaning only in a DPF system. If the parameter is not given, the default is NO, which means that any program which uses the MFCU cannot be requested successfully until the system operator allocates the MFCU to the CCP level. If the YES option is chosen, the MFCU is available to the CCP program level at the beginning of the CCP run.

RP1442-{NO / YES}

The RP1442 parameter declares whether or not the 1442 (Card Read Punch) is allocated to the program level in which the CCP is executing at the beginning of a CCP run. This parameter has meaning only in a DPF system. If the parameter is not given, the default is NO, which means that any program which uses the 1442 cannot be requested successfully until the system operator allocates the 1442 to the CCP level. If the YES option is chosen, the 1442 is available to the CCP program level at the beginning of a CCP run.

PGMREQL-{6 / n}

The PGMREQL parameter specifies the length, in bytes, of the longest possible program request command from any terminal used in this set. This length includes program name and the data accompanying the program request if this option is used. The maximum value allowed is 80 positions. If this parameter is not given, the default is six positions. The minimum value allowed is 2 positions.

COMMANDL-{20 / n}

The COMMANDL parameter specifies the length, in bytes, of the longest possible terminal command, excluding the program request command, that can be entered from any command terminal used in this set. The maximum value allowed is 80 characters. If this parameter is not given, the default is 20 characters. The minimum value allowed is zero characters.

TRACEBLK-{0 / 1 / 2 / 3 / 4 / 6 / 8 / 12 / 24}

The TRACEBLK parameter specifies the number of in-core blocks (256 bytes each) reserved at startup for the CCP execution-time trace tables. Each block can contain 16 trace entries. If this parameter is zero or omitted, the CCP trace is not used while the CCP is executing using this set. The value given here can be overriden during the CCP startup. For further discussion, see the TRACE operand on the $EFIL generation control statement, and the TRACE command which may be specified by the system operator (see *IBM System/3 Disk System Communication Control Program System Operator's Guide,* GC21-7581).

SQB-n

The SQB parameter specifies the number of sector queue blocks (SQB) the user desires. Any numeric value from 0 to 255 can be specified. If this parameter is omitted or zero, it indicates that no disk file sharing is permitted.

Sector queue blocks are used to protect data when disk files in update mode are shared. Multiple user programs can update the same file, but an updating program is not allowed to access a particular sector of data until the previous user has finished with the sector containing the data (the sector is written back into the file, a new sector is read back in from the file, or the previous program has terminated). If an SQB is not available when required, the user program requiring the SQB is cancelled.

The value of this parameter must be zero or omitted if MAXEUP-1 was specified at generation time in the $EFAC control statement. It must not be zero if the MAXEUP value was greater than 1 and FSHARE-YES was specified at generation time in the $EFAC control statement.

For a system in which multiple user programs can be executing and update file sharing is allowed, a sufficient number can be calculated by the following formula:

N x MAXEUP = n

where
N = the largest number of update files any user program accesses
MAXEUP = the value of that parameter in this statement

If file sharing is allowed, a SQB is needed for each shared file used in update mode in each concurrently executing program (see *Appendix F. Storage Estimates*).

FSB-n

The FSB parameter specifies the number of file specification blocks (FSB) the user desires. Any numeric value from 0 to 255 can be specified. If this parameter is omitted or zero, it indicates that symbolic file reference is not used.

The FSBs are used to equate symbolic disk file names with physical disk file names.

If a sufficient number of FSBs is not available during execution of the CCP, a terminal may not be able to run a requested program that references symbolic files.

A sufficient number of FSBs can be calculated as follows:

N x CT = n

where

N = the maximum number of reference-name/actual-name/FILE equates required to be in effect for any one command terminal

CT = the number of command terminals

The user can choose to have a lesser or greater number of FSBs than the suggested number.

If any symbolic disk file names are used and indicated on the $EFAC control statement, this value cannot be zero, and the parameter must not be omitted. If SYMFILE-NO was specified in the $EFAC statement, this parameter must be zero or omitted (see *Appendix F. Storage Estimates*).

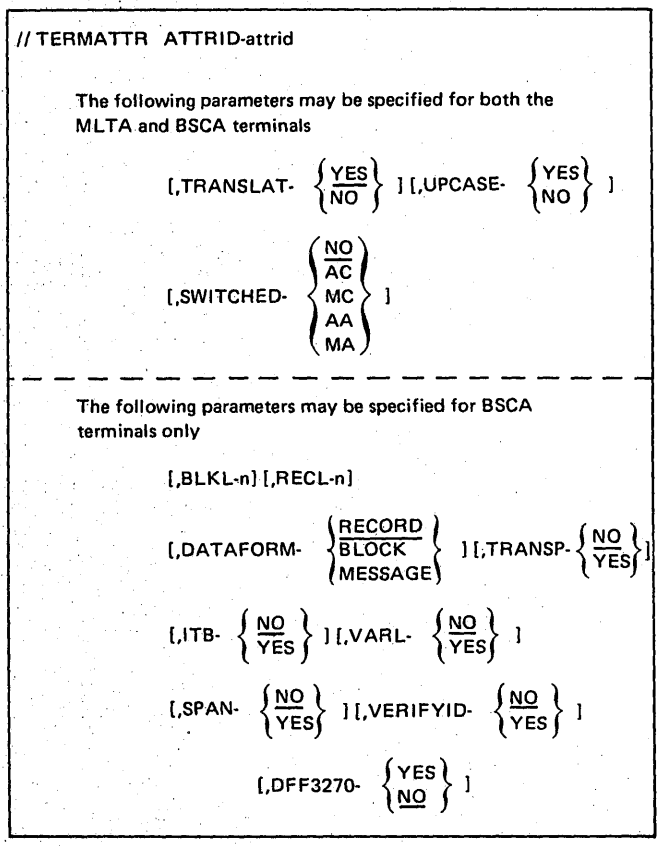
DFFPACK-{SYSTEM / PROGRAM}

The DFFPACK parameter specifies the pack whose object library contains the display formats for this set. The display formats must have previously been produced through the Display Format Generator routine ($CCPDF).

If this parameter is not specified, the use of the Display Format Facility (DFF) will not be allowed for this set. All other parameters referencing the DFF will be diagnosed as invalid.

TERMATTR Statement

Each TERMATTR statement defines certain attributes of a terminal and must follow the SYSTEM statement. The number of these statements required depends on the number and type of terminals used and how they are to perform in this assignment set. Each TERMATTR statement defines one terminal attributes set. One user program can reference different attributes sets for the same terminal depending on the use of that terminal. Each terminal statement (BSCATERM and MLTATERM) must reference a TERMATTR statement. One TERMATTR statement can provide the same attributes for several terminals and therefore be referenced by several terminals.



ATTRID-attrid

The ATTRID parameter must be specified and is a reference ID number used in a BSCATERM or MLTATERM statement to associate this attributes set with a specific terminal. The value for this parameter can be any value between 1 to 255 inclusive. At startup time an amount of core (in bytes) equal to five times the highest number specified by any ATTRID parameter in this set is reserved for the entire terminal attributes table. Thus, the ATTRID values should be chosen from the smallest numbers available.

TRANSLAT-{YES / NO}

The TRANSLAT parameter specifies whether or not the CCP will automatically translate input data from line code to EBCDIC, and output data from EBCDIC to line code. A YES value indicates translation will take place for MLTA transmission, and for BSCA transmission using ASCII code.

A YES value is also valid if the line code is EBCDIC, though it causes no actual translation to occur unless UPCASE-YES is also specified. This would be a consideration when defining attributes for an EBCDIC 3270 terminal with a typewriter keyboard.

TRANSLAT-YES must be specified if using DFF with ASCII 3270 terminals.

A value of NO indicates the CCP will not do the translation; either the data coming in or going out will be EBCDIC or the user will handle the responsibility of translation to and from the appropriate line code.

UPCASE-{YES / NO}

The UPCASE parameter specifies whether or not lower case characters transmitted from the terminal will be converted to upper case characters when presented to the user by the CCP. The YES value is valid only if the TRANSLAT parameter has the value YES. A value of YES indicates the CCP will convert lower case to upper case before presenting data to the user program. A value of NO indicates the CCP will not convert lower case characters to upper case coming from the terminal. If not specified, UPCASE will default to the same value as TRANSLAT.

SWITCHED-{NO / AC / MC / AA / MA}

The SWITCHED parameter specifies the options available on switched lines. This parameter must be specified in an attributes set used with a terminal which is on a switched line. The value AC is not valid for MLTA lines. The default is the value NO. The following explanations apply:

NO – Not a switched line terminal

AC – Auto Call, the CCP calls the terminal using the telephone number provided on the TERMNAME statement

MC – Manual Call, the system operator calls the terminal using the phone number printed by the CCP as provided on the TERMNAME statement

AA – Auto Answer, the CCP answers calls from this terminal

MA – Manual answer, the system operator answers the calls from this terminal

For MLTA, AA and MA are treated the same. For a command mode switched line, AC or MC must not be specified.

BLKL-n

The BLKL parameter defines the block length, in bytes, that is used for this terminal. It can be specified as 1 to 5 numeric digits with a valid range of 1 through 49182, but the actual value is limited by available core. This parameter is required when specifying attributes for any BSCA terminal. The value used here must be equal to RECL times the number of records in a block. Up to 255 records can be contained in one block.

The following are 3270 display formatting considerations:

- If this attributes set will be used with the Display Format Facility (DFF3270-YES), then this parameter will, in addition to the above, be used to calculate the display output hold area size. The display output hold area is an area in core that holds the 3270 data stream for output operations.
- The value for this parameter can vary from 1 to 5120 according to the performance desired. For best performance this size should be large enough to hold the largest output display format that uses this terminal attributes set, the size of which is printed by the Display Format Generation routine. Performance cannot be enhanced by specifying a larger output hold area.
- If the value is less than the largest format, display formats will be broken into output blocks and performance will be decreased. With blocking the minimum required size of the output hold area is 512.
- If two BSCA lines have terminals which use the DFF, two output hold areas are required, one per line. The size of the hold areas may differ in size according to the values specified for this parameter.

RECL-n

The RECL parameter defines the record length, in bytes, that is used for this BSCA terminal. The value must be specified as 1 to 5 numeric digits with a valid range of 1 to the value given for the BLKL parameter. Use of this parameter is valid only if the DATAFORM parameter on this statement has the value RECORD specified.

DATAFORM- { RECORD / BLOCK / MESSAGE }

The DATAFORM parameter defines the format in which the CCP will present terminal input data to the user program.

RECORD indicates the CCP presents a portion of an input block as a complete record to the user program.

BLOCK indicates a complete block (possibly consisting of multiple records) is presented as a unit of data.

MESSAGE indicates that all the data between the STX and the EOT is presented as one unit of data. MESSAGE is required if the DFF3270 parameter on this statement has a value of YES.

The default value is RECORD.

TRANSP- { NO / YES }

The TRANSP parameter specifies whether or not the EBCDIC transparency feature is used in transmission to or from this terminal. The value YES may be used only for a BSCALINE statement with the XMCODE parameter value of EBCDIC (must have the transparency feature installed on the adapter). The value YES is valid only if XPRNCY-YES was specified in the $EBSC generation control statement. The default, if the parameter is omitted, is NO, indicating EBCDIC transparency is not to be used with this terminal.

ITB- { NO / YES }

The ITB parameter specifies whether or not intermediate text blocks are sent or received when using this terminal. The value YES is valid only for BSCA terminals. If ITB is specified as YES (ITB-YES must also be specified on the $EBSC generation control statement) then the parameters VARL and SPAN on this statement must have the value NO. The default value is NO.

VARL- { NO / YES }

The VARL parameter specifies whether or not variable length records with record separator characters are used in data transmission to or from this terminal. The value YES is valid only for BSCA terminals. The value YES is not valid if the ITB parameter on this statement has a value of YES. The value YES is not valid for 3270 terminals. The default is NO.

SPAN- { NO / YES }

The SPAN parameter specifies whether or not input records can span input blocks. The value YES is valid only for BSCA terminal type 3735 or CPU. The value YES is not valid if the ITB parameter on this statement has a value of YES. The default is NO.

VERIFYID- { NO / YES }

The VERIFYID parameter specifies whether or not the CCP verifies the identification bytes sent from this terminal. YES indicates the CCP will verify. NO, the default value, indicates the CCP will not verify the ID of this terminal. The value YES is only valid for BSCA terminals on switched lines.

DFF3270- { YES / NO }

The DFF3270 parameter specifies whether or not the Display Format Facility (DFF) will be used with all terminals referencing this terminal attributes set. This parameter is valid only for 3270 terminals on a BSCA line. If this parameter is specified, the parameter DATAFORM-MESSAGE must be specified in this statement, and the SYSTEM statement in the assignment set must have the DFFPACK parameter specified. The default value is NO.

BSCALINE Statement

The BSCALINE statement defines the type of BSC line to be used and the features of the line. A BSCALINE statement must be followed by at least one BSCATERM statement. The BSCATERM statement defines the terminals which are attached to that line. If more than one communication line or adapter is attached to the System/3, another line statement may be given and follows the last BSCATERM or MLTATERM statement for the previous line. The order in which line statements are entered determines the order of priority during the CCP execution. At least one line statement must be given for each assignment set, either BSCALINE or MLTALINE.

```
// BSCALINE   TYPE-{PP|CS|MP|SW}   [,LINENUM-{1|2}]
              [,XMCODE-{EBCDIC|ASCII}]
              [,POLLLIST-'termid [,termid] . . .']
              [,NRETRY-{7|n}] [,IDEXSEND-exchngid]
              [,POLLLOOP-{256|n}]
              [,DBLBUF-{NO|YES}] [,WAIT-{180|n}]
```

TYPE-{PP|CS|MP|SW}

The TYPE parameter must be given and defines how this line is used.

One of the following values must be specified:

PP – Point-to-Point
Dedicated communication line (leased or private) network on this line between CPUs is to be supported by the CCP; one and only one BSCATERM statement must follow this statement.

CS – Control Station
Multipoint communication line (leased or private) network on this line is to be supported by the CCP; this processor is the control station, 1 to 34 BSCATERM statements must follow this statement, and the POLLLIST parameter on this statement must be used. The maximum number of terminals permitted on a BSCA control station line is 34.

SW – Switched
Telephone connected communication line; one or more terminals may be specified but only one terminal can communicate with the CCP at a time; one or more BSCATERM statements must follow this statement.

MP – Multipoint Tributary Station
Multipoint communication line (leased or private) network on this line is to be supported by the CCP; this System/3 is a tributary to a controlling CPU; one and only one BSCATERM statement must follow this statement and must describe this System/3 as a terminal.

LINENUM-{1|2}

The LINENUM parameter identifies which BSCA line this line statement defines. The numbers 1 and 2 identify BSCA adapter numbers 1 and 2 respectively. If not given, LINENUM defaults to line 1. Two BSCALINE statements, within one set, specifying the same line number are not accepted.

XMCODE-{EBCDIC|ASCII}

The XMCODE parameter defines the transmission code used by the terminals on this line. If not given, XMCODE defaults to EBCDIC. EBCDIC is BSCA EBCDIC code. ASCII is ASCII code.

POLLLIST-'termid [,termid] . . .'

The POLLLIST parameter defines the order in which the terminals attached to this line are to be polled. This parameter is valid only for TYPE-CS lines. The values can be given as a sublist: IDs are separated by a single comma and the entire parameter value is enclosed by apostrophes. Any value within the list must be a *termid* of a terminal specified on a BSCATERM statement. The *termid* given in a POLLLIST parameter in this BSCALINE statement must not appear in the POLLLIST parameter of another BSCALINE statement. A terminal ID can appear more than once in the same POLLLIST parameter. The maximum number of all terminal IDs that can appear in a poll list is 127.

The poll list must be given for all lines with TYPE-CS.

Note: For printers attached to a 3277 control unit, the *termid* of the terminals must appear in the values for POLLLIST even though they are not input devices. The printers will be polled for status only when the CCP detects an error condition.

NRETRY-{7/n}

The NRETRY parameter defines the number of retries which take place if there is a transmission error on the line before an error condition is considered to be uncorrectable. A 1 to 3-digit value up to 256 is acceptable. If the parameter is omitted, the default value of 7 is assumed. If the error persists after the specified number of retries, the error is logged on the system console. If a program is in control of the terminal, then an error return code is returned to the user program. If the terminal is not under the control of a user program, the CCP permits the system operator to bypass or retry the operation.

IDEXSEND-exchngid

The IDEXSEND parameter specifies identification characters this CPU (running CCP) sends to a remote terminal (CPUs or 3735 terminals) to allow the remote terminal to verify that this CPU is the one the remote terminal wants to communicate with. The identification characters can be specified as any 1 to 15 extended alphameric characters or any 2 to 30 hexadecimal characters. The identification characters must be expressed in the transmission code (EBCDIC or ASCII). If the code is ASCII, only the hexadecimal representation can be used. Two hexadecimal characters represent one byte, and therefore there must be an even number of hexadecimal characters specified. The value is identified as being hexadecimal by enclosing the value in asterisks.

Example: IDEXSEND-*C1C2C3C4C5*

This parameter should be specified only if the remote terminal expects to receive ID exchange characters.

This parameter must be given if the terminal(s) on this switched line will be 3735 terminals.

This parameter is valid only if the TYPE parameter value is SW.

POLLLOOP-{256/n}

The POLLLOOP parameter specifies the number of times the poll list should be used before a message is given to the system operator informing him that no terminal has responded with data. Any number from 1 to 256 may be given. The number 256 indicates an infinite number; that is, loop through the list an infinite number of times and do not give a message to the system operator. The default value is 256.

This parameter is valid only for TYPE-CS lines.

DBLBUF-{NO/YES}

The DBLBUF parameter specifies whether or not terminals on this line are double line buffered. YES indicates double buffering will be provided by the CCP; the core will be reserved at startup. NO indicates a single buffer will be provided by the CCP. The default is NO. Double buffering will generally improve data transmission time except, in an interactive environment such as exists with 3270 display stations. This specification refers to the line buffer, not the buffer work area specified by the MINTPBUF parameter on the SYSTEM statement.

WAIT-{180/n}

The WAIT parameter specifies a decimal delay count. The delay count is the number of seconds after receiving or transmitting a block of data that the CCP will wait for the user to receive or transmit another block of data for the same file. The CCP waits the specified number of seconds by using the WACK ENQ and TTD NAK line control sequences.

Except when the end of file has been received or transmitted, the CCP aborts the transmission and posts a completion code if the delay count is exhausted between transmissions.

If a value is not specified, an 180-second delay count is assumed. If a delay count is specified, consider the time that may be required for such items as device errors, halts, and ready I/O devices.

This parameter applies only when user programs are communicating with a terminal on this line.

BSCATERM Statement

The BSCATERM statement defines certain attributes of the terminals on BSCA lines. It also references terminal attributes sets that complete the terminal specifications. The BSCATERM statements must follow the BSCALINE statement for which the terminals are defined.

The number of BSCATERM statements allowed depends on the line type; this is defined by the TYPE parameter on the BSCALINE statement. The maximum number of terminals allowed in a set is 239.

```
// BSCATERM  TERMID-termid,TYPE-termtype,
               ATTRID-'attrid[,attrid] . . .'
             ,COMMAND-{NO|YES}  [,ONLINE-{YES|NO}]
             [,IDEXRCV-exchngid] [,OFFACTN-{HOLD|DROP}]
             [,ADDRCHAR—addressing characters]
             [,POLLCHAR—polling characters]
```

TERMID-termid

The TERMID parameter assigns permanent (within this set) identification characters to the terminal. The valid entries for this keyword are two (exactly two) non-blank extended alphameric characters and are the terminal ID referred to by the system operator. There is no default for this parameter. The value given must be unique for each terminal within this set. The value must not be the reserved ID, $C.

TYPE-termtype

The TYPE parameter specifies the terminal type the BSCATERM statement describes. For all lines, except BSCA switched, all terminals on a line must be of the same type. (All components of a 3270 system are considered to be of the same type.) Switched BSCA lines may have CPUs and 3735 terminals on the same line. CPU indicates all CPUs (S/360, S/370, S/7, S/3) capable of receiving or transmitting over binary synchronous communication lines, with the proper program support. There is no default for this parameter.

The following terminal types may be specified:

3275M1
3277M1
3284M1
3286M1
3275M2
3277M2
3284M2
3286M2
3735
CPU

Note: A BSCATERM statement is not allowed for the 3284 Model 3 attached to a 3275. 3270 type terminals are supported by the CCP on control station lines only. A 3735 type terminal on a nonswitched point-to-point line is supported as being on a control station line.

ATTRID-'attrid [,attrid] . . .

The ATTRID parameter specifies which terminal attributes set (specified by the TERMATTR statement) this terminal will use in the following cases:

- If the *attrid* specification is omitted in the TERMS parameter on the assignment PROGRAM statement.
- If no other specification is given when a user program acquires this terminal during execution.

The values given must correspond with a value given for the ATTRID parameter on a TERMATTR statement. In addition to the specifics given on this statement, the terminal attributes set will supply certain other specifics for this terminal. There is no default for this parameter.

This parameter can be entered as a sublist, referencing the maximum of 32 attributes sets. The first attributes set referenced will be the default used by startup. All the sets referenced will be considered in calculating the maximum line block size for BSCA lines.

It is required that all the attributes sets used with this terminal be given with this keyword.

COMMAND- {NO / YES}

The COMMAND parameter specifies whether or not this terminal is capable of requesting programs. If this parameter is given with the value YES, then the terminal must have both input and output capability. Any terminal can be specified as not being command capable; but, if the parameter is given with the value YES and the terminal does not have both input and output capability, an error diagnostic is given.

YES is invalid for 3735 terminals. There is no default for this parameter.

If command terminals are on a switched line, the attributes set must specify either auto answer or manual answer.

ONLINE- {YES / NO}

The ONLINE parameter is used to specify whether or not the terminal is available to be used at the beginning of a CCP run. The default value for the omitted parameter is YES, indicating it is available.

A terminal may be connected to the system, but temporarily not available because it is in the process of being serviced, or because no authorized person is presently at the terminal to use it. A value of NO indicates that this terminal should be treated by the CCP as if it were logically offline at startup.

This specification can be overridden during the execution of the CCP by the system operator.

IDEXRCV-exchngid

The IDEXRCV parameter specifies the verification characters the System/3 containing the CCP expects to receive when communicating with the proper remote switched station. The value for this parameter can be 1 to 15 extended alphameric characters or 2 to 30 hexadecimal characters. The identification characters must be expressed in the transmission code (EBCDIC or ASCII). If the code is ASCII, only the hexadecimal representation may be used. Two characters represent one byte, therefore there must be an even number of hexadecimal characters specified. The value is identified as being hexadecimal by enclosing the value in asterisks.

Example: IDEXRCV-*C1C2C3C4C5*

This parameter should only be specified if the remote terminal will send ID exchange characters.

OFFACTN- {HOLD / DROP}

The OFFACTN parameter applies only to command terminals (COMMAND-YES) and specifies the action to be taken with this terminal when the terminal operator issues the Sign-off command. The only two acceptable values are HOLD and DROP.

HOLD means the line continues to be monitored by the system after the Sign-off command is given. HOLD is the default value for a nonswitched line.

DROP means the terminal is set in offline status after the Sign-off command is given. The terminal is no longer monitored for input and, if the terminal is connected by a switched line, the line is disconnected. DROP is the default value for a switched line.

The terminal operator can override the parameter specified when using the Sign-off command.

ADDRCHAR-addressing characters

The ADDRCHAR parameter is the hexadecimal character representation of the address characters in the line code to be used with this terminal. The value for this parameter must be 4 to 14 hexadecimal characters in length. The identification characters must be expressed in the transmission code (EBCDIC or ASCII). For a multipoint line (that is, this CPU is a tributary station), this parameter refers to the characters by which this CPU will be addressed from the host CPU. This parameter is not valid for terminals on a switched line or a point-to-point line. The value is identified as being hexadecimal by enclosing the value in asterisks. For a description of the valid characters for System/3, see *IBM System/3 Model 10 Disk System Multiline/Multipoint Binary Synchronous Communications Reference Manual,* GC21-7573. The terminal address and transmission code are physically wired at each terminal. Consult your IBM Customer Engineer for the exact code wired.

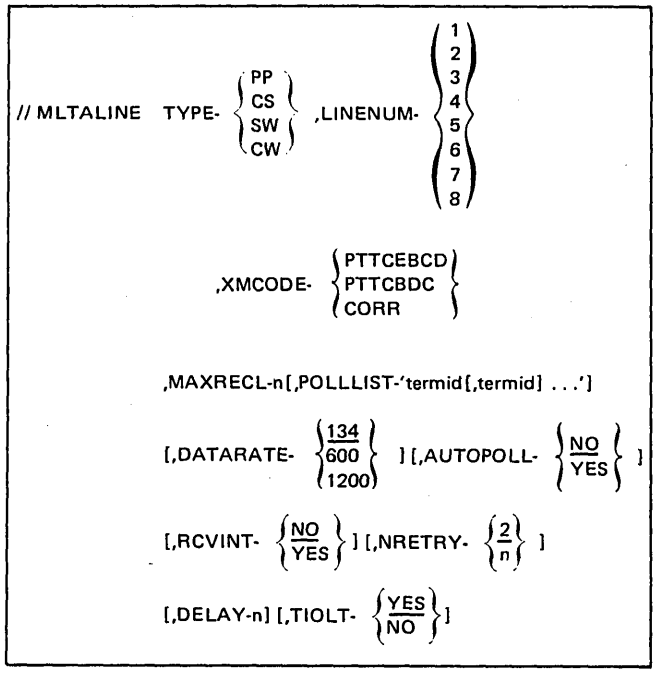
POLLCHAR-polling characters

The POLLCHAR parameter is the hexadecimal character representation of the address characters in the line code to be used with this terminal. The value of this parameter must be 4 to 14 hexadecimal characters in length. The identification characters must be expressed in the transmission code (EBCDIC or ASCII). For multipoint line (that is, this CPU is a tributary station) this parameter refers to the characters by which this CPU will be polled from the host CPU. This parameter is not valid for terminals on a switched line or point-to-point line. The value is identified

as being hexadecimal by enclosing the value in asterisks. For a description of the valid character for System/3, see *IBM System/3 Model 10 Disk System Multiline/Multipoint Binary Synchronous Communication Reference Manual,* GC21-7573. General poll is not supported for 3270 type terminals.

MLTALINE Statement

The MLTALINE statement is used to define the type of communication line to be used and the features of the line. A MLTALINE statement must be followed by at least one MLTATERM statement. The MLTATERM statement defines the terminals which are attached to that line. If more than one communication line or adapter is attached to the System/3, another line statement may be given and will follow the last BSCATERM or MLTATERM statement for the previous line. The order in which line statements are entered determines the order of priority during the CCP execution. At least one line statement must be given for each set, either MLTALINE or BSCALINE.



TYPE- {PP CS SW CW}

The TYPE parameter defines how this line will be used. There is no default.

The following values may be specified:

PP — Point-to-Point
Dedicated communication line (leased or private); one and only one MLTATERM statement must follow this statement.

CS – Control Station
Multipoint communication line (leased or private); this processor is the control station; all terminals on this line must have the Station Control hardware feature regardless of the number of terminals on the line; one or more MLTATERM statements must follow this statement.

SW – Switched
Telephone connected communication line; one and only one MLTATERM statement must follow this statement.

CW – Switched with Control Station Feature
Telephone connected communication line (valid only for lines supporting 1050D terminals); one and only one MLTATERM statement must follow this statement.

LINENUM-{1 2 3 4 5 6 7 8}

The LINENUM parameter identifies which MLTA line this statement defines. The numbers 1 to 8 identify the MLTA line numbers 1 to 8 respectively. There is no default for this parameter. Multiple MLTALINE statements, within one set, specifying the same line number are invalid.

XMCODE-{PTTCEBCD PTTCBCD CORR}

The XMCODE parameter defines the transmission code used on this line. The value given must correspond to the installed hardware feature on the terminal(s) to be used:

PTTCEBCD – Paper Tape Transmission Code EBCD.
PTTCBCD – Paper Tape Transmission Code BCD.
CORR – Correspondence code.

MAXRECL-n

The MAXRECL parameter defines the maximum record length, in bytes, which is used on this line. It must be specified by giving one to three numeric digits. The specified length should not include room for line control characters. However, it does include any terminal device control characters that are used. For example, carriage return and idle characters inserted by the CCP in output to a typewriter terminal are included in the record (16 characters at the beginning and/or end of the message; the 2740 Model 2 with Buffer Receive requires one character at the beginning and/or end of the message). The valid range, for non-command terminals, is 16 to 256 characters. The record length should not be less than the size of the largest record to be sent or received from an application program.

If any terminal on the line is a command terminal and that terminal is a 2740 Model 2, the minimum MAXRECL value is 77. If the command terminal is not a 2740 Model 2, the minimum MAXRECL value is 107.

Note: System operator initiated online tests require a minimum record length of 100 bytes.

POLLLIST-'termid [,termid] . . .'

The POLLLIST parameter defines the order in which the terminals attached to this line are to be polled. The values can be given as a sublist, for example, terminal IDs are separated by a single comma and the entire sublist must be IDs of terminals specified on MLTATERM statements. The IDs given in the POLLLIST parameter must be unique for each line statement. (A particular ID cannot appear in the POLLLIST parameter of more than one MLTALINE statement in a set.) A terminal ID can appear more than once in the same POLLLIST parameter. The maximum total number of terminal IDs that can appear in a POLLLIST parameter is 127. For terminal types 1050 or 1050D, the maximum number is 31.

The poll list must be given for all lines when TYPE-CS is specified on this statement.

DATARATE-{134 600 1200}

The DATARATE parameter specifies the data rate of transmission this line is capable of handling; 134.5, 600, or 1200 bits per second. A data rate 1200 is not available in the U.S.A. This parameter, if omitted, will default to 134 (meaning 134.5 bits per second).

AUTOPOLL- {NO / YES}

The AUTOPOLL parameter specifies whether or not the Autopoll feature is installed in the MLTA. Consult the IBM Customer Engineer to determine if this feature was installed. It may be installed if the 2740 Model 1 and Model 2 terminals have the station control feature, or if any terminals in the 1050 System are used. This parameter must be specified for every line statement if the feature is installed on the MLTA. If the parameter is omitted, the default is NO.

RCVINT- {NO / YES}

The RCVINT parameter specifies whether or not the receive interrupt feature is installed on the terminals. The default value, if the parameter is omitted, is NO; indicating it is not installed. The value YES is valid only for full duplex lines.

NRETRY- {2 / n}

The NRETRY parameter defines the number of retries which will take place if there is a transmission error on the line. A 1 to 3-digit value up to 256 is acceptable. If the parameter is omitted, the default value of 2 is assumed.

If the error persists after the specified number of retries, the error will be logged on the system console. If a program is in control of the terminal, then an error return code is returned to the user program. If the terminal is not under control of a user program, an error message is sent to the system operator.

DELAY-n

The DELAY parameter defines the delay time for control station type MLTA lines after the poll list has been used and before polling will start again This time is given in tenths of a second. If the number 15 were given, this would mean one and a half seconds.

The maximum value which can be given is 256 (25.6 seconds).

If this parameter is not specified, there is no delay.

TIOLT- {YES / NO}

The TIOLT parameter specifies whether or not the terminals on this line will be allowed to request an online test. The default is YES. This parameter does not affect the ability of the system operator to initiate an online test.

NO should be specified only if five consecutive nines (99999) must be entered as data.

Notes:

1. No additional core storage is required to utilize the online test function.
2. Online test requires a minimum record length of 100 bytes.

MLTATERM Statement

The MLTATERM statement defines certain attributes of the terminals on MLTA lines. It will also reference a terminal attributes set that will complete the terminal specifications. The MLTATERM statements must follow the MLTALINE statement for which the terminals are defined.

The number of MLTATERM statements allowed depends on the line type. This is defined by the TYPE parameter on the MLTALINE statement. The maximum number of terminals allowed in a set is 239.

```
// MLTATERM  TERMID-termid,TYPE-termtype,
                ,ATTRID-'attrid [,attrid] . . .' ,COMMAND- {NO|YES}
                [,ADDR-xx] [,ONLINE- {YES|NO} ] [,PINCOMP-n]
                [,POUTCOMP-n] [,OFFACTN- {HOLD|DROP} ]
```

TERMID-termid

The TERMID parameter assigns permanent (for this set) identification to this terminal. The valid entries for this parameter are any two (exactly two) non-blank extended alphameric characters and will be the terminal ID referred to by the system operator. There is no default for this parameter. The value given must be unique for each terminal within this set. The value must not be the reserved ID, $C.

TYPE-termtype

The TYPE parameter specifies the terminal type this MLTATERM statement describes. All terminals on any one MLTA line must be identical; (for example, 2740S, or 2740SC, or 1050). If a 1050 System is being described, refer to the parameters PINCOMP and POUTCOMP on this statement for further component description.

Any one of the following terminal types may be specified:

Character to Specify	Terminal and Features
1050	1050 on a nonswitched line
1050D	1050 on a dial (switched) line
2740	2740 Model 1 without additional features on a nonswitched line
2740C	2740 Model 1 with longitudinal redundancy checking (LRC) feature on a nonswitched line
2740D	2740 Model 1 without additional extra features on a dial (switched) line
2740DC	2740 Model 1 with longitudinal redundancy checking (LRC) feature on a dial (switched) line
2740DT	2740 Model 1 with transmit control feature on a dial (switched) line
2740DTC	2740 Model 1 with transmit control and longitudinal redundancy checking (LRC) features on a dial (switched) line
2740M2S	2740 Model 2 with station control feature on a nonswitched line
2740M2SB	2740 Model 2 with station control and buffer receive features on a nonswitched line
2740M2SC	2740 Model 2 with station control and longitudinal redundancy checking (LRC) features on a nonswitched line
2740M2SCB	2740 Model 2 with station control, longitudinal redundancy checking (LRC), and buffer receive features on a nonswitched line
2740S	2740 Model 1 with station control feature on a nonswitched line

Character to Specify	Terminal and Features
2740SC	2740 Model 1 with station control and longitudinal redundancy checking (LRC) features on a nonswitched line
2741	2741 on a nonswitched line
2741D	2741 on a dial (switched) line
CMCSTD	Communicating Magnetic Card SELECTRIC® Typewriter on a dial (switched) line. The CMCST is supported to the extent that it functions identically to a 2741D
SYS7C	System/7 functioning as a 2740 with longitudinal redundancy checking (LCR) feature on a nonswitched line. The System/7 is supported to the extent that it functions identically to a 2740C
SYS7DC	System/7 functioning as a 2740 with longitudinal redundancy checking (LRC) feature on a dial (switched) line. The System/7 is supported to the extent that it functions identically to a 2740DC
SYS7SC	System/7 functioning as a 2740 with station control and longitudinal redundancy checking (LRD) features on a nonswitched line. The System/7 is supported to the extent that it functions identically to a 2740SC

ATTRID-'attrid [,attrid] . . .'

The ATTRID parameter specifies which terminal attributes set (specified by the TERMATTR statement) this terminal will use in the following cases:

- If the *attrid* specification is omitted in the TERMS parameter on the assignment PROGRAM statement.
- If no other specification is given when a user program acquires this terminal during execution.

The values given must correspond with a value given for the ATTRID parameter on a TERMATTR statement. In addition to the specifics given on this statement, the terminal attributes set will supply certain other specifics for this terminal. There is no default for this parameter.

This parameter can be entered as a sublist, referencing the maximum of 32 attributes sets. The first attributes set referenced will be the default used by startup.

It is required that all the attributes sets used with this terminal be given with this keyword.

COMMAND-{NO / YES}

The COMMAND parameter specifies whether or not this terminal is capable of requesting programs. If this parameter is given with the value YES, then the terminal must have both input and output capability. Any terminal can be specified as not being command capable, but if this parameter is given with the value YES, and the terminal does not have both input and output capability, an error diagnostic is given. There is no default for this parameter.

ADDR-xx

The ADDR parameter is the 2-hexadecimal character representation of the transmission line code of the terminal address. The terminal address and transmission code are physically wired at each terminal. Consult your IBM Customer Engineer for the exact code wired.

ONLINE-{YES / NO}

The ONLINE parameter is used to specify whether or not the terminal is available to be used at the beginning of a CCP run. The default value for the omitted parameter is YES. A value of NO indicates that this terminal should be treated by the CCP as if it were logically offline at startup. This specification may be overridden during the execution of the CCP by the system operator.

PINCOMP-n

The PINCOMP parameter is used only for 1050 terminals, and specifies the principal (or only) input device for a 1050 System specified as a command terminal and indicates the component from which the CCP receives the terminal operator's input commands. For a non-command terminal, either this parameter or the POUTCOMP parameter is required; both may be specified if the terminal has both input and output capability.

The following numbers will apply for the 1050 System:

1 - Keyboard
2 - Reader 1
3 - Reader 2

POUTCOMP-n

The POUTCOMP parameter is used only for 1050 terminals, and specifies the principal (or only) output device of that terminal. This parameter is required for a 1050 System specified as a command terminal and indicates the component to which the CCP will output messages to the terminal operator. For a non-command terminal, either this parameter, or the PINCOMP parameter is required; both may be specified if the terminal has both input and output capabilities.

The following numbers will apply for the 1050 System:

5 - Printer 1
6 - Printer 2
7 - Punch 1
8 - Punch 2

OFFACTN- {HOLD / DROP}

The OFFACTN parameter is used only for command terminals (COMMAND-YES), and specifies the action to be taken with this terminal when the terminal operator issues the Sign-off command. The only two acceptable values are HOLD and DROP.

HOLD means the line will continue to be monitored by the system after the Sign-off command is given. HOLD is the default value for a nonswitched line.

DROP means that, when the Sign-off command is given:

- If the terminal is connected by a nonswitched line, that terminal's status is changed from **online** to **offline.**
- If the terminal is connected by a switched line, the line is disconnected.

DROP is the default for a switched line.

The terminal operator can override the parameter specified when using the Sign-off command.

TERMNAME Statement

The TERMNAME statement defines symbolic names to be associated with terminals and sub-terminals (a sub-terminal is a component of a 1050 system). This statement can also associate a telephone number to a specific terminal connected by a switched line. A maximum of 254 TERMNAME statements can be given for one assignment set.

At assignment time, every terminal must have at least one symbolic name associated with it. If more than one symbolic name is associated to a specific terminal by giving the same TERMID parameter value on more than one TERMNAME statement, the first one given will be called the primary name and those following will be secondary names.

```
// TERMNAME    NAME-termname[,TERMID-termid]
               [,MSTRNAME-termname] [,INCOMP-n]
               [,OUTCOMP-n] [,PHONENUM-number]
```

NAME-termname

The NAME parameter specifies a symbolic name to be associated with either a terminal or an input and/or output sub-terminal. If the parameter TERMID is specified on this statement, this NAME is associated with a specific terminal at the beginning of a CCP run. If the parameter MSTRNAME is specified, this NAME is the name of an input and/or output sub-terminal.

The value for this parameter must be one to six extended alphameric characters; it cannot be all blanks, CONSOL, or ALL. It must be different from any other name given on any other TERMNAME statement.

This symbolic name can be given and not associated with a terminal at assignment time. In this case it could, during a CCP run, be assigned to a terminal by the system operator. Also, if this symbolic name is assigned to a terminal, by using the TERMID parameter on this statement, it can be reassigned to another like terminal by the system operator during a CCP run.

TERMID-termid

The TERMID parameter associates the terminal which has this 2-character ID with the symbolic name given in the NAME parameter. Specifying this parameter implies the NAME parameter value is a terminal name and therefore the parameters INCOMP, OUTCOMP, and MSTRNAME cannot be given on this statement. The *termid* value must be a *termid* given for a terminal on a BSCATERM or MLTATERM statement.

The first TERMNAME statement that has both NAME and TERMID parameters given for a specific terminal is the primary name for this terminal.

MSTRNAME-termname

The MSTRNAME parameter is used to associate the sub-terminal with a terminal symbolic name. This parameter should only be used when describing sub-terminals and cannot be specified if the TERMID parameter is specified. If this parameter is specified, either the INCOMP and/or OUTCOMP parameters must also be specified. The value given here must match a value given as a terminal symbolic name on a previous TERMNAME statement.

When describing a 1050 System, this parameter will associate the sub-terminal specified on this statement to a 1050 System symbolic name.

INCOMP-n

The INCOMP parameter specifies an input sub-terminal (input component of a multi-component terminal) and implies the name given in the NAME parameter will be the symbolic name associated with this sub-terminal. If this parameter is specified, then the MSTRNAME parameter must also be specified to associate this sub-terminal to a terminal symbolic name.

The value for this parameter must be a single digit number. For the 1050 System the following numbers apply:

1 - Keyboard
2 - Reader 1
3 - Reader 2
4 - Any one input component of the polled line

OUTCOMP-n

The OUTCOMP parameter specifies an output sub-terminal (output component of a multi-component terminal) and implies the name given in the NAME parameter is the symbolic name associated with this sub-terminal. If this parameter is specified, then the MSTRNAME parameter must also be specified to associate this sub-terminal to a terminal symbolic name.

The value for this parameter must be a single digit number. For the 1050 System the following numbers apply:

5 - Printer 1
6 - Printer 2
7 - Punch 1
8 - Punch 2
9 - Any or all output components of the polled system

PHONENUM-number

The PHONENUM parameter defines the telephone number for this terminal. The use of a telephone number is required for switched lines using auto call or manual call. The value given must be a string of numeric digits representing the exact telephone number needed to place the call to this terminal. This parameter may be specified only if the TERMID parameter is given on this statement. A maximum of 25 digits can be given for any one telephone number.

DISKFILE Statement

The DISKFILE statement is used to describe the disk files which will be used during the execution of this set. The information given on this statement must correspond with the actual type of file as it exists or will exist on disk.

A DISKFILE statement must be given for each disk file used by any program in this set, specifying the name by which programs reference this file.

The number of DISKFILE statements, plus the number of SYMFILE statements, must not exceed 100 in one assignment set.

```
// DISKFILE    NAME-filename[,DEVICE- {5444} ]
                                      {5445}
                ,ORG- {C} ,RECL-n[,KEYL-n]
                      {D}
                      {I}
                [,KEYPOS-n] [,MSTRINDX- {YES} ]
                                        {NO}
                [,MIXSIZE-n]
```

NAME-filename

The NAME parameter specifies the name by which a program references the file. This name **must** be the same in **every** program in this assignment set that uses this file (except where the file is referenced by a // SYMFILE name). This name matches the name on an OCL // FILE NAME-cccccccc, . . . statement to be included at startup (unless that file is suppressed). As with a single program running under DSM, the name of the actual disk file is specified by the LABEL parameter of the OCL // FILE statement; if the LABEL parameter is omitted in the OCL // FILE statement, then this name is also the name of the actual disk file.

Note that **every** program in this assignment set that references the indicated file must reference it by **this** name.

DEVICE- {5444}
        {5445}

The DEVICE parameter defines the storage device for this file, 5444 or 5445. The default value for an omitted DEVICE parameter is 5444.

ORG- {C / D / I}

The ORG parameter specifies how the file is organized on disk. There is no default for this parameter.

The following values may be specified:

C - Consecutive (sequential)
D - Direct
I - Indexed

RECL-n

The RECL parameter specifies the record length of the file on disk. The record length on disk and the record length for that file in a program must be the same. The value given for this parameter may be from 1 to 4096.

KEYL-n

The KEYL parameter (required for indexed file) specifies the length of the key in an indexed file. The value can be from 1 to 29 but must correspond to the key length as defined in the file on disk.

KEYPOS-n

The KEYPOS parameter (required for indexed file) defines the location of the key within the record. Specifically, the position given must be the position of the first character of the key in the record. This value is not a displacement value. This parameter must be given for all indexed files and may be from 1 to 4096, but must correspond with the value used when the file was created.

MSTRINDX- {YES / NO}

The MSTRINDX parameter specifies whether or not this file should have an in-core master index. The parameter can be omitted and the MIXSIZE parameter can be used to specify a master index to be used and the core allocation for it. Master index is not supported for the 5445 disk files. This parameter is valid only if the ORG parameter has a value of I.

If MSTRINDX-YES is specified and the MIXSIZE parameter is omitted, a master index of one entry per index cylinder is built at startup.

The default is NO, unless MIXSIZE-n is specified with a non-zero value indicating no master index within the CCP. This master index is external to the user's program and will be the only master index used for this file when this program is run under the CCP. Thus, if space for a master index was included in the user program, it will not be used under the CCP.

The master index is a table containing entries for tracks in the index portion of an indexed data file. Each entry contains a track address and the lowest key field associated within that track. The most efficient size for the master index is equal to the key field length plus 2, multiplied by the number of tracks in the file index.

Use of a master index can significantly reduce the amount of time needed to process an indexed file on a 5444.

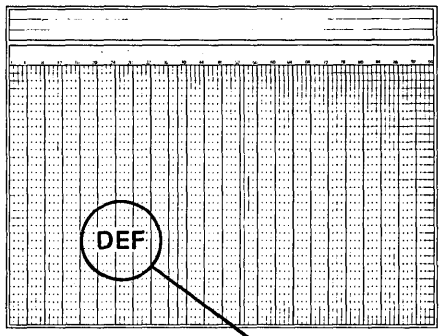
MIXSIZE-n

The MIXSIZE parameter specifies the number of bytes to be allocated to the in-core master index for this file. The user can specify up to 16,383 bytes for the master index. The minimum value that can be specified is the number specified for KEYL plus 5. This parameter is valid only if the ORG parameter has a value of I specified.

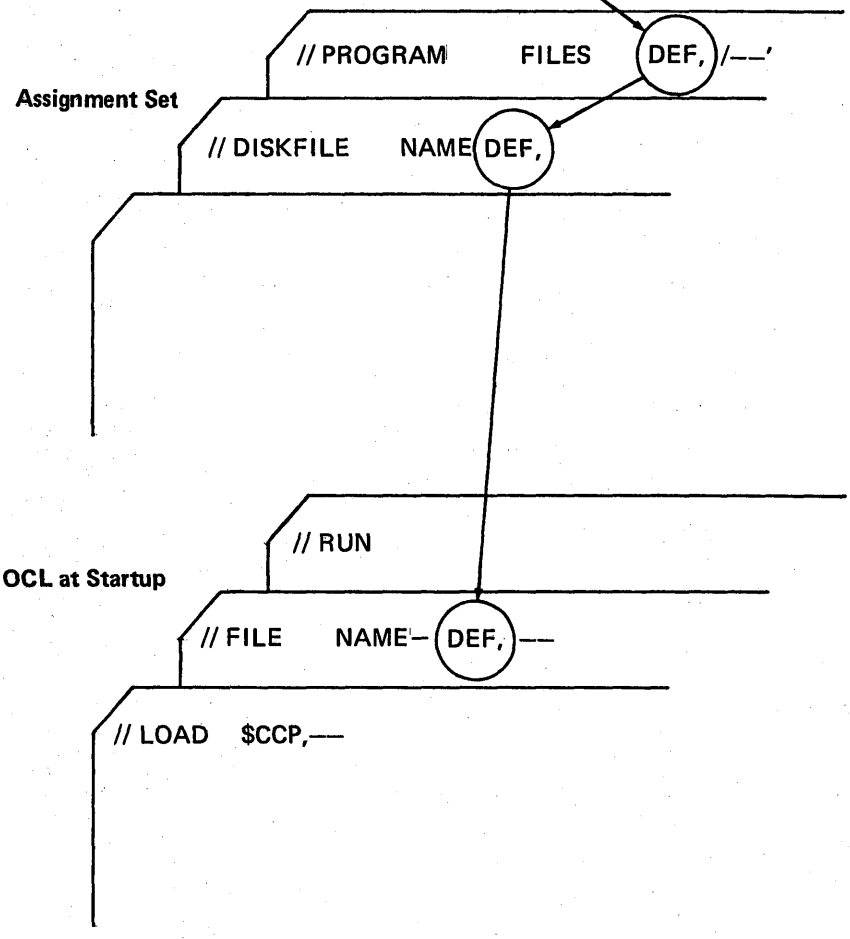
The following chart indicates results of various combinations of the MSTRINDX parameter and the MIXSIZE parameter:

	MIXSIZE Parameter	
	Nonzero	**Omitted**
MSTRINDX Parameter		
YES	Valid, the specified number of bytes will be used for master index	Valid, one entry per cylinder of index
NO	Invalid	Valid, no master index
Omitted	Valid, the specified number of bytes will be used for master index	Valid, no master index

The filename used in the program . . . .

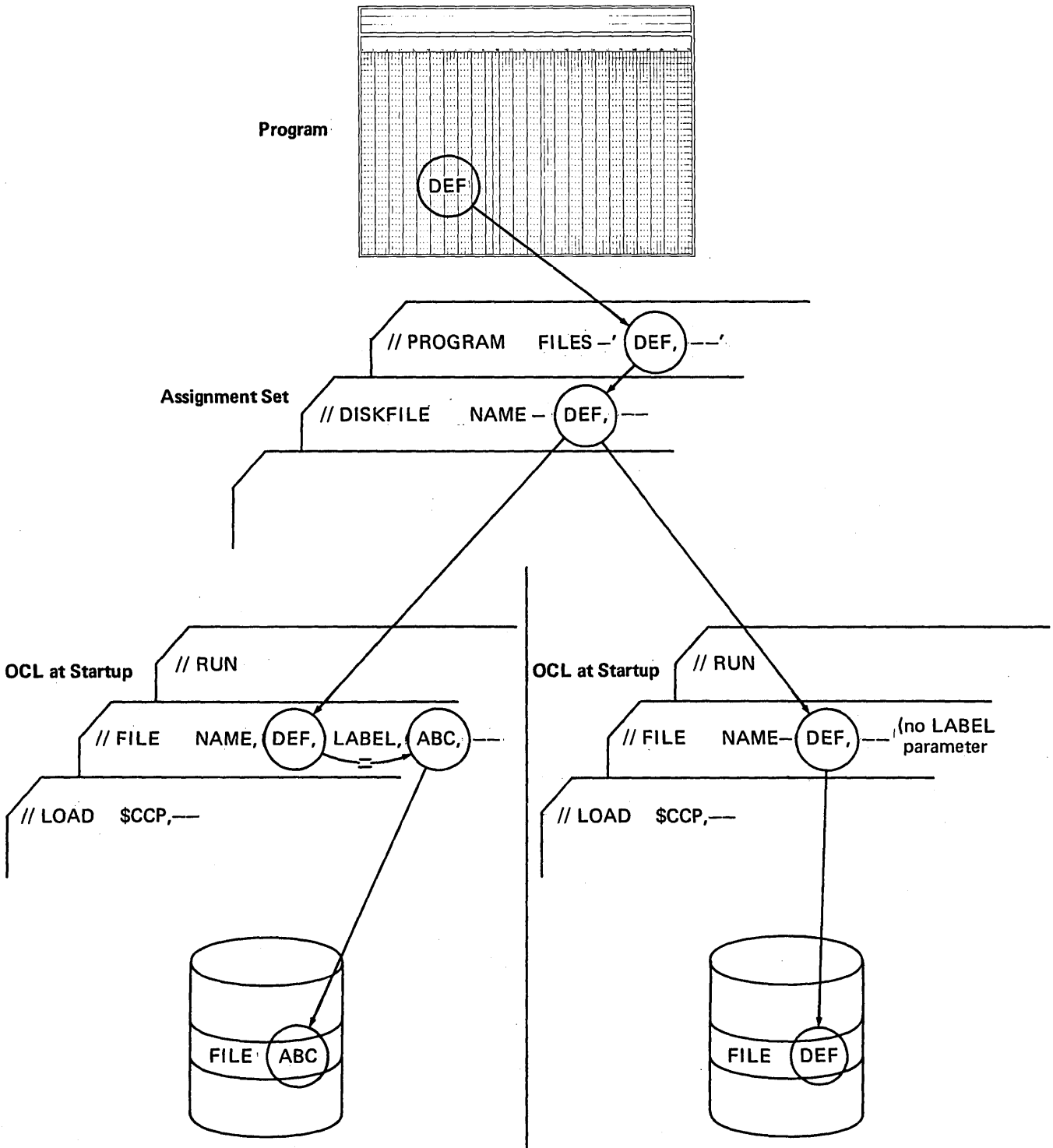


. . . . is the name that must be specified in a DISKFILE statement during assignment . . .



. . . . and is the name that must appear as the NAME parameter in an OCL FILE statement at the startup of CCP.

Relationship of Filename Used in Program to Name of File on Disk



If the LABEL parameter is specified in the OCL FILE statement, the actual file referenced is the file that resides on disk under the name specified by that LABEL parameter.

If no LABEL parameter is specified in the OCL FILE statement, the actual file referenced is the file that resides on disk under the name specified by the NAME parameter.

SYMFILE Statement

The SYMFILE statement defines the symbolic disk file reference name, and specifies the set of disk files with which that symbolic name can be validly associated by the terminal operator /FILE commands (see index entry *file command*).

```
// SYMFILE    NAME-ccccc,DISKFILE-'ccc...[,ccc...] ...'
```

NAME-cccccccc

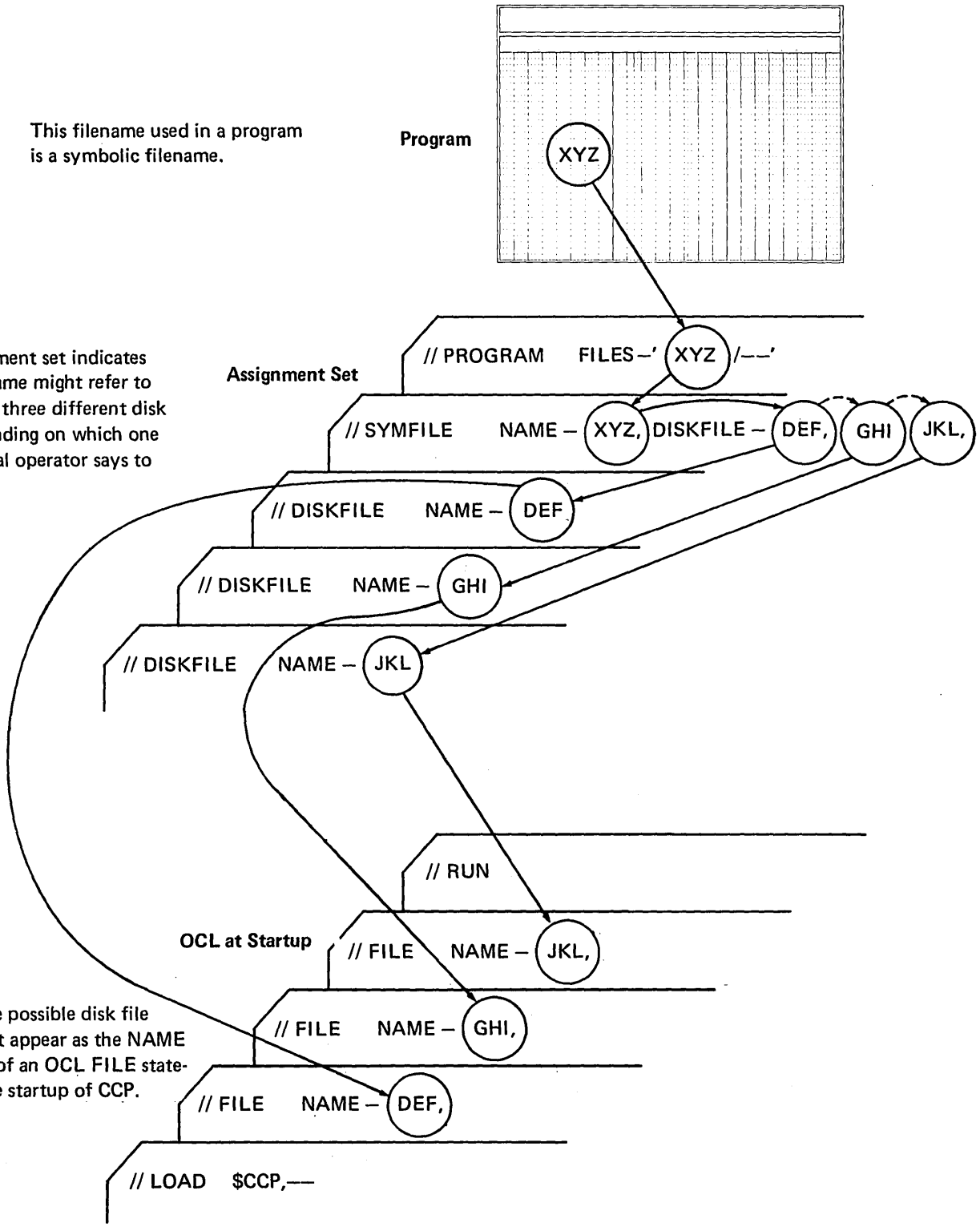
The NAME parameter specifies a symbolic name which will be used as a file reference name within one or more programs in this assignment set. This name does not identify a particular file (as a DISKFILE statement NAME parameter does). A File command from the terminal operator is required to specify which disk file is actually to be referenced in execution of the program. The File command must specify one of the disk files identified in this statement. It cannot duplicate any name on a DISKFILE statement or another SYMFILE statement.

DISKFILE-'ccc . . . [,ccc . . .] . . .'

The DISKFILE parameter specifies one or more file names previously specified by the NAME parameter on a DISKFILE statement. All files referenced in the DISKFILE parameter of the SYMFILE statement must have identical characteristics. A symbolic file name cannot reference both 5444 and 5445 disk files.

Symbolic Disk File Name

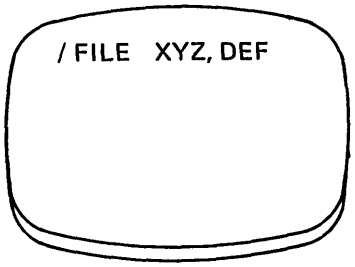
This filename used in a program is a symbolic filename.



The assignment set indicates that this name might refer to any one of three different disk files, depending on which one the terminal operator says to use.

Each of the possible disk file names must appear as the NAME parameter of an OCL FILE statement at the startup of CCP.

Before running the program, the terminal operator entered this command:



When the program is requested by the terminal operator, the program accesses the actual disk file identified by the disk file name DEF.

Because in this OCL, the disk file identified by the name DEF is specified as being on the disk under the name ABC, the file ABC is the actual file referenced in this run of the program.

If the terminal operator had entered:

/FILE XYZ,GHI

the actual file GHI would have been referenced. If the terminal operator had entered:

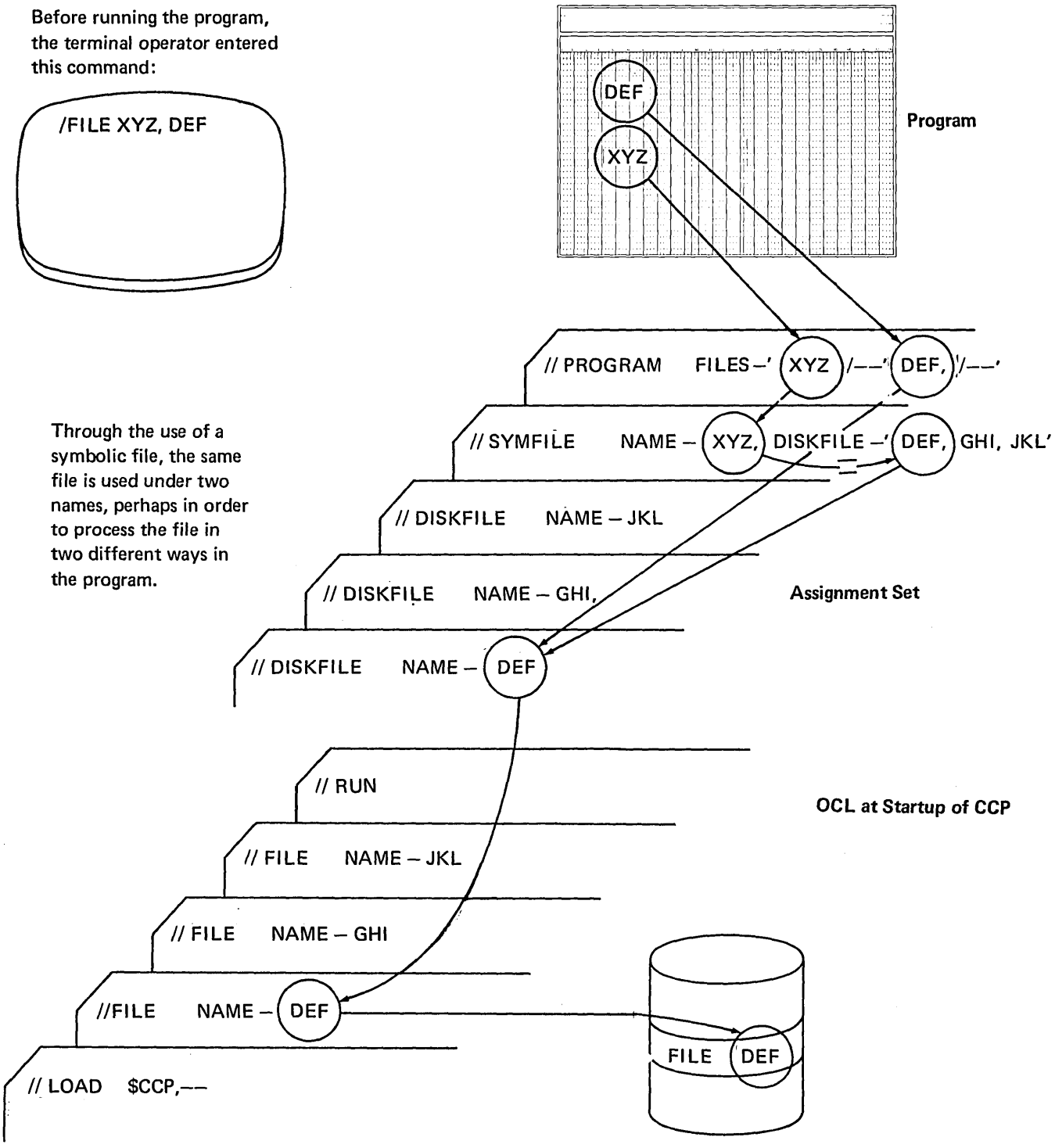
/FILE XYZ,JKL

the actual file MNO would have been referenced.

Two Filenames Used in a Program Refer to the Same File

Before running the program, the terminal operator entered this command:

/FILE XYZ, DEF



Through the use of a symbolic file, the same file is used under two names, perhaps in order to process the file in two different ways in the program.

PROGRAM Statement

The PROGRAM statement defines the logical structure and resource requirements of a user program. One PROGRAM statement must be given for each program which can be requested by a command terminal during the execution of the CCP using this assignment set. A maximum of 255 PROGRAM statements can be entered within one assignment set. For additional discussion about the programming languages supported, see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

```
// PROGRAM   NAME-pgmname,LANGUAGE- { FORTRAN }
                                    { COBOL   }
                                    { RPGII   }
                                    { ASSEM   }
          [,MRTMAX-n] [,REUSABLE- { NO  } ]
                                  { YES }
          [,NEVEREND- { NO  } ] [,RUNALONE- { NO  } ]
                      { YES }               { YES }
          [,PRINTER- { NO  } ] [,MFCU- { NO  } ]
                     { YES }           { YES }
          [,RP1442- { NO  } ] [,PGMDATA- { NO  } ]
                    { YES }              { YES }
          [,ENDMSG- { NO  } ]
                    { YES }
          [,TERMS-'termname[/attrid] [,termname[/attrid]]
            ...']
          [,FILES-'filename/access[/ { SHR   } ],...']
                                     { NOSHR }
          [,SHAREDIO- { NO  } ] [,PACK- { PROGRAM } ]
                      { YES }           { SYSTEM  }
          [,DFFMTERM-n] [,DFFNDF-n] [,DFFSFDT-n]
```

NAME-pgmname

The NAME parameter defines the name of a program which resides on a user's CCP pack or system pack. The name must be two to six extended alphameric characters, except that it cannot start with a $ or a numeric character.

LANGUAGE- { FORTRAN }
          { COBOL   }
          { RPGII   }
          { ASSEM   }

The LANGUAGE parameter specifies the programming language used to compile this program. There is no default for this parameter.

MRTMAX-n

The MRTMAX parameter specifies that the program is capable of servicing multiple requesting terminals, and the number of terminals it is capable of servicing at one time. For a discussion about multiple requesting terminals, see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579. If this parameter is omitted, it indicates that this program (a single requester program) is not capable of servicing multiple requesting terminals. The valid range of numbers for this parameter is 1 to 239.

REUSABLE- { NO }
          { YES }

The REUSABLE parameter specifies whether or not this program is reusable without the necessity of reloading it. RPG II and FORTRAN programs are not reusable programs. A never-ending program cannot be specified REUSABLE. The default value if this parameter is omitted is NO.

NEVEREND- { NO }
          { YES }

The NEVEREND parameter specifies whether or not the program is never ending. The term **never ending** defines a program which, once it is initiated and loaded into memory, will remain in memory communicating either with one or many terminals, or at times with none, until the shutdown of the CCP.

A reusable program cannot be specified as never ending.

RUNALONE- { NO }
          { YES }

The RUNALONE parameter states whether or not this program (when it is running) is required to be the only program executing under control of the CCP in a multiprogramming environment. This parameter applies only to a version of the CCP which was generated to handle more than one concurrently executing user program. This parameter has no effect on the other programming level in a DPF system. The default value is NO.

PRINTER- { NO }
         { YES }

The PRINTER parameter specifies whether or not the line printer is used in the program. The default is NO.

MFCU- {NO / YES}

The MFCU parameter specifies whether or not the MFCU is used in the program. The default is NO.

RP1442- {NO / YES}

The RP1442 parameter specifies whether or not the 1442 is used in the program. The default is NO.

PGMDATA- {NO / YES}

The PGMDATA parameter specifies whether or not a program request from a terminal can have data entered with it. The default value if the parameter is not given is YES. If a terminal operator gives data with the program request, and the value in this parameter is NO, then the terminal operator will receive an error message. A program specified as servicing multiple requesting terminals (parameter MRTMAX) must not specify PGMDATA-NO. If PGMDATA-YES is specified, the requesting terminal is not required to enter data. However, the program must be able to handle zero data length input.

If PGMDATA-NO is specified, any program request containing data is rejected by the CCP.

ENDMSG- {NO / YES}

The ENDMSG parameter specifies whether or not the terminal which requested this program will receive a message from the CCP whenever:

- The program goes to end-of-job
- The program releases the terminal

The default, if this parameter is omitted, is YES, indicating that a message is to be sent at these times. A specification ENDMSG-NO indicates that the following four messages are not to be sent to the requesting terminal:

S03	PROG END – PROCEED
S05	PROG END – SHUTDOWN
S01	PROG REL – PROCEED
S02	PROG REL – SHUTDOWN

Use of this parameter with the value NO may be particularly important if this program is requested by 3270 type terminals. When a requesting terminal goes from data mode (communicating with a program) to command mode (able to communicate with the CCP), the first 160 positions of the screen are cleared and set by the CCP. If the display is to remain intact after the program releases the terminal or goes to end-of-job, ENDMSG-NO must be specified. (The first 160 positions of the display are cleared unless ENDMSG-NO is specified.)

TERMS-'termname[/attrid] [,termname[/attrid]] . . .'

The TERMS parameter specifies terminals which are required to be available when this program is requested to execute. Terminal attributes sets can also be specified for these terminals.

The values for this parameter can be entered as split-values and/or sublists.

The value for **termname** is the symbolic name of a required terminal and must correspond to the NAME parameter on a TERMNAME statement. Each terminal name given for this parameter within one program must be unique. The terminal names do not have to be assigned to a specific terminal ID at assignment time; however, the system operator must assign them prior to the execution of this program. No two terminal names may be assigned to the same terminal ID.

The value for **attrid** specifies a terminal attributes set to be associated with this terminal during execution of this program. If the attrid value is omitted, the first attributes set given on the BSCATERM or MLTATERM statement apply.

The requesting terminal is not normally specified as a required terminal, but if it is, then the above conditions apply.

The maximum number of required terminals allowed within one PROGRAM statement is 80, or less depending on this formula:

$$T + 2F \leqslant 113$$

where

- F = the number of filenames entered in the FILES parameter on this statement
- T = the maximum number of required terminals allowed on this statement

FILES-'filename/access[/ {SHR / NOSHR}]

[,filename/access, . . .]'

The FILES parameter must be specified if disk files are used in this program. The values for each file are given as split-values and if multiple disk files are used, the entire value for the FILES parameter is a sublist. (The TERMS parameter on this statement defines for the maximum number of files that may be specified on the PROGRAM statement.)

The value for **filename** is required for each disk file used and must be a disk file name as specified in the program. Each filename specified here must correspond to the NAME parameter given on a DISKFILE statement or SYMFILE statement.

The value for **access** is required and describes the assumed organization and mode of access of this file as used by this application program. For additional discussion about the value for access, see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

Note: Creating new direct data disk files is not supported under the CCP. Direct files referenced by programs running under the CCP must already exist on disk.

The following terms may be entered for the access value:

CO –	Consecutive output
CG –	Consecutive input
CU –	Consecutive update
CA –	Consecutive add
DG –	Direct input
DGA –	Direct input binary keys
DU –	Direct input and update
DUA –	Direct input and update, binary
IS –	Indexed sequential input only
ISA –	Indexed sequential input and add
ISL –	Indexed sequential input with limits
ISU –	Indexed sequential input and update
ISUL –	Indexed sequential input and update with limits
ISUA –	Indexed sequential input, update, and add
IA –	Indexed add only
IR –	Indexed random input only
IRA –	Indexed random input and add
IRU –	Indexed random input and update
IRUA –	Indexed random input, update, and add
IO –	Ordered indexed load
IOU –	Unordered indexed load
IRANA –	Indexed random add data management access is used to read records, but no records will be added into the file
IRUANA –	Indexed random update and add data management access is used to read records (and possibly update records), but no records will be added into the file

The optional parameter

{SHR / NOSHR} specifies whether or not this file may be shared

with another program while this program is executing. The CCP allows the sharing of disk files in either input or update mode but the sharing of the same record at the same time is prohibited within the CCP. SHR, the default value for input, update, and add files if SQBs were specified in the SYSTEM statement, indicates this file can be shared with another program. NOSHR, the default for load files, or when SQBs were not specified in the SYSTEM statement, indicates this file is not shared with any other program during execution of this program. SHR must not be specified if the access is CO, IO, or IOU.

SHAREDIO- {NO / YES}

The SHAREDIO parameter indicates whether or not this program was compiled specifying shared I/O. A YES value indicates the program does use shared I/O and a value of NO indicates the program does not use shared I/O. The default value is NO. RPG II and FORTRAN are the only program languages which support shared I/O.

PACK- {PROGRAM / SYSTEM}

The PACK parameter specifies the disk pack which contains this program. PROGRAM is the default value and indicates this program resides on the pack $CCP was loaded from. SYSTEM indicates this program resides on the pack from which the user loaded DSM (IPL).

DFFMTERM-n

The DFFMTERM parameter specifies the maximum number of terminals this program will communicate with concurrently using the Display Format Facility (DFF).

A 1 to 3-digit value from 1 through 239 can be specified. If this parameter is omitted it is assumed none of the terminals serviced by this program will use DFF.

If this parameter is specified, the parameters DFFNDF and DFFSFDT must also be specified on this statement.

DFFNDF-n

The DFFNDF parameter specifies the maximum number of display format names that are referenced during any executions of this program.

A 1 to 3-digit value from 1 through 255 can be specified. If this parameter is omited, it is assumed no terminals serviced by this program will make use of DFF.

If this parameter is specified, the parameters DFFMTERM and DFFSFDT must also be specified on this statement.

DFFSFDT-n

The DFFSFDT parameter specifies the size of the largest field descripter table of any display format referenced by this program.

The value for this parameter can be from 1 through 3584 but must correspond to the largest field descripter table used, as printed by the Display Format Generation routine. If this parameter is omitted, it is assumed no terminals serviced by this program will make use of DFF.

If this parameter is given, the parameters DFFMTERM and DFFNDF must also be given on this statement.

ASSIGNMENT LIST PROGRAM

The Assignment List program ($CCPAL) can be executed anytime after the Assignment File Build program ($CC1BF) has been executed. $CCPAL has four purposes:

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 request count for each program in the assignment file, if the program request count option was chosen at generation time. The request counts can be listed either separately for each set or as a total for all sets. The user can clear the request count to zero either after listing the counts or without listing the counts.

3. List the assignment file directory.

4. List the CCP configuration. This is allowed anytime after generation, even before an assignment run.

The CCP Assignment List program ($CCPAL) can be used to list the following assignment file data:

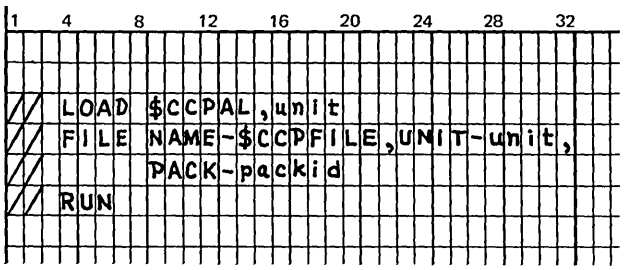
- Assignment File configuration record, directory, or assignment sets.

- An option to list and/or reset the program request-count (if request-count was selected at generation).

The input for the assignment list program is as follows:

- The previously built file $CCPFILE.

- The OCL statement provided by the user. The source statements must be input from the system input device. The following OCL statements are needed:



- The assignment control statements instructing $CCPAL of the function(s) it is to perform.

  A combination of parameters on the // LIST control statement to define the options the user wishes to exercise in listing the file, $CCPFILE.

- The /* statement at the end of all input control statements.

The output for the assignment list program is as follows:

- A user selected listing of portions of the file $CCPFILE.

- A user selected resetting of the program request-count within the file (if request-count was selected at generation).

All listings are output to the printer. The control statements are output to the system log device.

Keyword parameters are listed in the form used by the Assignment Build program control statements.

LIST Statement

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 request count should be printed for one or all of the sets in the file
- Whether the program request count should be reset to zero for one or all of the sets in the file

```
// LIST    [SET- { id     } ] [PGMSTAT- { NO  } ]
                 { ALL    }              { YES }
                 { DIR    }
                 { CONFIG }

        [RESETPS- { NO  } ]
                  { YES }
```

The statement identifier is the word LIST. Any number of LIST statements can be entered for a single execution of $CCPAL. No continuation statements are allowed. This control statement must be given but no parameters are required on the statement.

If no parameters are specified, the statement contains the following information only:

```
// LIST
```

All defaults are assumed, the contents of all the sets in the file are listed, and the program request-count is left untouched.

```
SET- { id     }
     { ALL    }
     { DIR    }
     { CONFIG }
```

The SET parameter specifies which parts of the $CCPFILE to list.

The value *id* refers to any valid assignment set that exists in the control file and causes that set to be printed provided no other parameters are included in this statement. If the parameters PGMSTAT and/or RESETPS is included, the *id* value refers to all request-counts within that set.

The value *ALL*, the default value, causes all assignment sets to be listed provided no other parameters are included in the statement. If the parameters PGMSTAT and/or RESETPS is included, the value *ALL* refers to all request-counts within all assignment sets in the file.

The value *DIR* causes the control file directory (contains the IDs of assignment sets contained in the file) to be printed. If the value SET-DIR is given, no other parameters can be given on this statement.

The value *CONFIG* indicates the CCP configuration record should be printed. If the value SET-CONFIG is given, no other parameters can be given on this statement.

```
PGMSTAT- { NO  }
         { YES }
```

The PGMSTAT parameter states whether or not the program request-counts should be printed on the printer. If the value for the SET parameter is *ALL* or not given, the value YES is given for this parameter, and the option was chosen at generation time, the program request-counts for all assignment sets are printed. If the value given is for a particular valid set ID and the value YES is given for this parameter, the request-counts for just that set are printed. The default, if this parameter is omitted, is NO. If the value YES is given for this parameter, and the system was defined at generation time as not containing request-counts, a diagnostic message is given and this statement is ignored.

```
RESETPS- { NO  }
         { YES }
```

The RESETPS parameter states whether or not the program request-counts should be reset to zero. If the value YES is given and the value *ALL* is given (or defaulted) for the SET parameter on this statement, the program request-counts for all sets in the file are reset to zero.

If the value YES is given and a valid set ID is given for the SET parameter on this statement, the program request-counts are reset to zero for just that set. The default, for the omitted parameter is NO. If the value YES is given with this parameter, and the system was defined at generation time as not containing request-counts, a diagnostic message is given and this statement is ignored.

Combinations of parameters and their associative meaning for the LIST statement follows. Whenever a parameter is omitted, the same effect would occur if the parameter were given with its default value.

// LIST
List all sets in the file

// LIST SET-id
List this particular set

// LIST SET-ALL
List all sets in the file

// LIST SET-DIR
List only the directory

// LIST SET-CONFIG
List only the configuration record

// LIST PGMSTAT-YES
List the program request-counts for all sets

// LIST RESETPS- YES
Reset the program request-counts to zero for all sets

// LIST SET-id,PGMSTAT-YES
List the program request-counts for this set only

// LIST SET-id,RESETPS-YES
Reset the program request-counts for this set only

// LIST PGMSTAT-YES, RESETPS-YES
List the program request-counts for all sets and reset them to zero

//LIST SET-id,PGMSTAT-YES,RESETPS-YES
List the program request-counts for this set and reset them to zero after listing them

SAMPLE ASSIGNMENT SET: OUTPUT FROM THE ASSIGNMENT BUILD PROGRAM ($CCPAS)

```
// LOG PRINTER
// LOAD $CCPAS,R2
// FILE NAME-$CCPWORK,RETAIN-S,TRACKS-3,UNIT-R2,PACK-CCPOBJ
// FILE NAME-$CCPFILE,RETAIN-P,UNIT-R2,PACK-CCPOBJ
// RUN
        // SET ID-0,ACTION-CREATE,DFLTEXEC-YES
        // SYSTEM MINUPA-21.00K,MINTPBUF-2840,MAXEUP-2,
        // PASSWORD-FECD,
        // COMMANDL-50,TRACEBLK-2,SQB-2,FSB-2,OFFPACK-PROGRAM,PGMREQL-15
        *
        *
        // TERMATTR ATTRID-1,TRANSLAT-NO,BLKL-512,DATAFORM-MESSAGE,
        // VERIFYID-NO,OFF3270-YES
        *
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP OR MLTALINE STMNT
        // BSCALINE TYPE-CS,LINENUM-1,POLLLIST-'00,01,10,11'
        // BSCATERM TERMID-00,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
        // ADDRCHAR-*60604040*,POLLCHAR-*40404040*
        // BSCATERM TERMID-01,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
        // ADDRCHAR-*6060C1C1*,POLLCHAR-*4040C1C1*
        // BSCATERM TERMID-10,TYPE-3277M2,ATTRID-1,COMMAND-NO,
        // ADDRCHAR-*61614040*,POLLCHAR-*C1C14040*
        // BSCATERM TERMID-11,TYPE-3277M2,ATTRID-1,COMMAND-NO,
        // ADDRCHAR-*6161C1C1*,POLLCHAR-*C1C1C1C1*
        *
        // TERMNAME NAME-CU0DV0,TERMID-00
        // TERMNAME NAME-CU0DV1,TERMID-01
        // TERMNAME NAME-CU1DV0,TERMID-10
        // TERMNAME NAME-CU1DV1,TERMID-11
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP
        // DISKFILE NAME-CGIVFIL1,DEVICE-5444,ORG-C,RECL-16
        *
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP
        // DISKFILE NAME-CGIVFIL2,DEVICE-5444,ORG-C,RECL-16
        *********NOTE THAT ONE DISKFILE STATEMENT -CGIVFILE- WOULD BE NEEDED
        **********IF SYMBOLIC FILES ARE NOT BEING USED.
        *
        *
        // DISKFILE NAME-DUMMY1,DEVICE-5444,ORG-D,RECL-256
        // DISKFILE NAME-DUMMY2,DEVICE-5444,ORG-I,RECL-64,KEYL-8,KEYPOS-1,
        // MSTRINDX-YES
        *********THIS STMNT TYPE REQD FOR CCPIVP IF SYMBOLIC FILES ARE USED.
        // SYMFILE NAME-CGIVFILE,DISKFILE-'CGIVFIL1,CGIVFIL2'
        *
        *
        *************THIS STMNT NECESSARY FOR CCPIVP,PACK AND PRINTER VALUES
        **************CAN BE CHANGED FOR YOUR CONFIG.
        // PROGRAM NAME-CCPIVP,LANGUAGE-ASSEM,PGMDATA-YES,
        // FILES-'CGIVFILE/CO/NOSHR',
        // PACK-PROGRAM,PRINTER-YES
        *
        *
        *********NOTE THAT CCPIVP MUST BE ON CORRECT PACK AT STARTUP OF CCP.
        ********* IF THE PRINTER IS TO BE USED CCPIVR MUST BE LINK EDITED FIRST
        *********AS CCPIVP.  CCPIVP MUST BE ON CORRECT PACK AT STARTUP.*******
        *
        // PROGRAM NAME-DUMMY1,LANGUAGE-RPGII,MRTMAX-2,PGMDATA-YES,
        // FILES-'DUMMY1/DU/SHR,DUMMY2/IRUA/SHR',PACK-SYSTEM,OFFMTERM-4,
```

```
  // DFFNDF-2,DFFSFDT-1006
  // PROGRAM NAME-DUMMY2,LANGUAGE-COBOL,MRTMAX-2,PGMDATA-YES,
  // FILES-'DUMMY1/DU/SHR,DUMMY2/IRANA/SHR',PACK-SYSTEM,DFFMTERM-2,
  // DFFNDF-1,DFFSFDT-396
  */*  REPLACE WITH /*
  /*

0       WARNING MESSAGES

0       TERMINATION MESSAGES
```

SAMPLE ASSIGNMENT SET: OUTPUT FROM THE ASSIGNMENT LIST PROGRAM ($CCPAL)

```
                $CCPFILE DIRECTORY LIST                        05/31/73

NUMBER OF SETS IN THIS CCP FILE                                       5

NUMBER OF ENTRIES AVAILABLE FOR SETS                                 20

DATA OBTAINED FROM LAST CCP RUN USING THIS FILE

     MEMORY SIZE OF OBJECT CPU                                   65536

     MAXIMUM NUMBER OF CORE DUMPS IN THIS CCP FILE                   4

     START OF MEMORY DUMPAREA  'C/S'                            180/00

     END OF MEMORY DUMP AREA  START OF TRACE  'C/S'             202/00

     END OF TRACE AREA 'C/S'                                    203/00

EXECUTION TIME - DEFAULT SET ID                                       0

DATE FILE LAST UPDATED                                         05/31/73

SET ID IN THE FILE                    LENGTH OF SET IN SECTORS
        B                                          52
        @                                           7
        0                                          10
        1                                           8
        2                                          10
```

```
                    $CCPFILE CONFIGURATION          05/31/73

MAXIMUM NUMBER OF CONCURRENT USER PROGRAMS   8
SIGN-ON SECURITY - CCP
I/O DEVICE SUPPORT   R1  F1  R2  F2  D1  D2  MFCU  RPI442  5203
PROGRAMMING LANGUAGE SUPPORT   COBOL  FORTRAN  ASSEMBLER  RPGII
DPF SYSTEM
UPDATE FILE SHARING
SYMBOLIC FILES
PROGRAM REQUEST-COUNT STATISTICS KEPT
DATA MODE ESCAPE CHARACTERS   //////
NUMBER OF MLTA LINES AVAILABLE TO CCP   8
MLTA TRANSMISSION CODES   CORRESPONDENCE  PTTCEBCD
MLTA TERMINAL SUPPORT   1050  1050D  2740  2740S  2740C  2740SC
     2740D  2740DT  2740DC  2740DTC  2740M2S  2740M2SB  2740M2SC
     2740M2SCB  2741  2741D  SYS7C  SYS7DC
NUMBER OF BSCA LINES AVAILABLE RO CCP   2
BSCA LINE-TYPE SUPPORT   POINT-TO-POINT  MULTIPOINT  CONTROL-STATION
     SWITCHED
BSCA FACILITIES   RECORD-SEPARATOR-X'1E'  GET-MESSAGE  ITB  EBCDIC
     AUTO-RESPONSE  RESIDENT-POLLING  BSCA-TRANSP
BSCA DEVICE-TYPE SUPPORT   3277M1  3275M2  3277M2  3286M2  3735  CPU
```

SET-a $CCPFILE SYSTEM INFORMATION TABLE 05/31/73

MAXEUP	MINTPBUF	PRINTER	RP1442	SQB	PGMREQL	DFFPACK
MINUPA	PASSWORD	MFCU	TRACEBLK	FSB	COMMANDL	
2	21.00K	2840	FECD	NO	NO	NO	2	2	2	15	50	PROGRAM

SET-a $CCPFILE TERMINAL ATTRIBUTES TABLE 05/31/73

ATTRID	UPCASE	DATAFORM	BLKL	TRANSP	SPAN	DFF3270
TRANSLAT	SWITCHED	RECL	ITB	VERIFYID	VARL	
1	NO	NO	NO	MESSAGE	512	NO	NO	NO	NO	NO	YES

SET-a $CCPFILE LINE CONTROL TABLE 05/31/73

LINENUM	DATARATE	RCVINT	DBLBUF	NRETRY	CALC BSCA	WAIT
TYPE	AUTOPOLL	TIOLT	XMCODE	BLOCK SIZE	RECL	
BSCA-1	CS	NO	EBCDIC	7	554	180

LINENUM	IDEXSEND	POLLLIST
DELAY/POLLLOOP		
BSCA-1	256	00,01,10,11

SET-a $CCPFILE TERMINAL USED TABLE 05/31/73

LINE	TYPE	ONLINE	ATTRID	PINCOMP	ADDRCHAR	POLLCHAR
TERMID	COMMAND	OFFACTN	ADDR	POUTCOMP	IDEXRCV	
B1 00	3277/84/86-M2	YES	YES	HOLD	1	60604040	40404040
B1 01	3277/84/86-M2	YES	YES	HOLD	1	6060C1C1	4040C1C1
B1 10	3277/84/86-M2	NO	YES		1	61614040	C1C14040
B1 11	3277/84/86-M2	NO	YES		1	6161C1C1	C1C1C1C1

SET-a $CCPFILE TERMINAL NAME TABLE 05/31/73

NAME	TERMID	MSTRNAME	INCOMP	OUTCOMP	ID	PHONENUM
CU0DV0	00					
CU0DV1	01					
CU1DV0	10					
CU1DV1	11					

```
SET-a                    $CCPFILE FILE CONTROL TABLE                    05/31/73

FILENAME      DEVICE     ORG      RECL      KEYL     KEYPOS    MSTRINDX   MIXSIZ

CGIVFIL1        5444       C        16
CGIVFIL2        5444       C        16
DUMMY1          5444       D       256
DUMMY2          5444       I        64         8          1         YES         0

SYMBOLIC NAME  DISKFILE NAME/S

CGIVFILE       CGIVFIL1  CGIVFIL2
```

```
SET-a                              $CCPFILE PROGRAM CONTROL TABLE                                        05/31/73

 PROG          RUNALONE     NEVEREND      MFCU        SHAREDIO     PGMDATA      DFFSFDT          PACK
 NAME  LANGUAGE      MRTMAX      PRINTER      RP1442      REUSABLE     ENDMSG          DFFWKA

CCPIVP  ASSEM     NO          NO     YES     NO     NO     NO     NO     YES    YES                      PROGRAM
DUMMY1  RPGII     NO     2    NO     NO      NO     NO     NO     NO     YES    YES    1024     1536     SYSTEM
DUMMY2  COBOL     NO     2    NO     NO      NO     NO     NO     NO     YES    YES     512      768     SYSTEM

     PROG      FILES                                TERMS
     NAME          FILENAME    ACCESS      SHR       TERMNAME  ATTRID

    CCPIVP         CGIVFILE    LOAD        NO
    DUMMY1         DUMMY1      UPDATE      YES
                   DUMMY2      ADD         YES
    DUMMY2         DUMMY1      UPDATE      YES
                   DUMMY2      REF         YES
/*
```

SAMPLE ASSIGNMENT SET: CALCULATION OF CORE SIZES

```
// LOG PRINTER
// LOAD $CCPAS,R2
// FILE NAME-$CCPWORK,RETAIN-S,TRACKS-3,UNIT-R2,PACK-CCPOBJ
// FILE NAME-$CCPFILE,RETAIN-P,UNIT-R2,PACK-CCPOBJ
// RUN
        // SET ID-3,ACTION-CREATE,DFLTEXEC-YES
        // SYSTEM MINUPA-21.00K,MINTPBUF-2840,MAXEUP-2,
        // PASSWORD-FECD, 14              15
        // COMMANDL-50,TRACEBLK-2,SQB-2,FSB-2,DFFPACK-PROGRAM,PGMREQL-15
        *          2                  3     4                              1
        *
        // TERMATTR ATTRID-1,TRANSLAT-NO,BLKL-512,DATAFORM-MESSAGE,
        // VERIFYID-NO,DFF3270-YES          13 12
        *
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP OR MLTALINE STMNT
        // BSCALINE TYPE-CS,LINENUM-1,POLLLIST-'00,01,10,11'
        // BSCATERM TERMID-00,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
        // ADDRCHAR-*60604040*,POLLCHAR-*40404040*
        // BSCATERM TERMID-01,TYPE-3277M2,ATTRID-1,COMMAND-YES,OFFACTN-HOLD,
        // ADDRCHAR-*6060C1C1*,POLLCHAR-*4040C1C1*
        // BSCATERM TERMID-10,TYPE-3277M2,ATTRID-1,COMMAND-NO,
        // ADDRCHAR-*61614040*,POLLCHAR-*C1C14040*
        // BSCATERM TERMID-11,TYPE-3277M2,ATTRID-1,COMMAND-NO,
        // ADDRCHAR-*6161C1C1*,POLLCHAR-*C1C1C1C1*
        *
        // TERMNAME NAME-CU0DV0,TERMID-00
        // TERMNAME NAME-CU0DV1,TERMID-01
        // TERMNAME NAME-CU1DV0,TERMID-10
        // TERMNAME NAME-CU1DV1,TERMID-11
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP
        // DISKFILE NAME-CGIVFIL1,DEVICE-5444,ORG-C,RECL-16
        *
        *
        *********THIS STMNT TYPE REQD FOR CCPIVP
        // DISKFILE NAME-CGIVFIL2,DEVICE-5444,ORG-C,RECL-16
        *********NOTE THAT ONE DISKFILE STATEMENT -CGIVFILE- WOULD BE NEEDED
        *********IF SYMBOLIC FILES ARE NOT BEING USED.
        *
        *
        // DISKFILE NAME-DUMMY1,DEVICE-5444,ORG-D,RECL-256
        // DISKFILE NAME-DUMMY2,DEVICE-5444,ORG-I,RECL-64,KEYL-8,KEYPOS-1,
        // MSTRINDX-YES
        *********THIS STMNT TYPE REQD FOR CCPIVP IF SYMBOLIC FILES ARE USED.
        // SYMFILE NAME-CGIVFILE,DISKFILE-'CGIVFIL1,CGIVFIL2'
        *
        *
        **********THIS STMNT NECESSARY FOR CCPIVP,PACK AND PRINTER VALUES
        **********CAN BE CHANGED FOR YOUR CONFIG.
        // PROGRAM NAME-CCPIVP,LANGUAGE-ASSEM,PGMDATA-YES,
        // FILES-'CGIVFILE/CO/NOSHR',
        // PACK-PROGRAM,PRINTER-YES
        *
        *
        *********NOTE THAT CCPIVP MUST BE ON CORRECT PACK AT STARTUP OF CCP.
        ********* IF THE PRINTER IS TO BE USED CCPIVR MUST BE LINK EDITED FIRST
        *********AS CCPIVP.  CCPIVP MUST BE ON CORRECT PACK AT STARTUP.*******
        *
        // PROGRAM NAME-DUMMY1,LANGUAGE-RPGII,MRTMAX-2,PGMDATA-YES,
        // FILES-'DUMMY1/DU/SHR,DUMMY2/IRUA/SHR',PACK-SYSTEM,DFFMTERM-4,
                                                                    5
```

```
9// DFFNDF-2,DFFSFDT-1006 7
  // PROGRAM NAME-DUMMY2,LANGUAGE-COBOL,MRTMAX-2,PGMDATA-YES,
  // FILES-'DUMMY1/DU/SHR,DUMMY2/IRANA/SHR',PACK-SYSTEM,DFFMTERM-2,
10// DFFNDF-1,DFFSFDT-396 8                                          6
  */*  REPLACE WITH /*
  /*

0       WARNING MESSAGES

0       TERMINATION MESSAGES
```

1 Program request with no data to be keyed in for DUMMY1 and DUMMY2: DUMMY1, DUMMY2 = 6 + 0, program request, data for CCPIVP = 6 + 8 = 14. Therefore the value of the PGMREQL is 14.

2 The maximum terminal command length is 50 positions.

3 Only two user programs may run at one time. It is possible to specify three programs, but only two of them can run at one time and the worse case if the two programs (DUMMY1 and DUMMY2) using the shared direct file in update mode. Therefore the value of SQB is 2.

4 The CCPIVP program is the only program using symbolic files, and it uses only one symbolic file. Since only two copies of CCPIVP can run concurrently in this system, two File commands can potentially be in the system at one time. Therefore the value of FSB is 2.

5 DUMMY1 communicates to two requesting terminals and when loaded, issues acquires to two additional terminals. DUMMY1 may run only with one of the terminals at a time, but it can run with up to four terminals at the same time. (Reference 11)

6 DUMMY2 communicates with up to two requesting terminals at any one time. (Reference 11)

7 1006 is the largest field descriptor table used by this program (DUMMY1 uses format $ZOREN and $Z0009). (Reference 11)

8 396 is the size of the only format used by this program (DUMMY2 uses only format $Z0009). (Reference 11)

9 DUMMY1 uses two data formats. (Reference 11)

10 DUMMY2 uses one data format. (Reference 11)

11 Use 5 7 9 in the formula (127 + [DFFMTERM x 37] + [DFFNDF x 18] + [DFFSFDT rounded up to the next multiple of 256]) rounded up to the next multiple of 256 for additional core to be added to the size of when it is executing under the CCP. For example: 127 + 4 x 37 + 2 x 18 + 1024 (1006 rounded up to the next multiple of 256) = 1536 (1335 rounded up to the next multiple of 256). Refer to the following printout of a $CCPDF run:

$ZOREN DISPLAY FORMAT INFORMATION

EXECUTION-TIME-DATA OUTPUT RECORD AREA FORMAT

FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION
$ZOREN	006	0006									

INPUT RECORD AREA FORMAT

FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION	FIELD NAME	FIELD LENGTH	END POSITION
- AID-	001	0001	CUSNL	006	0007	SNAME	022	0029	SADDR1	022	0051
SADDR2	022	0073	SADDR3	022	0095	QTY1	004	0099	ITEMN1	008	0107
QTY2	004	0111	ITEMN2	008	0119	QTY3	004	0123	ITEMN3	008	0131
QTY4	004	0135	ITEMN4	008	0143	QTY5	004	0147	ITEMN5	008	0155
QTY6	004	0159	ITEMN6	008	0167	QTY7	004	0171	ITEMN7	008	0179

INFORMATION FOR USE DURING CCP ASSIGNMENT STAGE

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 1006
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 1080
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0240

Use 6 8 10 in the above formula to determine additional core to be added to the size of DUMMY2. For example: 135 + 2 x 37 + 1 x 14 + 512 (396 rounded up to the next multiple of 256) = 768 (735 rounded up to the next multiple of 256). Refer to the following printout of a $CCPDF run:

```
                              $Z0009 DISPLAY FORMAT INFORMATION


   EXECUTION-TIME-DATA OUTPUT RECORD AREA FORMAT

FIELD  FIELD   END      FIELD  FIELD   END      FIELD  FIELD   END      FIELD  FIELD   END
NAME   LENGTH POSITION  NAME   LENGTH POSITION  NAME   LENGTH POSITION  NAME   LENGTH POSITION
$Z0009 006    0006


   INPUT RECORD AREA FORMAT

FIELD  FIELD   END      FIELD  FIELD   END      FIELD  FIELD   END      FIELD  FIELD   END
NAME   LENGTH POSITION  NAME   LENGTH POSITION  NAME   LENGTH POSITION  NAME   LENGTH POSITION
- AID- 001    0001     NAME@  020    0021     NAME#  020    0041     FIRST# 038    0079
SECON@ 038    0117     THIRD# 038    0155     FORTH@ 038    0193     FIFTH# 038    0231
SIXTH@ 038    0269     ONE    044    0313     TWO    044    0357     THREE  044    0401
FOUR   044    0445     FIVE   044    0489     SIX    044    0533


   INFORMATION FOR USE DURING ASSIGNMENT STAGE

THE DECIMAL LENGTH OF THE FIELD DESCRIPTOR TABLE IS 0396
THE DECIMAL LENGTH OF THE OUTPUT TEXT IS 0457
THE DECIMAL LENGTH OF THE INPUT TEXT IS 0579
```

12 For this line the output hold area could be the maximum of 1280 (1078, length of output from $ZOREN, rounded up to the next multiple of 256) and 512 (457, length of output from $Z0009, rounded up to the next multiple of 256) for full performance. But in order to conserve core storage, 512 is used. For output text lengths above 256, 512 is the smallest output hold area supported by the Display Format Facility (DFF). The size of the output hold area is specified in the BLKL parameter value.

Note: When an output hold area less than the output text length is encountered by DFF, DFF sends the output in segments using the output hold area.

13 This value is used by $CCPAS to calculate the line buffer size as well as the output hold area. The output from $CCPAL indicates the actual line buffer size.

14 Based on the fact that only two tasks (user programs) can run concurrently and that the largest programs (DUMMY1 and DUMMY2) are multiple request programs (cannot have multiple copies in core concurrently). The worse case for two concurrent user programs is calculated for the following:

1. CCPIVP core size is 4028 bytes (no DFF).

2. DUMMY1 core size is 8448 (8193 rounded up to the next multiple of 256) (compiler output, including linkage edit) + 1536 (for DFF program control information) or 9984.

3. DUMMY2 core size is 6144 (6140 rounded up to the next multiple of 256) (compiler output, including linkage edit) + 768 (for DFF program control information) or 6912.

4. Using the worse case of DUMMY1 and DUMMY2 in core concurrently, the value of the MINUPA parameter would be 9984 + 6912 = 16896 (16.50K).

Note: If the operating environment is such that DUMMY1 and DUMMY2 would never run concurrently, the value of parameter MINUPA could be made less (DUMMY1 and CCPIVP run concurrently, DUMMY2 and CCPIVP run concurrently).

Since the DFF control routine also requires space in the user program area, additional core must be added to the 16.50K. The control routine occupies 4096 bytes. The DFF output hold areas also occupy space. In this example with one line, the DFF output hold area equals 512 bytes, therefore 4608 bytes must be added to the 16896 value to obtain 21504 (or 21.00K rounded up to the next multiple of 256).

15 Analyzing the program REQL and COMMANDL parameter values and the user programs (CCPIVP, DUMMY1, and DUMMY2), the following statements are made:

1. The teleprocessing buffer area is calculated using the following formula:

   Buffer Area = 1.2 x ([TC x L] + [TN x D])

   where
   - L = Maximum command length specified
   - D = Maximum length for Invite Input and/or Get operations issued by the user programs
   - TC (number of command terminals plus console) = 5
   - L = 50 (maximum command length specified – COMMAND-50)

   When no programs are running, the MINTPBUF parameter must be large enough to handle 250 bytes (four terminals on one line and the console [5 x 50]).
   - TN (total number of terminals, not including the console) = 4
   - D = 579 (largest input text listed for the formats to be used).

2. Based on the analysis of the worse possible case when all terminals are in control of a program (CCPIVP, DUMMY1, or DUMMY2), the worse case is when DUMMY1 is running and is communicating to all four terminals and the console is being used for CCP commands. (The maximum input for CCPIVP is 8 characters.)

3. Buffer Area = 1.2 x ([1 x 50] + [4 x 579])
   = 1.2 x (50 + 2316)
   = 2839.2
   = 2840 is the value for the MINTPBUF parameter (this value if for maximum performance).

Notes:

- If two lines were being used the consideration for *worse case* in this sample environment (number 2 above) would add the fact that DUMMY2 could be running concurrently with DUMMY1 and there would be additional terminals on line 2.

- If the value of the MINTPBUF parameter is less than 2840, the system may run satisfactorily if timing is such that the largest input format (579) is not in use at all four terminals concurrently.

USER SECURITY DATA PROGRAM ($CCPAU)

The intention of the User Security Data program ($CCPAU) is to take input from the system input device, process it, and, if correct, write it out to the object module $CC4Z9 as the user's security data. This program is provided only if at generation SECURE-USER was specified in the $ESEC statement.

Input is in the form of security data records which can be one of seven types. The type of record is specified by the first position of the input. The type indicator can be any of the following: C, Z, X, I, J, K, and L. In addition, a comment input record is allowed and is designated by an asterisk (*) in position 1. Comment records are logged to the system log device.

Each type of input record is intended to define a unique type of data. The following apply to all input data types:

- Positions 2 and 3 define the number of data bytes (including the sign) to be used from the input record and the data must begin in position 5.

- Position 4 is expected to be blank. Some record types may define signed (+ or –) values. In this case the sign, if present, is entered in position 5 and the data begins in position 6. The position occupied by the sign in the input record is counted in the number of input data characters that is specified in positions 2-3.

Each input record defines data to be placed in successive locations in the object module $CC4Z9.

The following input data types are accepted by $CCPAU. The data type is signified by the character specified in position 1:

'C' – Character String
The data can include any character from the extended character set including blanks. The output is the characters specified, in EBCDIC. The output length is the number of bytes specified in positions 2-3.

'Z' – Zoned Decimal
The data can include any decimal digits and may be preceeded by + or –. The output is a signed zoned-decimal number. The output length is the number of digits entered (that is, the number of bytes specified in positions 2-3, less one if a sign was specified).

'X' – Hexadecimal Data
The count in positions 2-3 must be even. Every character of input data must be in the range 0-9, A-F. The output is a set of bit configurations corresponding to the hexadecimal input. The output length is half the number of bytes as were specified in positions 2-3.

'I' – One-Byte Binary Integer
The input data is limited to three positions plus an optional sign. The numeric value must not exceed 255. The resultant data will occupy one byte in $CC4Z9, and be represented as a binary integer.

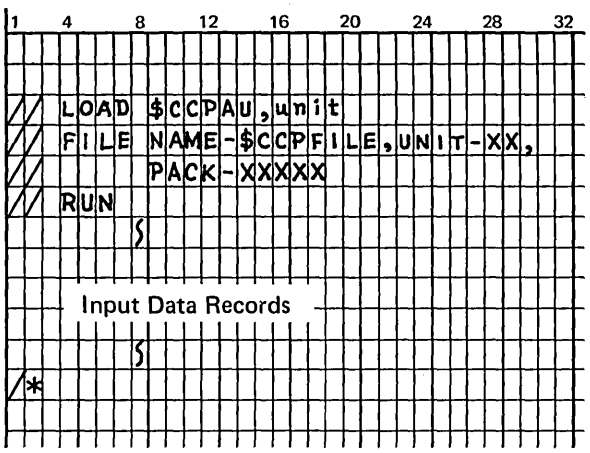
'J' – Two-Byte Binary Integer
The input data is limited to five positions plus an optional sign. The numeric value must not exceed 65535. The resultant output will occupy two bytes in $CC4Z9, and be represented as a binary integer.

'K' – Three-Byte Binary Integer
The input data is limited to eight positions plus an optional sign. The numeric value must not exceed 16777215. The resultant output will occupy three bytes in $CC4Z9, and be represented as a binary integer.

'L' – Four-Byte Binary Integer
The input data is limited to ten positions plus an optional sign. The numeric value must not exceed 4294967295. The resultant output will occupy four bytes in $CC4Z9, and be represented as a binary integer.

$CCPAU will continue to accept input data from the system input device until a /* is read, or until enough data to fill $CC4Z9 has been accumulated. At this time the data to be written to $CC4Z9 will be printed, in hexadecimal character form, on the system log device so that the data may be verified. After printing has finished, a CPU-A3 halt will occur. Option 0 as a reply will cause the data to be written to $CC4Z9. Option 2 will cause the job to be terminated and $CC4Z9 not to be updated.

The following OCL statements are needed:



For a description of writing a user security program, see *IBM System/3 Model 10 Disk System Communication Control Program Programmer's Reference,* GC21-7579.

Appendix A. Generation Control Statement Summary Chart

The following is a summary chart containing all valid CCP generation control statements. The statements are listed in alphabetic order. Three items are given for each statement:

- Name
- Format of the statement with all valid operands
- Synopsis of the functional description

For more detailed information on any of the CCP generation control statements, see index entry *generation control statements.*

Name	Generation Control Statement Description	Function
$EBSC	$EBSC [BSCA- {0 / 1 / 2}] [,DIAL- {YES / NO}] [,PP- {YES / NO}] [,MP- {YES / NO}] [,CS- {YES / NO}] [,GETMSG- {YES / NO}] [,ITB- {YES / NO}] [,RECSEP- {1E / XX}] [,ASCII- {YES / NO}] [,EBCDIC- {YES / NO}] [,RESPOL- {YES / NO}] [,AUTORS- {YES / NO}] [,XPRNCY- {YES / NO}]	Indicates general specifications concerning Binary Synchronous Communications (BSC) support.
$EBSD	$EBSD TYPE- {3275M1 / 3277M1 / 3284M1 / 3286M1 / 3275M2 / 3277M2 / 3284M2 / 3286M2 / 3735 / CPU}	Indicates a BSC device type which the CCP is to support.
$EFAC	$EFAC [MAXEUP- {1 / n}] [,DPF- {YES / NO}] [,ESCAPE- {NO / 'cccccc' / X'xxxxxxxxxxxx'}] [,PGMCNT- {YES / NO}] [,FSHARE- {YES / NO}] [,SYMFIL- {YES / NO}] [,FORMAT- {YES / NO}]	Indicates options which determine the CCP facilities to be included during generation.

Name	Generation Control Statement Description			Function
$EFIL		$EFIL	[SETS- {1 / n}] [,PROGS- {10 / n}] [,DFILES- {5 / n}] [,TERMS- {1 / n}] [,DUMPS- {1 / n}] [,CORE- {24K / 32K / 48K / 64K}] [,TRACE- {1 / n}] ,FLUNIT- {R1 / F1 / R2 / F2} ,FLPACK-pack [,TRKLOC-n]	Indicates user options on sizes of items which will affect the allocated size and location of $CCPFILE.
$EGEN		$EGEN	DSUNIT- {F1 / R1} CCUNIT- {F1 / R1 / F2 / R2} WKUNIT- {unit / 'unit,unit,unit'} WKPACK- {pack / 'pack,pack,pack'} ,DIUNIT- {F1 / R1 / F2 / R2} [MINRES- {YES / NO}]	Indicates where various unit and pack names are located during the CCP generation.
$EIOD		$EIOD	[CARD- {NO / MFCU / 1442 / 'MFCU,1442'}] [,PRINTR- {NO / 5203 / 1403}] [,DISKS- {NO / R2 / 'R2,F2'}] [,D5445- {NO / D1 / 'D1,D2'}]	Indicates the unit record devices which may be used by the user programs under the CCP and disk drives attached to the object system beyond the minimum requirements.
$EMLA		$EMLA	LINES-n[,XLATE- {YES / NO}]	Indicates whether the Multiple Line Terminal Adapter (MLTA) is to be used by the CCP and, if so, the number of MLTA lines. Also indicates whether or not line translation is always used on all MLTA operations.

Name	Generation Control Statement Description	Function
$EMLD	$EMLD TYPE-{1050, 1050D, 2740, 2740S, 2740C, 2740SC, 2740D, 2740DT, 2740DC, 2740DTC, 2740M2S, 2740M2SB, 2740M2SC, 2740M2SCB, 2741, 2741D, CMCSTD, SYS7C, SYS7SC, SYS7DC} ,XMCODE-{CORR, PTTCEBCD, PTTCBCD}	Indicates an MLTA terminal type to be supported with its features and the transmission code required on the line.
$EPLG	$EPLG LANG-{COBOL, FORTRAN, ASSEM, RPGII} ,PPUNIT-{R1, F1, R2, F2}	Indicates a programming language the user wishes to use for program preparation under the CCP.
$ESEC	$ESEC [SECURE-{NO, CCP, USER}] [,LUSI-{0, n}]	Indicates the type of terminal sign-on security to be used (if any).

Appendix B. Assignment Control Statement Summary Chart

The following is a summary chart containing all valid CCP assignment control statements. The statements are listed in alphabetic order. Three items are given for each statement:

- Name
- Format of the statement with all valid parameters
- Synopsis of the functional description

For more detailed information on any of the CCP assignment control statements, see index entry *assignment control statements.*

Name	Assignment Control Statement Description	Function
BSCALINE	// BSCALINE TYPE-{PP / CS / MP / SW} [,LINENUM-{1 / 2}] [,XMCODE-{EBCDIC / ASCII}] [,POLLLIST- 'termid [,termid] . . .'] [,NRETRY-{7 / n}] [,IDEXSEND—exchngid] [,POLLLOOP-{256 / n}] [,DBLBUF-{NO / YES}] [,WAIT-{180 / n}]	Defines the type of BSC line to be used and the features of the line.
BSCATERM	// BSCATERM TERMID-termid,TYPE-termtype, ATTRID-'attrid[,attrid] . . .' ,COMMAND-{NO / YES} [,ONLINE-{YES / NO}] [,IDEXRCV-exchngid] [,OFFACTN-{HOLD / DROP}] [,ADDRCHAR—addressing characters] [,POLLCHAR—polling characters]	Defines certain attributes of the terminals on BSCA lines.
DISKFILE	// DISKFILE NAME-filename[,DEVICE-{5444 / 5445}] ,ORG-{C / D / I} ,RECL-n[,KEYL-n] [,KEYPOS-n] [,MSTRINDX-{YES / NO}] [,MIXSIZE-n]	Describes the disk file which will be used during the execution of this set.

Name	Assignment Control Statement Description	Function
LIST	// LIST [SET- {id / ALL / DIR / CONFIG}] [PGMSTAT- {NO / YES}] [RESETPS- {NO / YES}]	This is a control statement for $CCPAL that defines which options the user wishes to exercise in listing the contents of the file, $CCPFILE.
MLTALINE	// MLTALINE TYPE- {PP / CS / SW / CW} ,LINENUM- {1 / 2 / 3 / 4 / 5 / 6 / 7 / 8} ,XMCODE- {PTTCEBCD / PTTCBDC / CORR} ,MAXRECL-n[,POLLLIST-'termid[,termid] . . .'] [,DATARATE- {134 / 600 / 1200}] [,AUTOPOLL- {NO / YES}] [,RCVINT- {NO / YES}] [,NRETRY- {2 / n}] [,DELAY-n] [,TIOLT- {YES / NO}]	Defines the type of communication line to be used and the features of the line.
MLTATERM	// MLTATERM TERMID-termid,TYPE-termtype, ,ATTRID-'attrid [,attrid] . . .' ,COMMAND- {NO / YES} [,ADDR-xx] [,ONLINE- {YES / NO}] [,PINCOMP-n] [,POUTCOMP-n] [,OFFACTN– {HOLD / DROP}]	Defines certain attributes of the terminals on MLTA lines.

Name	Assignment Control Statement Description	Function
PROGRAM	// PROGRAM NAME-pgmname,LANGUAGE-{FORTRAN / COBOL / RPG II / ASSEM} [,MRTMAX-n] [,REUSABLE-{NO / YES}] [,NEVEREND-{NO / YES}] [,RUNALONE-{NO / YES}] [,PRINTER-{NO / YES}] [,MFCU-{NO / YES}] [,RP1442-{NO / YES}] [,PGMDATA-{NO / YES}] [,ENDMSG-{NO / YES}] [,TERMS-'termname[/attrid] [,termname[/attrid]] ...'] [,FILES-'filename/access[/{SHR / NOSHR}],...'] [,SHAREDIO-{NO / YES}] [,PACK-{PROGRAM / SYSTEM}] [,DFFMTERM-n] [,DFFNDF-n] [,DFFSFDT-n]	Defines the logical structure and resource requirements of a user program.
SET	// SET [ID-{1 / c}] [,ACTION-{CREATE / REPLACE / DELETE / SYSMOD}] [,DFLTEXEC-{YES / NO}]	Specifies which set is to be operated upon and what operation is to occur.
SYMFILE	// SYMFILE NAME-ccccc,DISKFILE-'ccc...[,ccc...]...'	Defines a symbolic file name and specifies the disk files with which it can be validly associated.

Name	Assignment Control Statement Description	Function
SYSTEM	// SYSTEM MINUPA-nn.nnK,MINTPBUF-n [,MAXEUP-{1/n}] [,PASSWORD-password] [,PRINTER-{NO/YES}] [,MFCU-{NO/YES}] [,RP1442-{NO/YES}] [,PGMREQL-{6/n}] [,COMMANDL-{20/n}] [,TRACEBLK-{0/1/2/3/4/6/8/12/24}] [,SQB-n] [,FSB-n] [,DFFPACK-{SYSTEM/PROGRAM}]	Defines parts of the environment in which the CCP will execute.
TERMATTR	// TERMATTR ATTRID-attrid The following parameters may be specified for both the MLTA and BSCA terminals [,TRANSLAT-{YES/NO}] [,UPCASE-{YES/NO}] [,SWITCHED-{NO/AC/MC/AA/MA}] --- The following parameters may be specified for BSCA terminals only [,BLKL-n] [,RECL-n] [,DATAFORM-{RECORD/BLOCK/MESSAGE}] [,TRANSP-{NO/YES}] [,ITB-{NO/YES}] [,VARL-{NO/YES}] [,SPAN-{NO/YES}] [,VERIFYID-{NO/YES}] [,DFF3270-{YES/NO}]	Defines certain attributes of a terminal.

Name	Assignment Control Statement Description	Function
TERMNAME	// TERMNAME NAME-termname[,TERMID-termid] [,MSTRNAME-termname] [,INCOMP-n] [,OUTCOMP-n] [,PHONENUM-number]	Defines symbolic names to be associated with terminals and sub-terminals.

GENERATION STAGE MESSAGES

These messages will be issued at generation time to inform the user of invalid specifications of operands in the Generation Control statements.

The format of each message is as follows:

Positions 1-7: *ERROR* or WARNING. *ERROR* indicates generation will proceed no further. WARNING indicates an error was detected, the default value is assumed, and generation continues.

Positions 10-89: Message number and message

Examples

```
*ERROR*  CC906 COMMA AFTER LAST OPERAND
                BUT NO CONTINUATION INDICATOR

WARNING  CC482 RESPOL-YES SPECIFIED WITH
                CS-NO – TREATED AS RESPOL-NO
```

The errors signaled by the CCP generation fall into the following classes:

1. Error in the format of the statement itself, such as:

   - Unknown statement type
   - Incorrect keyword
   - Invalid continuation of the statement

   Errors in this class do not permit the statement to be examined further; therefore, there might be other errors in the statement that cannot be detected if an error of this class is detected.

2. Error in the sequence of statements. A statement which must precede this statement of another type, but that other statement has preceded this one. This error can also occur because of an error of class #1 on the preceding statement.

3. A required operand is missing. Certain operands are required. If these are not specified, generation cannot proceed successfully.

4. The parameter specified in an operand is not one of the valid choices. If an invalid parameter is specified, generation cannot proceed successfully.

5. Conflict among operands specified. Either:

   - Two operands specified cannot be specified together, or
   - An operand specified requires a certain other operand be specified, and it was not.

The messages issued by the CCP generation explain in their texts the reason the diagnostic was issued. Consult *Chapter 6. Generation Stage,* for a full explanation of the specification rules. The possible diagnostics are listed by the generation statements for which they might be issued.

Messages Associated with the $EIOD Statement

ERROR	CC005	$EIOD STATEMENT NOT IN PROPER SEQUENCE
ERROR	CC010	INVALID 'CARD' PARAMETER – MUST BE MFCU/1442/'MFCU,1442'/NO
ERROR	CC015	INVALID 'PRINTR' PARAMETER – MUST BE 5203/1403/NO
ERROR	CC020	INVALID 'DISKS' PARAMETER – MUST BE R2/'R2,F2'/NO
ERROR	CC025	INVALID 'D5445' PARAMETER – MUST BE D1/'D1,D2'/NO

Messages Associated with the $EFAC Statement

ERROR	CC050	$EFAC STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC055	INVALID 'XAMEUP' PARAMETER – MUST BE DIGIT IN RANGE 1-8
ERROR	CC060	INVALID 'DPF' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC065	INVALID 'FSHARE' PARAMETER – MUST BE YES/Y/NO/N
WARNING	CC067	FSHARE-YES SPECIFIED WITH MAXEUP-1 – TREATED AS FSHARE-NO
ERROR	CC075	INVALID 'SYMFIL' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC080	INVALID 'PGMCNT' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC085	INVALID 'ESCAPE' PARAMETER – MUST BE '6 CHARS'/X'12 CHARS'/NO
ERROR	CC090	INVALID 'FORMAT' PARAMETER – MUST BE YES/Y/NO/N

Messages Associated with the $EPLG Statement

ERROR	CC100	$EPLG STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC105	INVALID 'LANG' PARAMETER – MUST BE COBOL/RPGII/ASSEM/FORTRAN
ERROR	CC110	DUPLICATE $EPLG STATEMENT FOR { RPGII / COBOL / FORTRAN / ASSEM } LANGUAGE
ERROR	CC115	INVALID 'PPUNIT' PARAMETER – MUST BE R1/F1/R2/F2
ERROR	CC120	MISSING 'LANG' OPERAND – MUST BE SPECIFIED
ERROR	CC125	MISSING 'PPUNIT' OPERAND – MUST BE SPECIFIED

Messages Associated with the $ESEC Statement

ERROR	CC150	STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC155	INVALID 'SECURE' PARAMETER – MUST BE CCP/USER/NO
ERROR	CC160	INVALID 'LUSI' PARAMETER – ONLY 0 IS VALID UNLESS SECURE-USER
ERROR	CC165	LUSI-0 SPECIFIED, OR 'LUSI' OPERAND OMITTED, WITH SECURE-USER
ERROR	CC170	INVALID 'LUSI' PARAMETER – MUST BE NUMBER IN RANGE 1-4096

Messages Associated with the $EFIL Statement

ERROR	CC200	$EFIL STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC205	INVALID 'SETS' PARAMETER – MUST BE NUMBER IN RANGE 1-25
ERROR	CC210	INVALID 'PROGS' PARAMETER – MUST BE NUMBER IN RANGE 1-255
ERROR	CC215	INVALID 'DFILES' PARAMETER – MUST BE NUMBER IN RANGE 0-50
ERROR	CC220	INVALID 'TERMS' PARAMETER – MUST BE NUMBER IN RANGE 1-254
ERROR	CC225	INVALID 'DUMPS' PARAMETER – MUST BE NUMBER IN RANGE 1-9
ERROR	CC230	INVALID 'CORE' PARAMETER – MUST BE 24K/32K/48K/64K
ERROR	CC235	INVALID 'TRACE' PARAMETER – MUST BE NUMBER IN RANGE 1-20
ERROR	CC240	MISSING 'FLUNIT' OPERAND – MUST BE SPECIFIED
ERROR	CC242	INVALID 'FLUNIT' PARAMETER – MUST BE R1/F1/R2/F2

ERROR	CC244	MISSING 'FLPACK' OPERAND – MUST BE SPECIFIED
ERROR	CC245	INVALID 'FLPACK' PARAMETER – MUST BE 1-6 CHARACTERS
ERROR	CC248	INVALID 'TRKLOC' PARAMETER – MUST BE NUMBER IN RANGE 8-405

Messages Associated with the $EMLA Statement

ERROR	CC250	$EMLA STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC255	INVALID 'LINES' PARAMETER – MUST BE NUMBER IN RANGE 0-8
ERROR	CC260	MISSING 'LINES' OPERAND – MUST BE SPECIFIED IF STATEMENT USED
ERROR	CC265	LINES-0, BUT OTHER KEYWORD SPECIFIED WITH NON-DEFAULT PARAMETER
ERROR	CC270	INVALID 'XLATE' PARAMETER – MUST BE YES/Y/NO/N

Messages Associated with the $EMLD Statement

ERROR	CC300	$EMLD STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC305	STATEMENT USED, BUT NO MLTA LINES SPECIFIED
ERROR	CC310	MISSING 'TYPE' OPERAND – MUST BE SPECIFIED
ERROR	CC315	INVALID 'TYPE' PARAMETER – MUST BE MLTA TERMINAL DESIGNATION
ERROR	CC320	MISSING 'XMCODE' OPERAND – MUST BE SPECIFIED
ERROR	CC325	INVALID 'XMCODE' PARAMETER – MUST BE CORR/PTTCEBCD/PTTCBCD
ERROR	CC330	XMCODE { PTTCEBCD / PTTCBCD / CORR } NOT VALID FOR TERMINAL TYPE SPECIFIED

Messages Associated with the $EBSC Statement

ERROR	CC400	$EBSC STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC405	MISSING 'BSCA' OPERAND – MUST BE SPECIFIED IF STATEMENT USED
ERROR	CC410	INVALID 'BSCA' PARAMETER – MUST BE NUMBER IN RANGE 0-2
ERROR	CC415	BSCA-0, BUT OTHER OPERAND SPECIFIED WITH NON-DEFAULT PARAMETER
ERROR	CC420	INVALID 'DIAL' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC425	INVALID 'PP' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC430	INVALID 'MP' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC435	INVALID 'CS' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC440	BSCA PRESENT BUT NO LINE TYPES SPECIFIED
ERROR	CC445	INVALID 'GETMSG' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC450	INVALID 'ITB' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC455	INVALID 'RECSEP' PARAMETER – MUST BE TWO HEX DIGITS
ERROR	CC460	INVALID 'ASCII' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC465	INVALID 'EBCDIC' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC470	BSCA PRESENT BUT NEITHER TRANSMISSION CODE IS USED
ERROR	CC472	EBCDIC-YES AND ASCII-YES SPECIFIED WITH BSCA-1 – NOT POSSIBLE
ERROR	CC480	INVALID 'RESPOL' PARAMETER – MUST BE YES/Y/NO/N
WARNING	CC482	RESPOL-YES SPECIFIED WITH CS-NO – TREATED AS RESPOL-NO
ERROR	CC485	INVALID 'AUTORS' PARAMETER – MUST BE YES/Y/NO/N
WARNING	CC487	AUTORS-YES SPECIFIED WITH MS-NO – TREATED AS AUTORS-NO
ERROR	CC490	INVALID 'XPRNCY' PARAMETER – MUST BE YES/Y/NO/N
ERROR	CC492	XPRNCY-YES SPECIFIED WITH EBCDIC-NO – NOT VALID

Messages Associated with the $EBSD Statement

ERROR	CC500	$EBSD STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC505	$EBSD STATEMENT USED, BUT NO BSC ADAPTERS SPECIFIED
ERROR	CC510	MISSING 'TYPE' OPERAND – MUST BE SPECIFIED
ERROR	CC515	INVALID 'TYPE' PARAMETER – MUST BE BSCA TERMINAL DESIGNATION

Messages Associated with the $EGEN Statement

ERROR	CC600	$EGEN STATEMENT OUT OF SEQUENCE – OR PRECEDING STATEMENT ERROR
ERROR	CC605	DUPLICATE $EGEN STATEMENT – CONTENTS IGNORED
ERROR	CC610	MISSING 'DSUNIT' OPERAND – MUST BE SPECIFIED
ERROR	CC615	INVALID 'DSUNIT' PARAMETER – MUST BE R1/F1
ERROR	CC620	MISSING 'CCUNIT' OPERAND – MUST BE SPECIFIED
ERROR	CC625	INVALID 'CCUNIT' PARAMETER – MUST BE R1/F1/R2/F2
ERROR	CC630	MISSING 'WKUNIT' OPERAND – MUST BE SPECIFIED
ERROR	CC635	INVALID 'WKUNIT' PARAMETER – MUST BE R1/F1/R2/F2 OR SERIES OF 3
ERROR	CC637	WKUNIT/WKPACK ERROR – PACKS ****** AND ****** BOTH ON UNIT **
ERROR	CC640	MISSING 'WKPACK' OPERAND – MUST BE SPECIFIED
ERROR	CC645	INVALID 'WKPACK' PARAMETER – MUST BE 1-6 CHAR NAME OR SERIES OF 3
ERROR	CC650	MISSING 'DIUNIT' OPERAND – MUST BE SPECIFIED
ERROR	CC655	INVALID 'DIUNIT' PARAMETER – MUST BE R1/F1/R2/F2
ERROR	CC660	'CCUNIT' SAME AS 'DIUNIT' – NOT PERMITTED
ERROR	CC665	'PPUNIT' FROM $EPLG STATEMENT SAME AS 'DIUNIT' – NOT PERMITTED
ERROR	CC670	INVALID 'MINRES' PARAMETER – MUST BE YES/Y/NO/N

Messages Associated with Special Situations

ERROR	CC700	NO MLTA OR BSCA SUPPORT SPECIFIED – AT LEAST ONE REQUIRED
ERROR	CC705	FORMAT-YES SPECIFIED IN $EFAC REQUIRES GETMSG-YES IN $EBSC
ERROR	CC710	NO PROGRAMMING LANGUAGE SUPPORTED – AT LEAST ONE REQUIRED
ERROR	CC715	FORMAT-YES IN $EFAC REQUIRES 3270 DISPLAY DEVICE IN $EBSD

Messages Associated with Each Statement

ERROR	CC901	INVALID STATEMENT TYPE OR SOURCE LIBRARY ON WRONG PACK
ERROR	CC902	INVALID STATEMENT TYPE, OR PREVIOUS ERROR ON STATEMENT
ERROR	CC903	A KEYWORD USED IS NOT VALID FOR THIS STATEMENT TYPE
ERROR	CC904	A PARAMETER IS MISSING OR HAS INVALID FORM
ERROR	CC905	A DELIMITER IS EITHER INVALID OR WRONGLY PLACED IN THE STATEMENT
ERROR	CC906	COMMA AFTER LAST OPERAND BUT NO CONTINUATION INDICATOR
ERROR	CC907	ON CONTINUATION CARD, COLS. 1-13 MUST BE BLANK – THEY ARE NOT
ERROR	CC909	ERROR '**' ISSUED BY MACRO PROCESSOR – POSSIBLE CCP ERROR
ERROR	CC990	INCOMPLETE INPUT TO CCP GENERATION
ERROR	CC991	NOT A VALID STATEMENT FOR CCP GENERATION

ASSIGNMENT STAGE MESSAGES

These messages will be issued at assignment time to inform the user of invalid specifications of operands in the assignment control statement. Wherever possible the position with which the error is associated, is marked by an asterisk in the following print line.

The actual format of the message is as follows:

Positions 1-5: Message number

Position 7: T, W, or I. T indicates an error was found in a statement of this assignment set; this set of statements will not be processed by the assignment stage; however, the remaining statements in this set are also analyzed for errors.

W indicates an error was detected, the default value is assumed, and assignment continues.

I indicates information only.

Positions 10-89: Message

Examples

CA004 T VALUE FOR KEYWORD IS NOT YES/NO

CA163 W MAXBLKL TOO SMALL, VALUE GIVEN IGNORED

The following is a list of the message numbers and messages associated with the Assignment Build control statements:

CA002 T DELIMITING APOSTROPHE MISSING

Reason – Only one apostrophe was found in a keyword value.

System Action – The set is not processed.

CA003 T INVALID KEYWORD FOR THIS STATEMENT OR PAST COL 71

Reason – The keyword is not correct for this statement type. The keyword extends into column 72; the keyword is expected but not given or not followed by a hyphen.

System Action – The set is not processed.

CA004 T VALUE FOR KEYWORD IS NOT YES/NO

Reason – The value for the keyword must be YES or NO.

System Action – The set is not processed.

CA005 T KEYWORD GIVEN PREVIOUSLY IN THIS STATEMENT

Reason – There are duplicate keywords within a statement; not allowed.

System Action – The set is not processed.

CA006 T DATA IS IN COLUMN 72

Reason – A format or punctuation error occurred. The keyword or value must not be in column 72.

System Action – The set is not processed.

CA007 T VALUE IS NOT NUMERIC OR INVALID SIZE NUMBER GIVEN

Reason – The value must be numeric and within the specified range.

System Action – The set is not processed.

CA008 T VALUE IS NOT CORRECT RESPONSE FOR THIS KEYWORD

Reason — The value given is not valid for this keyword.

System Action — The set is not processed.

CA009 T SUBLIST NOT VALID FOR THIS KEYWORD

Reason — An apostrophe preceded a value but there was no apostrophe after the value.

System Action — The set is not processed.

CA011 T INCORRECT USE OF SUBLIST VALUES

Reason — Two apostrophes were given in front of a value or there were embedded blanks.

System Action — The set is not processed.

CA012 T TOO MANY OR TOO FEW CHARACTERS IN VALUE

Reason — The value length was not specified correctly.

System Action — The set is not processed.

CA013 T INCORRECT USE OF SPLIT-VALUE

Reason — A blank follows a slash, or there are too many slashes for this keyword.

System Action — The set is not processed.

CA014 T KEYWORD EXPECTED OR INCORRECT USE OF PUNCTUATION

Reason — There is no hyphen to indicate a keyword (syntax error), or attempted to use a sublist and an apostrophe did not precede the first value.

System Action — The set is not processed.

CA015 T VALUE GIVEN IS TOO LONG

Reason — The value cannot exceed the specified maximum value.

System Action — The set is not processed.

CA025 T REQUIRED KEYWORD NOT GIVEN

Reason — A required keyword was not given.

System Action — The set is not processed.

CA026 W START-UP DEFAULT SET DELETED

Reason — Deletion of the startup default set has been specified.

System Action — The set is deleted; the default set ID is set to null.

CA027 T FOUND HYPHEN INSTEAD OF COMMA OR APOSTROPHE

Reason — The delimiting apostrophe for a previous sublist was not given, or the value was not given for the previous keyword.

System Action — The set is not processed.

CA028 T TOO MANY STATEMENTS GIVEN

Reason – The maximum number of statements allowed was exceeded.

BSCATERM, MLTATERM	239 maximum
TERMNAME	254 maximum
DISKFILE, SYMFILE	100 maximum
PROGRAM	255 maximum
BSCATERM, MLTATERM	Too many terminals for line type

System Action – The set is not processed.

CA029 T TOO MANY ENTRIES GIVEN

Reason – The sublist exceeded the maximum number of entries allowed.

System Action – The set is not processed.

CA033 T INVALID TERMINAL ID

Reason – At least one of the ID characters was not a non-blank extended alphameric character, or the ID was a restricted ID ($C).

System Action – The set is not processed.

CA040 T DISK ERROR OCCURRED ON EITHER $CCPWORK OR $CCPFILE

Reason – A disk hardware error on a read or write operation occurred.

System Action – The set is not processed.

CA041 T STATEMENT NOT IN PROPER SEQUENCE

Reason – The statements must be in proper sequence.

System Action – The set is not processed.

CA043 W SET ID NOT FOUND IN DIRECTORY

Reason – A // SET statement with ACTION-DELETE was specified, but the set ID is not in the directory.

System Action – The statement is ignored.

CA044 T CREATE SET ID EXISTS ALREADY

Reason – A // SET statement with ACTION-CREATE was specified, and the ID given is the same as an ID that already exists in the directory.

System Action – The set is not processed.

CA045 T CCP OR AUX PROG EXECUTING IN OTHER LEVEL

Reason – To run the assignment program, the other level must not be running a program supplied with the CCP.

System Action – The set is not processed.

CA046 W REPLACE SET ID NOT FOUND, ASSUME ACTION-CREATE

Reason – A // SET statement with ACTION-REPLACE was specified, but the ID given does not exist in the directory.

System Action – The set is processed assuming ACTION-CREATE.

CA051 T STATEMENT DOES NOT START WITH '//'

Reason — The statement read was not a comment, /*, or a valid control statement.

System Action — The set is not processed.

CA052 T INVALID STATEMENT IDENTIFIER

Reason — The statement identifier following the // is unidentified.

System Action — The set is not processed.

CA053 T $CCPFILE UNUSABLE

Reason — The $CCPFILE has been altered, and is not usable. The first two characters of of the configuration record do not match those inserted by generation.

System Action — The set is not processed.

CA054 T SYSIN READ ERROR

Reason — A read error occurred on the system reader.

System Action — The set is not processed.

CA055 T SYMBOLIC NAME IS REQUIRED FOR ALL DEFINED TERMINALS

Reason — The MLTATERM or BSCATERM statements defined a terminal, and there was not a TERMNAME statement specified to define a symbolic name for that terminal.

System Action — The set is not processed.

CA060 T NO MORE ROOM IN $CCPFILE OR $CCPWORK

Reason — The end of the extents has been reached in $CCPWORK or $CCPFILE.

System Action — The set is not processed. The operator may rerun with a larger work file.

CA061 T INVALID ADDR/POLL CHARACTERS

Reason — The ADDRCHAR or POLLCHAR parameter values are not valid.

System Action — The set is not processed.

CA070 T ATTRIBUTES SET WAS NOT SPECIFIED FOR THIS TERMINAL

Reason — The attributes set to be used with a terminal was not associated with any terminal on that line.

System Action — The set is not processed.

CA071 T DUPLICATE REQUIRED TERMINALS INVALID

Reason — The same terminal name was referenced twice, or two terminal names referenced the same terminal ID.

System Action — The set is not processed.

CA072 T TERMINAL NAME INVALID

Reason — A restricted terminal name was given (for example, CONSOL, ALL, or all blanks).

System Action — The set is not processed.

CA073 T ADDR INVALID WITH LINE TYPE SW OR PP

Reason – A terminal on SW or PP type lines cannot have an address.

System Action – The set is not processed.

CA074 T ADDR REQUIRED WITH LINE TYPE CS OR CW

Reason – A terminal on CS or CW type lines must have an address.

System Action – The set is not processed.

CA075 T TERMINAL TYPE 2740 INVALID WITH RECEIVE INTERRUPT

Reason – 2740 terminals cannot support receive interrupt.

System Action – The set is not processed.

CA076 T DELAY/AUTOPOLL IS INVALID WITH LINE TYPE SW OR PP

Reason – Line types SW and PP cannot support delay or autopoll.

System Action – The set is not processed.

CA077 T ADDR/POLLCHAR INVALID WITH LINE TYPE SW OR PP

Reason – Line types SW and PP do not support ADDRCHAR or POLLCHAR.

System Action – The set is not processed.

CA078 W PGMREQL LESS THAN MINIMUM REQUIRED, ASSUME DEFAULT

Reason – The PGMREQL parameter requires a minimum value of 2.

System Action – The set is processed using the value of 6 for the PGMREQL parameter.

CA098 W MINTPBUF TOO SMALL, CALCULATED VALUE USED

Reason – The minimum TP buffer size is equal to the larger of the following:

1. The value of PGMREQL + 4 plus a variable based on core area required for the largest assignment PROGRAM statement.
2. Value of COMMANDL + 4.

Note: If the calculated value is used, this value may be displayed using $CCPAL and will appear as the value for MINTPBUF.

System Action – The set is processed using the larger of 1 or 2 as a value for MINTPBUF.

CA099 I PREVIOUS SET NOT PROCESSED

Reason – At least one termination error was found while processing this set.

System Action – The set is not processed.

CA101 T SET ID NOT VALID

Reason – ID is not an extended alphameric character (0-9, A-Z, #, $, or @).

System Action – The set is not processed.

CA102 T MINUPA VALUE INVALID

Reason — The character K was not specified as the last character of the value, or the value is not in 256-byte increments, or the value is less than 5K.

System Action — The set is not processed.

CA103 T MAXEUP GREATER THAN GIVEN AT GENERATION

Reason — The value for MAXEUP given here was greater than what was specified for MAXEUP on the $EFAC generation control statement.

System Action — The set is not processed.

CA104 W SQB MUST BE ZERO FOR ONE USER PROGRAM, VALUE IGNORED

Reason — The value must be zero if MAXEUP-1 was specified.

System Action — The set is processed with the value of zero for SQB.

CA106 T SQB VALUE NOT ALLOWED, GENERATION CONFLICT

Reason — No file sharing was specified during generation.

System Action — The set is not processed.

CA108 T FSB VALUE IS NOT ALLOWED, GENERATION CONFLICT

Reason — No symbolic file names were specified during generation.

System Action — The set is not processed.

CA109 T PASSWORD SHOULD HAVE BEEN GIVEN

Reason — The use of a password was specified during generation.

System Action — The set is not processed.

CA110 T PASSWORD NOT SUPPORTED AT GENERATION

Reason — The use of a password was not specified during generation.

System Action — The set is not processed.

CA111 T GENERAL POLL NOT SUPPORTED BY CCP

Reason — BSCA line EBCDIC and POLLCHAR - second character is X'7F'; or BSCA line ASCII and POLLCHAR - second character is X'22'.

System Action — The set is not processed.

CA115 W DUPLICATE 'ATTRID' VALUE GIVEN, STATEMENT IGNORED

Reason — Duplicate values for ATTRID are not allowed.

System Action — The set is processed, and the statement is ignored.

CA116 T UPPER CASE REQUIRES TRANSLATE

Reason — UPCASE-YES is valid only if TRANSLAT-YES is specified.

System Action — The set is not processed.

CA117 T BSCA NOT SUPPORTED AT GENERATION, BSCA KEYWORD GIVEN

Reason – The BSCA is not specified in the configuration record.

System Action – The set is not processed.

CA118 W VARL/SPAN INVALID WITH ITB, ASSUME VARL/SPAN-NO

Reason – VARL and SPAN are not valid when ITB has been specified.

System Action – The set is processed with NO assumed as the value for VARL/SPAN

CA119 T DUPLICATE TERMINAL ADDRESS ON SAME LINE IS INVALID

Reason – The ADDR value is a duplicate of a previous ADDR value on this line.

System Action – The set is not processed.

CA120 T RECL ONLY VALID IF DATAFORM-RECORD IS SPECIFIED

Reason – When RECL-n is specified, DATAFORM-RECORD must also be specified.

System Action – The set is not processed.

CA121 T RECL LARGER THAN BLKL OR INVALID BLOCKING FACTOR

Reason – The record length cannot be specified as larger than the block length, or blocking factor exceeds 255.

System Action – The set is not processed.

CA122 T RECL NOT EVEN MULTIPLE OF BLKL

Reason – The record length must be an even multiple of the block length, except if SPAN-YES is specified.

System Action – The set is not processed.

CA125 T IDEXSEND/IDEXRCV IS VALID ONLY ON SWITCHED LINES

Reason – IDEXSEND/IDEXRCV is valid only if TYPE-SW is specified.

System Action – The set is not processed.

CA126 T POLLLIST AND POLLLOOP ARE VALID ONLY ON CS LINES

Reason – POLLLIST and POLLLOOP are valid only if TYPE-CS is specified.

System Action – The set is not processed.

CA127 T TRANSPARENCY INVALID WITH XMCODE-ASCII

Reason – TRANSP-YES cannot be specified with XMCODE-ASCII.

System Action – The set is not processed.

CA129 T MAXRECL IS LESS THAN MINIMUM REQUIRED

Reason – A value less than the minimum required (16 bytes) by MAXRECL has been specified.

System Action – The set is not processed.

CA133 W MAXRECL SIZE WILL NOT SUPPORT ON-LINE TEST

Reason – The online test and the maximum record size specified is less than 100.

System Action – The set is processed.

CA135 T LINE NUMBER GIVEN ON PREVIOUS STATEMENT

Reason – The same line cannot be specified twice.

System Action – The set is not processed.

CA136 T LINENUM OR LINE TYPE NOT SUPPORTED AT GENERATION

Reason – LINENUM or LINETYPE is not specified in the configuration record; or the configuration record specifies one BSCA line, and both BSCA lines have been entered.

System Action – The set is not processed.

CA138 T TYPE-CS REQUIRES A POLL LIST

Reason – TYPE-CS is valid only if POLLLIST is specified.

System Action – The set is not processed.

CA140 T INVALID HEXADECIMAL CHARACTER ENTERED

Reason – Wrong length, leading asterisk and no trailing asterisk, or an invalid hexadecimal character has been specified in one of the following:

MLTA	ADDR
BSCA	IDEXSEND IDEXRCV ADDRCHAR POLLCHAR

System Action – The set is not processed.

CA141 T TOO MANY OR TOO FEW CHARACTERS IN VALUE

Reason – IDEXRCV exceeds 15 characters, ADDRCHAR or POLLCHAR exceeds 7 characters, ADDRCHAR or POLLCHAR less than 2 characters, or ADDRCHAR not same length as POLLCHAR.

System Action – The set is not processed.

CA142 T TRANSMISSION CODE NOT SUPPORTED AT GENERATION

Reason – The transmission code is not specified in the configuration record.

System Action – The set is not processed.

CA160 T TYPE REQUIRES EDEXSEND OR ADDRCHAR/POLLCHAR

Reason – The 3735 type terminal on a switched line requires IDEXSEND, or the 3735 type terminal on a CS line requires ADDECHAR/POLLCHAR.

System Action – The set is not processed.

CA161 T DEFINED TERMINAL NOT IN POLL LIST

Reason – A terminal was defined as being on a CS line, but the ID was not found in the POLL LIST for that line.

System Action – The set is not processed.

CA162 W TYPE NOT COMMAND CAPABLE, ASSUME COMMAND-NO

Reason – COMMAND-YES is not supported for the 3735 or BSCA terminals on a switched auto or manual call line.

System Action – The set is processed and COMMAND-NO is assumed.

CA163 T POLL LIST TOO LONG

Reason – The MLTA poll list is limited to a maximum of 127 IDs; each 1050 entry in a poll list requires four.

System Action – The set is not processed.

CA164 T VERIFYID REQUIRES IDEXRCV

Reason – If IDs are to be verified, then IDEXRCV must be specified.

System Action – The set is not processed.

CA165 W VERIFYID IGNORED, INVALID WITH LINE TYPE SPECIFIED

Reason – The value YES is only valid for BSCA terminals on switched lines.

System Action – The set is processed assuming VERIFYID-NO.

CA166 W SPAN VALID ONLY FOR 3735 AND CPU, ASSUME SPAN-NO

Reason – The value YES is valid only for BSCA terminal type 3735 or CPU.

System Action – The set is processed assuming SPAN-NO.

CA167 T SPAN/VARL REQUIRES RECL TO BE OMITTED OR EQUAL BLKL

Reason – If span records or variable length records is specified, the record length must equal the block length or be omitted.

System Action – The set is not processed.

CA168 W OFF ACTION APPLIES TO COMMAND TERMINALS ONLY

Reason – COMMAND was specified as NO and OFFACTN was specified.

System Action – The set is processed and the OFFACTN value is ignored.

CA172 T TERMINAL ID GIVEN ON PREVIOUS STATEMENT

Reason – The same terminal ID cannot be specified twice.

System Action – The set is not processed.

CA173 T TERMINAL TYPE NOT SUPPORTED AT GENERATION

Reason – The terminal type is not specified in the configuration record.

System Action – The set is not processed.

CA174 T COMBINATION OF TERMINAL TYPES ON THIS LINE INVALID

Reason – A 3270 type terminal was specified on other than a control station line, or a 3735 terminal was specified on a PP or MP line. All the terminals on a line must be of the same type.

System Action – The set is not processed.

CA175 T ATTRIBUTES SET INVALID

Reason – The attributes required have not been previously defined on a TERMATTR statement, or an attribute set was specified for a BSCA terminal and no block length was specified (a required parameter for BSCA terminals).

System Action – The set is not processed.

CA176 T LINE TYPE CONFLICTS WITH ATTRIBUTES SWITCHED VALUE

Reason – If the attributes specify switched line, the MLTALINE statement associated with a terminal which uses this attribute set must have TYPE-SW or TYPE-CW specified.

System Action – The set is not processed.

CA177 W MAXRECL TOO SMALL, MINIMUM REQUIRED VALUE USED

Reason – If any terminal on a line is a command terminal, the minimum MAXRECL value is as follows:

1. Terminal type 2740 Model 2, minimum value of 77.
2. Any other terminal type, minimum value of 107.

System Action – The set is processed, assuming the minimum required value.

CA178 T PREVIOUS POLLLIST CONTAINS AN INVALID TERMINAL ID

Reason – All terminal IDs in a POLLLIST must have been defined on BSCATERM or MLTATERM statements following the appropriate line statement.

System Action – The set is not processed.

CA179 T SYSTEM STATEMENT SPECIFIES NO SYMBOLIC FILES

Reason – The FSB value in the SYSTEM statement was zero or omitted indicating no symbolic files are to be used with this assignment set.

System Action – The set is not processed.

CA185 T 1050 REQUIRES PINCOMP AND/OR POUTCOMP

Reason – The 1050 on a dial (switched) line requires that PINCOMP and POUTCOMP be specified.

System Action – The set is not processed.

CA186 T TERMINAL TYPE CONFLICTS WITH LINE TYPE OR XMCODE

Reason – The terminal type specified conflicts with the line type or transmission code specified.

System Action – The set is not processed.

CA187 T PINCOMP/POUTCOMP VALID ONLY FOR 1050

Reason – The PINCOMP and POUTCOMP parameters can be specified only for the 1050 system.

System Action – The set is not processed.

CA188 W TERMINAL ATTRIBUTES CONTAIN BSCA KEYWORDS

Reason – The terminal attributes should contain MLTA specifics only.

System Action – The set is processed.

CA195 T DUPLICATE TERMINAL NAME ENTERED

Reason – Two terminals cannot have the same name.

System Action – The set is not processed.

CA196 T TERMID NOT DEFINED ON BSCATERM OR MLTATERM STATEMENT

Reason – The terminal ID has not been specified.

System Action – The set is not processed.

CA197 T PHONE NUMBER VALID ONLY IF TERMINAL ID IS SPECIFIED

Reason – The terminal ID has not been specified.

System Action – The set is not processed.

CA198 T INCOMP/OUTCOMP/MSTRNAME INVALID WITH TERMINAL TYPE

Reason – INCOMP, OUTCOMP and MSTRNAME have been specified for a terminal type other than the 1050.

System Action – The set is not processed.

CA200 T MASTER NAME REQUIRES INCOMP OR OUTCOMP

Reason – If MSTRNAME is specified, either INCOMP and/or OUTCOMP must also be specified.

System Action – The set is not processed.

CA202 T PHONE NUMBER IS REQUIRED FOR AUTO OR MANUAL CALL

Reason – PHONENUM is required for switched lines using auto call or manual call.

System Action – The set is not processed.

CA203 T MASTER NAME NOT PREVIOUSLY DEFINED

Reason – The value specified by MSTRNAME must match a value given as a terminal symbolic name on a previous TERMNAME statement.

System Action – The set is not processed.

CA209 W MSTRINDX INVALID WITH 5445, ASSUME NO MSTRINDX

Reason – Master index is not supported for the 5445 under the CCP.

System Action – The set is processed using no master index.

CA210 T DEVICE NOT SUPPORTED AT GENERATION

Reason – The 5445 specified is not specified in the configuration record.

System Action – The set is not processed.

CA212 T KEYL AND KEYPOS REQUIRED FOR INDEXED FILES

Reason – Indexed files require that KEYL and KEYPOS be specified.

System Action – The set is not processed.

CA213 T KEYL, KEYPOS, AND MASTER INDEX REQUIRES ORG-I

Reason – KEYL, KEYPOS, and MSTRINDX were specified for other than an indexed file.

System Action – The set is not processed.

CA214 T DISKFILE NAME NOT PREVIOUSLY DEFINED

Reason – A DISKFILE name specified on a SYMFILE statement has not been previously defined by a DISKFILE statement

System Action – The set is not processed.

CA216 T SYMBOLIC NAME CANNOT REFERENCE BOTH 5444 AND 5445 FILES

Reason – The SYMFILE name referencing the 5444 and 5445 files is not supported under the CCP.

System Action – The set is not processed.

CA217 T DISKFILE CONTAINS DUPLICATE ENTRIES

Reason – The DISKFILE parameter on a SYMFILE statement contains two or more identical entries.

System Action – The set is not processed.

CA218 T INVALID FILE NAME - $CCPFILE

Reason – $CCPFILE is a restricted name and must not be used.

System Action – The set is not processed.

CA219 T DUPLICATE FILE NAME ENTERED

Reason – The same file cannot be specified twice.

System Action – The set is not processed.

CA220 W MSTRINDX KEYWORD CONFLICT, ASSUME NOT MASTER INDEX

Reason – If MIXSIZE is specified, then MSTRINDX-NO cannot be specified.

System Action – The set is processed assuming no master index.

CA221 T INCONSISTENT DISKFILE ATTRIBUTES

Reason – The symbolic file references files with different organization, record length, key length or key position.

System Action – The set is not processed.

CA232 T TRANSLATION OF LINE CODE REQUIRED BY GENERATION

Reason – Attributes set specifies TRANSLAT-NO, and translation was specified at generation that all MLTA terminals would require line code translation.

System Action – The set is not processed.

CA233 T DFF SUPPORT WITH ASCII CODE REQUIRES TRANSLAT-YES

Reason – Display Format Facility support used with line code of ASCII requires translation under the CCP.

System Action – The set is not processed.

CA234 W SQB OMITTED, CCP GENERATED TO SHARE FILES

Reason – The SQB parameter was omitted in the SYSTEM statement, but the CCP was generated to permit disk file sharing.

System Action – The set is processed assuming no disk file sharing.

CA235 T PROGRAM NAME PREVIOUSLY GIVEN

Reason – The same program cannot be specified twice.

System Action – The set is not processed.

CA236 W MRTMAX/PGMDATA CONFLICT, ASSUME PGMDATA-YES

Reason – A program specified as servicing multiple requesting terminals (MRTMAX) must have specified PGMDATA-YES.

System Action – The set is processed assuming PGMDATA-YES.

CA237 W REUSABLE/NEVEREND CONFLICT, ASSUME NEVEREND-NO

Reason – A never-ending program cannot be specified REUSABLE.

System Action – The set is processed assuming NEVEREND-NO.

CA238 W REUSABLE/LANGUAGE CONFLICT, ASSUME REUSABLE-NO

Reason – RPG II and FORTRAN programs are not reusable programs.

System Action – The set is processed assuming REUSABLE-NO.

CA239 T PRINTER, MFCU OR RP1442 NOT SUPPORTED AT GENERATION

Reason – The printer, MFCU, or RP1442 was not specified in the configuration record.

System Action – The set is not processed.

CA240 T TERMINAL NAME NOT DEFINED ON TERMNAME STATEMENT

Reason – The required terminal name has been omitted on the TERMNAME control statement.

System Action – The set is not processed.

CA242 T FILE ACCESS INCONSISTENT WITH DISK-FILE STATEMENT

Reason – The file access and the file organization must correspond.

System Action – The set is not processed.

CA245 W SHR INCONSISTENT WITH ACCESS OR SQB, ASSUME NOSHR

Reason – SHR was specified for a file whose access is CO, IO, or IOU; or SHR was specified although this assignment set does not support file sharing (either because the generation did not support it, or because no SQBs were specified in the SYSTEM statement.

System Action – The set is processed as if NOSHR had been specified for the use of this file by this program.

CA246 T FILE NAME NOT PREVIOUSLY DEFINED

Reason – The required file name has been omitted.

System Action – The set is not processed.

CA247 T LANGUAGE SPECIFIED NOT SUPPORTED AT GENERATION

Reason – The language to be supported is not specified in the configuration record.

System Action – The set is not processed.

CA248 W SHAREDIO NOT VALID FOR LANGUAGE, ASSUME SHAREDIO-NO

Reason – SHAREDIO-YES is valid only for RPG II and FORTRAN programs.

System Action – The set is processed assuming SHAREDIO-NO.

CA249 T FILE ACCESS INCONSISTENT WITH PREVIOUS ACCESS

Reason – Access of IO prohibits all other types of access, or access of IOU prohibits all other types of access.

System Action – The set is not processed.

CA250 W POSSIBLE FILE ACCESS CONFLICT

Reason – The same file is to be accessed by method CA and CO.

System Action – The set is processed.

CA251 T 3270 FORMATTING KEYWORD CONFLICT

Reason – DFFPACK was specified on SYSTEM statement, but Display Format Facility was not supported at generation.

– DFF3270-YES was specified in TERMATTR statement, but DFFPACK was not specified on the SYSTEM statement.

– DFF3270-YES specified in TERMATTR statement prohibits the parameters TRANSP, ITB, VARL, SPAN and requires DATAFORM to be specified as MESSAGE.

– A DFF parameter was specified in the PROGRAM statement but no BSCA terminal referenced an attributes set that specified DFF3270-YES.

– There were inconsistent DFF parameters on the PROGRAM statement, if any DFF parameter is given, all three must be given.

System Action – The set is not processed.

CA253 W MASTER INDEX SIZE TOO SMALL, ASSUME NO MASTER INDEX

Reason – The minimum allowed is key length plus 5.

System Action – The set is processed with no master index.

CA254 T INVALID PROGRAM NAME

Reason – The program name does not consist of two to six valid extended alphameric characters. Program names are not allowed to start with $.

System Action – The set is not processed.

CA255 T MRTMAX OR REUSABLE PROHIBITS USE OF SYMBOLIC FILES

Reason – Symbolic files cannot be used if the parameters MRTMAX or REUSABLE are specified.

System Action – The set is not processed.

ASSIGNMENT LIST MESSAGES

These messages will be issued at assignment list time to inform the user of invalid specifications of operands in the assignment list control statement.

The actual format of the message is as follows:

Positions 1-5: Message number

Position 7: T, W, or I. T indicates an error was found in a statement of this assignment list; this list will not be processed by the assignment list program.

W indicates an error was detected; the statement is not processed and the next statement is read.

I indicates information only.

Positions 10-89: Message

Example

CL001 T $CCPFILE NOT INITIALIZED

The following is a list of the message numbers and messages associated with the assignment list control statements.

CL001 T $CCPFILE NOT INITIALIZED

Reason – The configuration record in $CCPFILE has been destroyed.

System Action – The statements are read through to the /* statement and then to the end of job, but are not processed.

Programmer Response – Run $CC1BF (Part of generation).

CL002 W DUPLICATE KEYWORD

Reason – The same keyword appears twice on one statement.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL003 W INVALID KEYWORD

Reason – An invalid keyword appears on the statement.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL004 W INVALID DELIMITER

Reason – A comma or blank does not follow a valid keyword or parameter.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL005 W INVALID KEYWORD PARAMETER

Reason – The parameter following the valid keyword is not valid for that keyword.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL006 W INVALID IDENTIFIER IN COL'S 1-3

Reason – The first three positions do not contain //b.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL007 W INVALID STATEMENT IDENTIFIER

Reason – The statement is not // LIST.

System – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL008 W SET ID NOT FOUND

Reason – This set is not in this $CCPFILE.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL009 W SET HAS NO REQUEST COUNT

Reason – A request count is specified for a set which does not have request counts.

System Action – The statement is not processed and the next statement is read.

Programmer Response – Correct the error and try again.

CL010 T CCP BEING RUN IN OTHER LEVEL

Reason — The other programming level in a DPF system is running a CCP or a CCP-related program.

System Action — The statements are read through to the /* statement and then to the end of job, but are not processed.

Programmer Response — Wait until the job in the other programming level has completed, then run assignment list.

CL011 I THIS $CCPFILE CONTAINS NO SETS

Reason — This informational message is given when $CCPAL is run with a request to list all the sets and there are no sets in the $CCPFILE requested. They have not been assigned or have all been deleted.

System Action — Read the next statement.

Programmer Response — No response is required. This message is merely to explain why no sets were listed.

U– HALTS FROM THE SCP GENERATOR

Subidentifier GE Generation Error

Reason — Error occurred during generation.

Recovery — Control cancel (option 2). Contact IBM for programming support.

Subidentifier GV Invalid Call

Reason — Invalid call.

Recovery — Immediate cancel (option 3). Contact IBM for programming support.

Subidentifiers OB, SC, and WK

Reason — Work file space is too small or not available.

OB — Object file space is too small. Increase the space specification on the $WORK FILE card.

SC — End-of-file encountered on source file. Contact IBM for programming support.

WK — Work file space too small or is not available. If not using $WORK2 FILE card, add one. If using $WORK2 FILE card, increase the space specification.

Recovery — Immediate cancel (option 3).

U– HALT FROM $CC1BF

Subidentifier or Subhalt AF Permanent Disk File Error

Reason — In writing the initial form of the file $CCPFILE, an uncorrectable disk error occurred.

Recovery — Immediate cancel (option 3). Run the alternate track assignment program. Then rerun $CC1BF.

Subidentifier or Subhalt PU Unable to Allocate System Punch Device

Reason – The system punch device could not be allocated. Either it is in use by the other program level in a DPF system, or the system has an internal error.

Recovery – Control cancel (option 2). Immediate cancel (option 3). Perform one of the following:

- If the device is in use by the other program level, run $CC1PP after the other level completes processing and goes to end of job.
- If the system has an internal error, contact IBM Field Engineering for program support.

Subidentifier or Subhalt FE Permanent Disk File Error

Reason – In reading from the file $SOURCE, which contain the output from the first pass of generation, an uncorrectable disk error was encountered.

Recovery – Control cancel (option 2). Immediate cancel (option 3). Run the alternate track assignment program. Run the $DELET program to delete the file $SOURCE. Then begin generation again with the // LOAD $CPXDV, ## step.

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 (Appears identical to a 2741)
- 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 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
- 5410 Model A16 Processing Unit with 32,768 bytes of main storage if 3270 DFF is to be supported
- 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 (available as an RPQ only)
- 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 cannot 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 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).

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 E: 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 terminal is released by a Release 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 data 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.

data terminal. See *non-command terminal.*

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 requesting terminal program. A type of application program under the CCP that can process requests from more than one requesting 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.

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). Also referred to as data terminal.

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. An 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 requested.

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 request count. The optional CCP program management function of maintaining a count of the number of times each application program is requested.

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 *requesting 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).

requesting 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*). Requesting 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 sign-on command, or entering the sign-on command with a password or other user-specified security data.

single requesting terminal program. A type of application program under the CCP that can process a request from only one requesting 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 system services.

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.

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.

MAIN STORAGE ESTIMATES FOR THE CCP

The following tables are designed for calculating the core storage requirements of an executable CCP system during the planning phase for a CCP installation. The CCP system core requirements are determined by options specified during the CCP generation and assignment. Additional core is required in the User's Program Area if the Display Format Facility (DFF) is utilized to support the 3270 terminal system.

The core requirements are organized into four tables:

Table 1 - Base System Size

Table 2 - Core requirements derived from CCP generation options

Table 3 - Core requirements derived from CCP assignment options (additionally, core layout is given)

Table 4 - Core requirements derived from DFF generation

Tables 2 through 4 are constructed as a list of options with the corresponding core (given in number of bytes) needed for that option. The option, so far as possible, is given as a keyword with a reference to a particular statement for CCP generation or assignment.

The tables are built in a manner that the sum of the core for the options chosen should be added to the appropriate base system size (Table 1) to determine the total executable CCP system core requirements. If any part of a listed option meets the requirements of the desired system configuration, then the core on that line in the table should be added to the system size. However, for Table 2 options which are *indented* are to be included in the system size only if the desired system configuration also includes the preceding *non-indented* option.

It should be noted that the User's Program Area will contain the user's program, any appropriate DSM modules included at linkage edit of the application program, and the core requirements of DFF if that support is desired on the system being configured.

MLTA Minimum System Residence (MINRES-YES, $EGEN)	**Bytes**
CCP	8459
MLTA IOCS	4235
Common Check Routine	546
BSCA Minimum System Residence (MINRES-YES, $EGEN)	
CCP	9680
BSCA IOCS	3999
Common Check Routine	546
Both MLTA and BSCA Minimum System Residence (MINRES-YES, $EGEN)	
CCP	11293
BSCA IOCS	3999
MLTA IOCS	4235
Common Check Routine	546

Table 1. Base System Size

Options and Resultant Core Required (add to the appropriate Base System Size, Table 1)	Bytes
DPF-YES ($EFAC)	52
MINRES-NO ($EGEN)	276
MAXEUP-2 to 8 ($EFAC)	65
MAXEUP-2 to 8 ($EFAC) and/or FORMAT-YES ($EFAC)	-93
FSHARE-YES ($EFAC)	642
D5445-(YES) ($EIOD)	71
LANG-RPG II ($EPLG)	3
LANG-RPG II ($EPLG) or FORMAT-YES ($EFAC)	3
ESCAPE-(YES) ($EFAC)	43
Both $EMLA and $EBSC	3
Both $EMLA and $EBSC	216
$EMLA	
ESCAPE-(YES) ($EFAC)	85
TYPE-(Station Control Terminals) ($EMLD)	758
TYPE-(No Checking-Basic Terminals) ($EMLD)	242
TYPE-(Checking Terminals) ($EMLD)	228
TYPE-(Dialed Feature Terminals) ($EMLD)	311
TYPE-(Buffered Terminals) ($EMLD)	35
TYPE-(1050 Terminals) ($EMLD)	81
TYPE-(2741 Terminals) ($EMLD)	39
XLATE-NO ($EMLA)	85
9 x [number of line code types (XMCODE, $EMLD)]	
TYPE-1050D ($EMLD)	51
TYPE-2740 ($EMLD)	51
TYPE-2740C and/or SYS7C ($EMLD)	51
TYPE-2740S and/or 2740M2S and/or 2740M2SB ($EMLD)	51
TYPE-1050 and/or 2740SC and/or 2740M2SC and/or 2740M2SCB and/or SYS7SC ($EMLD)	51
TYPE-2740D ($EMLD)	51
TYPE-2740DC and/or SYS7DC ($EMLD)	51
TYPE-2740DT ($EMLD)	51
TYPE-2740DTC ($EMLD)	51
TYPE-2741 ($EMLD)	51
TYPE-2741D and/or CMCSTD ($EMLD)	51

Table 2. Core Requirements Derived From CCP Generation Options (Part 1 of 2)

Options and Resultant Core Required (add to the appropriate Base System Size, Table 1) (Continued)	Bytes
$EBSC	
FORMAT-YES ($EFAC)	381
MINRES-NO ($EGEN)	57
RESPOL-YES ($EBSC)	1792
AUTORS-YES ($EBSC)	139
ESCAPE-(YES) ($EFAC)	44
TYPE-(3270) ($EBSD)	16
TYPE-CPU ($EBSD)	6
PP-YES ($EBSC) and/or MP-YES ($EBSC)	6
CS-YES ($EBSC)	68
MINRES-NO ($EGEN)	139
PP and/or MP and/or Dial ($EBSC)	6
CS-YES and/or PP and/or MP and/or Dial ($EBSC)	9
CS-YES and/or Dial ($EBSC)	6
PP and/or MP ($EBSC)	3
TYPE-(3270) ($EBSD)	180
ESCAPE-(YES) ($EFAC)	33
XPRNCY-YES ($EBSC)	24
ITB-YES ($EBSC)	50
XPRNCY-YES ($EBSC)	58
ASCII-YES ($EBSC)	74
TYPE-(3270) ($EBSD)	24
BSCA-2 ($EBSC)	203

Table 2. Core Requirements Derived From CCP Generation Options (Part 2 of 2)

The specifications made in an assignment set represent the standard operating environment for a CCP run. Certain resources specified in an assignment set as being available to the CCP run can, on an exception basis, be suppressed by the system operator at the startup of the CCP run. Such a suppression reduces the control block core requirements defined in Table 3.

The CCP components listed in Table 3 are in the same sequence as positioned in core.

CCP Components	Internal Terminology	Details for Estimating Core Requirements
DSM Supervisor	–	
Resident CCP	$CC4	(Generation core requirements from Table 2)
CCP Trace (optional)	CCTR	Minimum is zero bytes. If TRACEBLK value is greater than zero in assignment or startup: 896 + 256 x (TRACEBLK value, SYSTEM statement)
BSCA Trace (optional)	CCBS	(549 bytes if TRACEMLMP specified at startup)
MLTA Trace (optional)	CCML	(1433 bytes if TRACEMLTA specified at startup)
Program Request Count Table (optional)	–	(If PGMCNT-YES at generation: 2 x (number of PROGRAM statements in this assignment set))

Table 3 (Part 1 of 3). Core Requirements Derived From CCP Assignment Options

CCP Components	Internal Terminology	Details for Estimating Core Requirements	
Terminal Attribute Table	Terminal Attribute Set	TAS 5 x (highest user ATTRID number, TERMATTR statement)	_____
Control Information and Buffer Space for each Communications Line		The following control blocks and data areas for MLTA and/or BSCA lines are positioned in core in the same sequence as the MLTALINE and/or BSCALINE statements were specified in the assignment stage. **MLTA** Adapter DTF (one only for all MLTA lines) 33 bytes	_____
		Line 1 Line 2 Line 3 Line 4 Line 5 Line 6 Line 7 Line 8	
	Define The File/Line Control Block	DTF/LCB (one for each MLTALINE statement) 125 bytes ___ ___ ___ ___ ___ ___ ___ ___	
	Statistical Data Recording Area	SDR 5 x (number of MLTATERM statements this MLTALINE statement) ___ ___ ___ ___ ___ ___ ___ ___	
	Polling List	6 + (3 x number of entries in POLLLIST, MLTALINE statement) ___ ___ ___ ___ ___ ___ ___ ___	
	Line Buffer	MAXRECL value, MLTALINE statement ___ ___ ___ ___ ___ ___ ___ ___	
		MLTA Line Totals ___ + ___ + ___ + ___ + ___ + ___ + ___ + ___ =	_____
		BSCA	
	Define The File/ Line Control Block	DTF/LCB (one for each BSCALINE statement) 136 bytes ___ ___	
	ASCII Translation Buffer	(only if XMCODE-ASCII, BSCALINE statement) The size of *one* single line buffer as calculated below; do *not* double the value here if DBLBUF-YES, BSCALINE statement ___ ___	
	Line Buffer	The largest BLKL value used on this line + C + ITB count (double if DBLBUF-YES, BSCALINE statement) ___ ___ C = 44 if ITB-Y *and* TRANSP-Y, TERMATTR statement = 42 otherwise ITB count = 0 if ITB-N, TERMATTR statement = 1 if ITB-Y, TRANSP-N, TERMATTR statement = 5 if ITB-Y, TRANSP-Y, TERMATTR statement	
	—	IDEXSEND bytes (if TYPE-SW, BSCALINE statement) ___ ___	
	—	IDEXRCV bytes, BSCATERM statement (if TYPE-SW, BSCALINE statement) ___ ___ 4 + (each IDEXRCV length + 3)	
	Addressing List	If TYPE-CS, BSCALINE statement (number of BSCATERM statements) x (bytes of ADDRCHAR characters + 3) + 1 ___ ___	
	Polling List	If TYPE-CS, BSCALINE statement (number of entries in POLLLIST, BSCALINE statement) x (bytes of POLLCHAR + 3) + 1 ___ ___	
	Statistical Data Recording Area	SDR 4 + (number of BSCATERM statements) x 2 x (7 + bytes of POLLCHAR) ___ ___	
		BSCA Line Totals ___ + ___ =	_____
	Checklist	3 x (number of MLTALINE and BSCALINE statements)	_____

Table 3 (Part 2 of 3). Core Requirements Derived From CCP Assignment Options

CCP Components	Internal Terminology	Details for Estimating Core Requirements
Control Information for each Terminal	Terminal Unit Block	TUB 51 x (number of MLTATERM and BSCATERM statements) + 1 _____
	Terminal Name Table	TNT 23 + (11 x number TERMNAME statements) _____
Control Information for each User Task	Task Control Block	TCB 74 x (MAXEUP value, SYSTEM statement) _____
	Contents Directory Entry	CDE 24 x (MAXEUP value, SYSTEM statement) _____
Control Information for User Programs	Program Characteristics Table	PCT Number of PROGRAM statements rounded up to the next multiple of 6 _____
Control Information for each Disk Data File Used	Short DTF For File	SDF Choose the appropriate following calculation lines according to DISKFILE statement organizations in this assignment set. The control blocks are positioned in core in the same sequence as the DISKFILE statements were specified in the assignment stage. Direct (5444) 29 x ____ (number of this type of disk file) = ____ Direct (5445) 32 x ____ (number of this type of disk file) = ____ Consecutive (5444) 30 x ____ (number of this type of disk file) = ____ Consecutive (5445) 32 x ____ (number of this type of disk file) = ____ Index (Load)(5444) 40 x ____ (number of this type of disk file) = ____ Index (Load)(5445) 43 x ____ (number of this type of disk file) = ____ Index (Random)(5444) 45 x ____ (number of this type of disk file) *See Note.* = ____ Index (Random)(5445) 47 x ____ (number of this type of disk file) = ____ *Note:* If MIXSIZE, DISKFILE statement is specified for an Index Random file, put the sum of all MIXSIZE values here. (Each master index follows, in core, the short DTF for that file.) = ____ Total _____
Control Information for Symbolic File	Symbolic File Table	XDT (One table per SYMFILE statement) Table = 11 + n n = number of DISKFILEs referenced by a SYMFILE statement Sum of all SYMFILE statement Tables +1 _____
CCP Calculated Master Indexes	–	One master index for each DISKFILE statement using the MSTRINDX-YES, DISKFILE statement Each master index length is: (KEYL + 2) x (number of disk *cylinders* in index) + 3 Sum for all master indexes _____
Control Information for Sharing of Files	Sector Queue Blocks	SQB 12 x (number of SQBs, SYSTEM statement) _____
Control Information for Associating Symbolic Files	File Specification Blocks	FSB 4 x (number of FSBs, SYSTEM statement) _____
Dynamic TP Buffer Hold Area	–	At least equal to MINTPBUF, SYSTEM statement in assignment (may be increased at the end of startup) _____
User Program Area	–	At least equal to MINUPA, SYSTEM statement in assignment (may be increased at the end of startup). Minimum is 5K. Always on 256-byte boundary. _____

Table 3 (Part 3 of 3). Core Requirements Derived From CCP Assignment Options

If the Display Format Facility (DFF) is used, space in the user program area is occupied by that facility while (and only while) a user program which utilizes that facility is executing. Table 4 defines the space requirements in the user program area for DFF.

If any currently executing user program uses DFF	DFF Control Routine	4096 bytes
For each BSCA line using DFF	Output Hold Area (one per line) Line1_____ + Line2_____ = ____ bytes No Blocking: If length of output text of largest format used on system is less than 256, then 256 bytes If length of output text of largest format used on system is greater than 256, round up to the next multiple of 256 Blocking: Minimum length of output hold area is 512 bytes. Must be increased by increments of 256 bytes (up to length of output text of largest format used).	
For any currently executing user program utilizing DFF	Program Appended Storage Terminal Table and Format Table (Minimum 256 bytes)	____ bytes
	Contains enough space for one of the following combinations of Terminal Table and Format Table: 1 terminal + (1 to 5 formats) 2 terminals + (1 to 3 formats) 3 terminals + 1 format Otherwise, use this formula for calculating Terminal Table and Format Table space: (37 x T) + (18 x M) + 127 rounded up to the next multiple of 256 T = DFFMTERM, PROGRAM statement M = DFFNDF, PROGRAM statement	
	Field Descriptor Table (DFFSFDT, PROGRAM statement) rounded to the next multiple of 256	____ bytes
	Decimal length of FDT is calculated: For the first 1 to 17 fields, 256 bytes For each 18 additional fields or a fraction thereof, an additional 256 bytes	

Table 4. Core Requirements (In User Program Area) Derived From DFF Generation

DISK STORAGE ESTIMATES FOR THE CCP

Space occupied on the production pack (pack on 'CCUNIT' specified in $EGEN generation control statement). All space defined is in the object library:

Base number	1104 sectors	_____
If MLTA supported, add	48 sectors	_____
If BSCA supported, add	48 sectors	_____
If DFF supported, add	48 sectors	_____
	Total	_____

Space occupied on a program preparation pack (pack on 'PPUNIT' specified in $EPLG generation control statement):

For LANG-ASSEM, 190 sectors in source library	_____
For LANG-RPGII, 9 sectors in object library	_____
For LANG-COBOL, 2 sectors in object library	_____
For LANG-FORTRAN, 2 sectors in object library	_____
On a program preparation pack, if unit record devices are to be used by application programs (in $EIOD statement, CARD-1442/MFCU/'MFCU 1442' or PRINTR-5203/1403:	
Add 18 sectors in object library if RPG II is supported on the pack	_____
Add 18 sectors in object library if COBOL, and/or FORTRAN and/or Basic Assembler is supported on the pack	_____
Total	_____

Size of $CCPFILE (determined from specifications in $EFIL generation control statement and generation statements defining lines and terminals):

Determine the number of sectors per assignment set as follows:

Start with one sector	1
For TERMS-n: 1 - 14 terminals, add 3 sectors	
15 - 22 terminals, add 4 sectors	
23 - 28 terminals, add 5 sectors	
29 - 42 terminals, add 6 sectors	
43 - 44 terminals, add 7 sectors	
45 - 51 terminals, add 8 sectors	_____
For BSCA-n ($EBSC) and LINES-n ($EMLA):	
Add one sector if the sum of BSCA-n plus LINES-n is less than seven; add two sectors if the sum of BSCA-n plus LINES-n is seven or greater	_____
If any switched lines are to be supported, add one sector	_____
For DFILES-n, add one sector per 12 or fraction thereof	_____
For PROGS-n, add one sector per 5 or fraction thereof	_____
Total	_____

Determine the space for *all* assignment sets by multiplying the total above by the number specified for SETS-n. To this product add 39 sectors. _____

Determine the number of sectors required by each storage dump.

If CORE-24K, then 96 sectors	
If CORE-32K, then 144 sectors	
If CORE-48K, then 192 sectors	
If CORE-64K, then 264 sectors	_____

Multiply the above number of sectors by the number of dumps specified (DUMPS-n) for the total number of sectors allocated for core dump space. +_____

Add the number of sectors specified for the CCP trace to disk (TRACE-n, multiply the number of tracks specified by 24 to obtain the number of sectors). +_____

The sum is the total number of sectors allocated for $CCPFILE _____

Work file space required during generation: Three work files are used, the filenames and the sectors required are:

For $SOURCE, 2880 sectors _____

For $WORK, 960 sectors _____

For $WORK2, 960 sectors _____

Total _____

Appendix G. Installation Verification Program

The installation verification program (CCPIVP) is furnished as a load (O) module and as a (R) module on the CCP PID pack. CCPIVP is a single requester program allowing data with the program request. The program communicates with the console (5471) after being requested.

Because of the design of the program, it can be requested from a command terminal as well as the console. However, once CCPIVP is loaded, it communicates solely through the console. If CCPIVP is requested by a terminal rather than the console, the requester is released immediately.

Functions that can be exercised by CCPIVP include:

1. Load from the console.
2. Release of a requesting terminal (other than the console).
3. Data with the request.
4. Single requester programs.
5. Program queuing.
6. Program request resource allocation.
7. Symbolic files including the File command, if specified via assignment.
8. Use of the printer under the CCP, if $$LPRT is link edited and PRINTER-YES is specified on the PROGRAM assignment control statement.
9. Use of Put Wait NNL, Put then Get NNL and Put No Wait to the console.
10. Allocation, Open, Close of a 5444 consecutive file under the CCP. Based upon input from the console:
    a. CGIVFILE is allocated, opened as a consecutive output 5444 file.
    b. Records are written to the file.
    c. CGIVFILE is closed.
    d. CGIVFILE is opened as an input file.
    e. CGIVFILE is dumped to the console, or to the printer if the optional link edit was performed.
    f. CGIVFILE is closed.
11. On unexpected return codes (non 00 for Puts, non 00 or non 01 on Put then Gets), the ability to retry the operation.
12. Concurrent execution of more than one copy of CCPIVP provided symbolic files are used and the program does not use the printer.
13. Use of the console to enter operator commands or to communicate with the program.
14. Closing of CGIVFILE at shutdown.

The sample assignment deck includes the PROGRAM and DISKFILE statements necessary to run CCPIVP. They can be modified according to the user's configuration.

After generation is completed, an Assignment Build program must be run including a DISKFILE, SYMFILE, and PROGRAM statement for CCPIVP in the assignment set. The CCPIVP load (O) module can be executed under the CCP as it exists, but in this form, supports only the console as the output printer. If the user elects to do the optional link edit to allow line printer support to be included, then the load (O) module from the output of the link edit will replace the provided load module. The following is the printout of the overly linkage editor core usage map and cross reference list:

START ADDRESS	CATEGORY	NAME AND ENTRY	CODE LENGTH HEXADECIMAL	DECIMAL	REFERENCED BY
1500	0	CCPIVP	0C0C	3084	
210C	2	$$CSOP	001D	29	CCPIVP
2129	2	$$CSIP	0027	39	CCPIVP
2150	2	$$SRBR	0079	121	$$CSIP $$CSOP
21C9	2	$$SRUA	0026	38	$$CSIP $$CSOP
21EF	2	$$SRDF	001C	28	$$CSOP
220B	2	$$SRTC	001C	28	$$CSIP $$CSOP $$SRDI
220B		DMSRLU			
221C		DMSRTC			$$CSIP $$CSOP
221F		DMSRER			$$CSIP $$CSOP $$SRDI
2227	2	$$SRMO	0081	129	$$SRBR
22A8	2	$$SRSB	0043	67	$$SRBR
22EB	2	$$SRDI	0038	56	$$SRSB $$SRBR
230A		DMSRPD			$$SRSB
2303		DMSRRD			$$SRSB
2323	2	$$SRBP	002F	47	$$SRSB $$SRUA

```
OL100 I  THE TOTAL CORE USED BY CCPIVP IS  3666 DECIMAL.
OL101 I  THE START CONTROL ADDRESS OF THIS MODULE IS 1500.
OL104 I  TOTAL NUMBER OF LIBRARY SECTORS REQUIRED IS   16
         NAME-CCPIVP,PACK-CCPOBJ,UNIT-R2,RETAIN-P,LIBRARY-U
```

Application | Date

Program Name | Number | Programm

OCL STATEMENTS

```
1   4   8   12  16  20  24  28  32  36  40  44  48  52  56  60  64  68  72  76
// LOAD $OLINK,XX
// FILE NAME-$SOURCE,TRACKS-10,RETAIN-S,PACK-XXXXXX,UNIT-XX
// FILE NAME-$WORK,TRACKS-10,RETAIN-S,PACK-XXXXXX,UNIT-XX
// RUN
// PHASE NAME-CCPIVP,UNIT-XX,RETAIN-R
// OPTIONS MAP-XREF
// INCLUDE NAME-CCPIVR,UNIT-XX
// INCLUDE NAME-$$CSIP,UNIT-XX
// INCLUDE NAME-$$CSOP,UNIT-XX
// EQUATE OLDNAME-$$LPRT,NEWNAME-$NLPRT
// EQUATE OLDNAME-@@LPRT,NEWNAME-$$LPRT
// INCLUDE NAME-$$LPRT,UNIT-XX
// INCLUDE NAME-$NLPRT,UNIT-XX
// END
```

pack to which assignment indicates CCPIVP resides (CCP production pack)

pack containing the DSM disk data management modules

pack containing the CCP unit record modules

optional and specified if the printer is to be used in CCPIVP

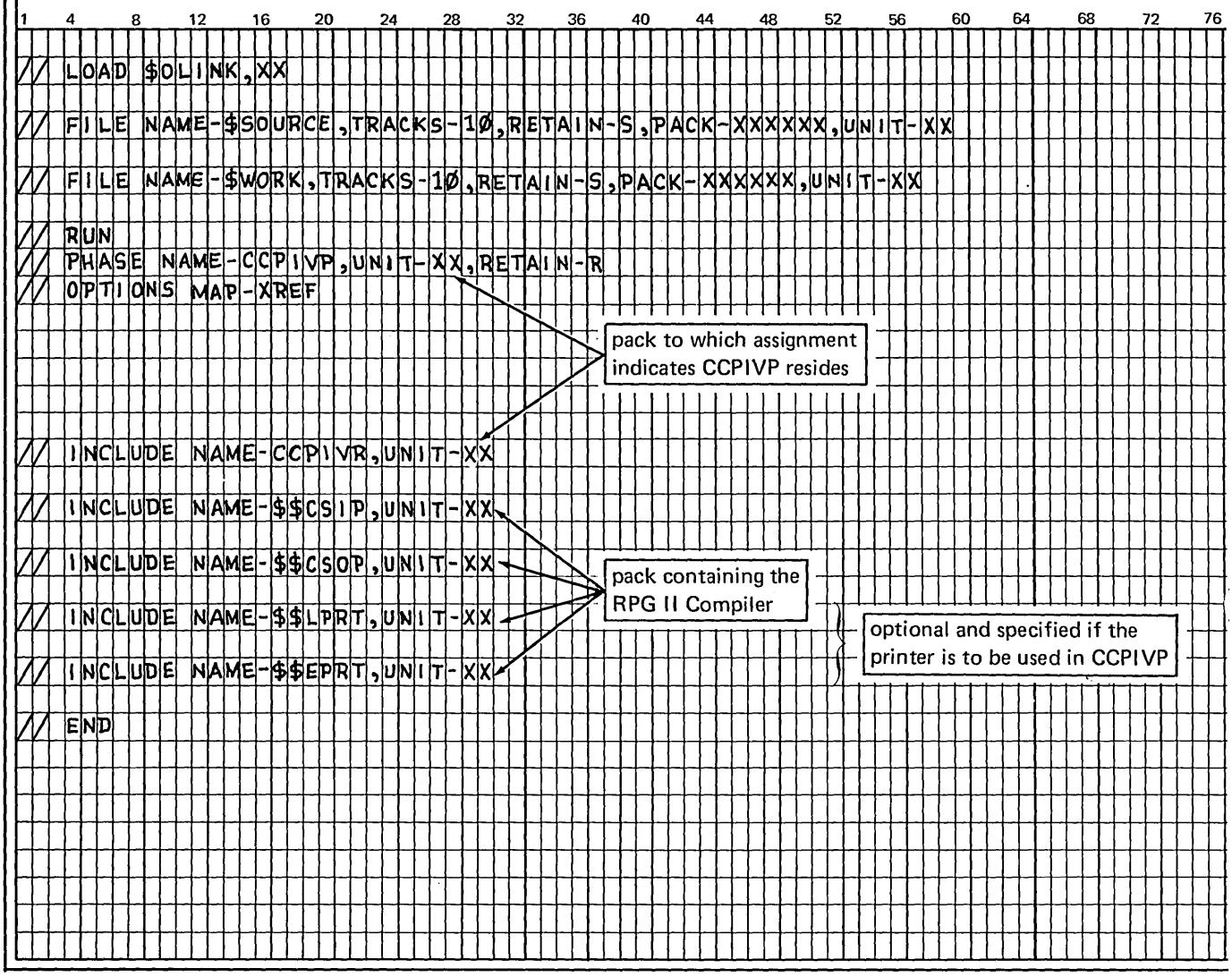
Example 1. Link Editing CCPIVP with a Non-RPG II Pack (with ASSEM, COBOL, or FORTRAN)

Application | Date

Program Name | Number | Programm

OCL STATEMENTS

```
// LOAD $OLINK,XX
// FILE NAME-$SOURCE,TRACKS-10,RETAIN-S,PACK-XXXXXX,UNIT-XX
// FILE NAME-$WORK,TRACKS-10,RETAIN-S,PACK-XXXXXX,UNIT-XX
// RUN
// PHASE NAME-CCPIVP,UNIT-XX,RETAIN-R
// OPTIONS MAP-XREF
// INCLUDE NAME-CCPIVR,UNIT-XX
// INCLUDE NAME-$$CSIP,UNIT-XX
// INCLUDE NAME-$$CSOP,UNIT-XX
// INCLUDE NAME-$$LPRT,UNIT-XX
// INCLUDE NAME-$$EPRT,UNIT-XX
// END
```



Example 2. Link Editing CCPIVP with a RPG II Pack (with or without ASSEM, COBOL, or FORTRAN)

Loading the CCP to Run CCPIVP

// LOAD $CCP,xx

// FILE (For each actual file to be used by CCPIVP)
See the assignment deck actually used to determine the NAME-name entry.

Note: CCPIVP uses the file name CGIVFILE in its disk DTFs.

// RUN

Procedure for Requesting CCPIVP

1. /FILE CGIVFILE, actual file name (if symbolic files are used)

2a. To load from the console:

.CCPIVPbmm/dd/yy

8 char

2b. To load from a terminal:

CCPIVPbmm/dd/yy

8 char

Operating Instructions with CCPIVP

1. Enter data as prompted from the console messages.

2. Stand-alone halts are issued by CCPIVP to provide the user a place to catch a non-normal situation while running CCPIVP.

Halts Issued by CCPIVP

Halt U0

Reason – This copy of CCPIVP was requested by a terminal and the release operation code failed.

Recovery – 1. Press the HALT/RESET or START button; CCPIVP will go to end-of-job.

2. Take a dump of the main storage contents by pressing SYSTEM RESET and START, or by using the appropriate stand-along core dump program.

Halt U1

Reason – Return code other than 00 or 01 received from accept input operation (only can happen on data program request).

Recovery – 1. Press the HALT/RESET or START button; error message is written, input data required.

2. Take a dump of the main storage contents by pressing SYSTEM RESET and START, or by using the appropriate stand-alone core dump program.

Note: If you enter the correct data and length of data and get error messages from CCPIVP, this may indicate that the CCP is passing invalid information in the parameter list or record area.

Halts U2 - UA

Reason — An unexpected return code non 00 received on an output operation or a non 00 or 01 received on an input operation. U2 through UA correspond to a particular operation issued in the program at a particular time.

Recovery — 1. Press the HALT/RESET or START button; completion code is printed on the console and input required.

2. Take a dump of the main storage contents by pressing SYSTEM RESET and START, or by using the appropriate stand-alone core dump program.

After pressing the HALT/RESET or START button on a U2 through UA halt, the following is printed:

TNAME-ssssss, UNEXPCTD RET CODE rrrr ENTER TA OR xx

rrrr = return code in hexadecimal
TA = operation retried
xx = any characters other than TA; program goes to EOJ
ssssss = symbolic terminal name of console

Halt Ud

Reason — An error has occurred using the disk file and control has returned to the application program indicating a non-recoverable condition.

Recovery — 1. Press the HALT/RESET or START button; CCPIVP goes directly to EOJ.

2. Take a dump of the main storage contents by pressing SYSTEM RESET and START, or by using the appropriate stand-alone core dump program.

Halt UP

Reason — An error has occurred in using the printer and control has returned to CCPIVP indicating a non-recoverable error.

Recovery — 1. Press the HALT/RESET or START button; CCPIVP goes directly to EOJ.

2. Take a dump of the main storage contents by pressing SYSTEM RESET and START, or by using the appropriate stand-alone core dump program.

Appendix H. External Description of the CCP Trace Facility

As an option selected at each operational startup, the CCP will maintain an in-core circular trace of function requests and the initiation of those functions.

Each trace entry is 16 bytes in length, aligned in main storage on a 16-byte boundary. Each entry represents an action on the part of the CCP or the user program. The first byte of each entry represents the action taken; the second byte indicates the task by which, or for which, the action was taken.

The action is determined by the right half of the first byte as follows:

1 - Task entered the CCP dispatcher
2 - Task issued branch to DSM disk I/O start
3 - Task issued branch to DSM disk I/O wait
4 - Branch to DSM disk I/O wait made by the CCP for task
5 - Task issued branch to DSM general entry
6 - Not used
7 - Task posted another task that an event awaited is complete
8 - Task issued a request to get main storage
9 - Task issued a request to free main storage
A - Task issued a request to get a completed communications operation
B - Task started communications operation
C - Task issued communications operation request
D - System task issued communications operation request
E - Return to user task upon a completed communications operation

The left half of the first byte is normally zero; however, to indicate a certain status of the trace table, other bits are used.

X'20' — This bit will be on in the first byte of the last entry of the **in-core** trace each time it is written to $CCPFILE. If this bit is on in the first entry of the trace table dumped from disk, it indicates that the trace area on disk has been wrapped.

X'40' — This bit is set on in the first byte of the last entry that was traced prior to a Trace Off or Shutdown command.

The task by which, or for which, the action was performed is determined by the second byte as follows:

C3 - Communications management
E3 - Termination
D7 - Command Processor
E2 - Shutdown
F1 through F8 - User task 1 through user task 8

When the in-core CCP trace has been selected at startup, the system operator (or IBM Customer Engineer) can initiate the writing of the in-core trace table to disk by entering a Trace On command. For a complete description of the Trace command, see *IBM System/3 Disk System Communications Control Program System Operator's Guide,* GC21-7581.

The disk area to which trace entries are written is a part of $CCPFILE. When trace-to-disk has been initiated, the CCP trace routine writes its in-core trace table to the next available disk sectors every time that in-core table becomes full. When the trace area on disk is exhausted, the trace routine wraps around to resume the writing of the in-core trace table at the beginning of the trace area on disk.

When trace to disk is terminated by the system operator, a stopper bit is placed into the last trace entry written to disk, and an indication is incorporated in that entry indicating whether or not a wrap-around on disk occurred.

When trace is dumped from disk, it is dumped in the sequence that the actions occurred.

In main storage, the beginning of the trace entries can be found within a dump as follows:

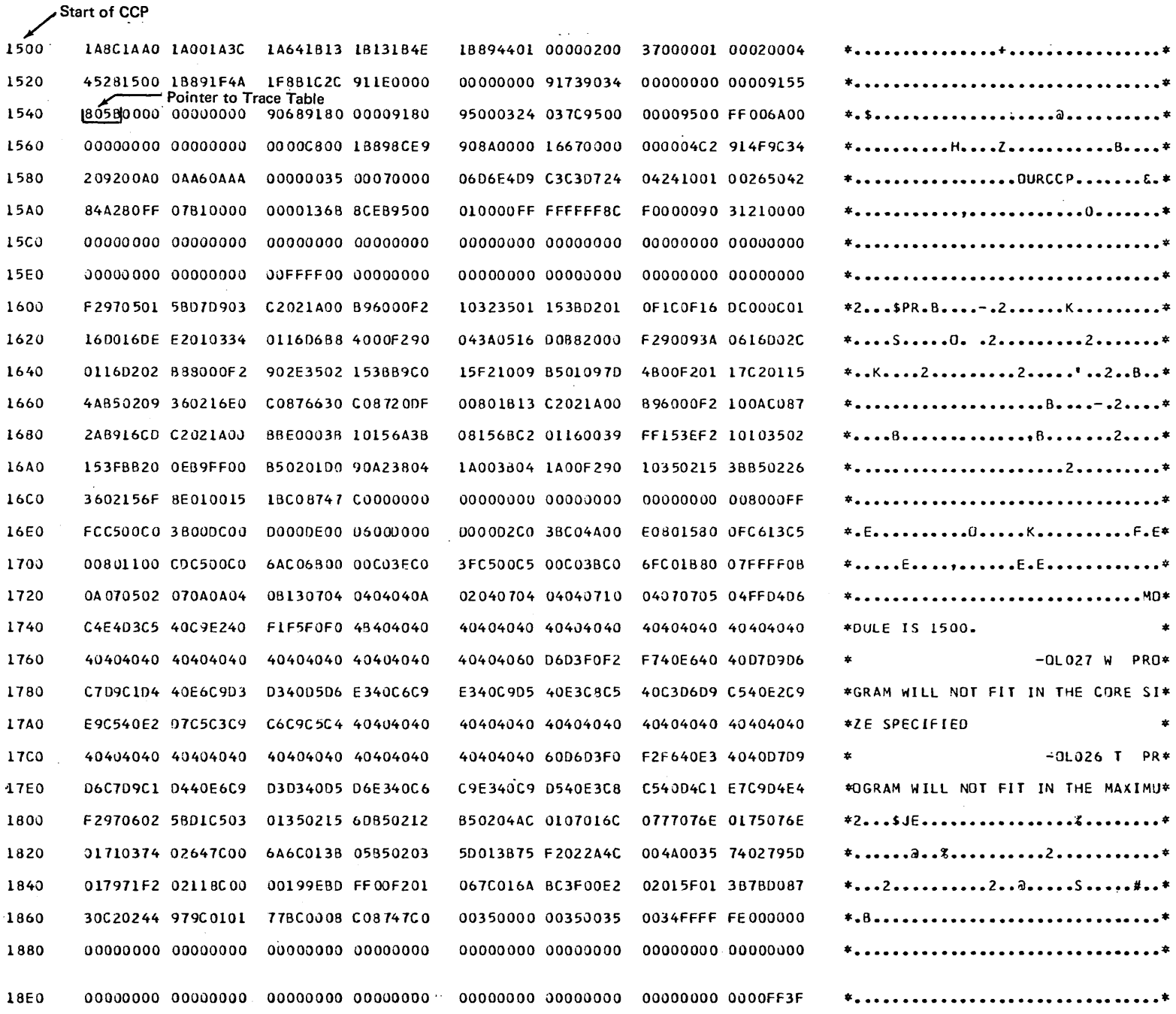
1. Find the beginning of the CCP program level.

2. In bytes X'40' and X'41' of the program level, find the address of the trace control block.

3. At that location minus one, find the first byte of a 6-byte table, containing respectively:

   - The 2-byte address of the last byte of the first entry in core.
   - The 2-byte address of the byte following the last entry in core.
   - The 2-byte address of the last byte of the most recent entry in core; once again, the trace is a circular table in main storage, the last entry in core being followed by the earliest entry in core.

The following is a sample dump of a CCP:




```
1500  1A8C1AA0 1A001A3C 1A641B13 1B131B4E 1B894401 00000200 37000001 00020004  *.............+.............*
1520  45281500 1B891F4A 1F8B1C2C 911E0000 00000000 91739034 00000000 00009155  *.............................*
1540  805B0000 00000000 90689180 00009180 95000324 037C9500 00009500 FF006A00  *.$.................@.........*
1560  00000000 00000000 0000C800 1B898CE9 908A0000 16670000 000004C2 914F9C34  *.........H....Z...........B....*
1580  209200A0 0AA60AAA 00000035 00070000 06D6E4D9 C3C3D724 04241001 00265042  *.................OURCCP.......&.*
15A0  84A280FF 07B10000 0000136B 8CEB9500 010000FF FFFFFF8C F0000090 31210000  *..........,...........0.......*
15C0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *.............................*
15E0  00000000 00000000 00FFFF00 00000000 00000000 00000000 00000000 00000000  *.............................*
1600  F2970501 5BD7D903 C2021A00 B96000F2 10323501 153BD201 0F1C0F16 DC000C01  *2...$PR.B....-.2......K.........*
1620  16D016DE E2010334 0116D6B8 4000F290 043A0516 D0B82000 F290093A 0616D02C  *....S.....O. .2.........2.......*
1640  0116D202 B88000F2 902E3502 153BB9C0 15F21009 B501097D 4B00F201 17C20115  *..K....2.........2.....'..2..B..*
1660  4AB50209 360216E0 C0876630 C0872ODF 00801B13 C2021A00 B96000F2 100AC087  *.....................B....-.2....*
1680  2AB916CD C2021A00 BBE0003B 10156A3B 08156BC2 01160039 FF153EF2 10103502  *....B.............,B.......2....*
16A0  153FBB20 0EB9FF00 B50201D0 90A23804 1A003B04 1A00F290 10350215 3BB50226  *.....................2.........*
16C0  3602156F 8E010015 1BC08747 C0000000 00000000 00000000 00000000 008000FF  *.............................*
16E0  FCC500C0 3B00DC00 D000DE00 D600D000 D000D2C0 3BC04A00 E0801580 0FC613C5  *.E.........O.....K..........F.E*
1700  00801100 CDC500C0 6AC06B00 00C03EC0 3FC500C5 00C03BC0 6FC01B80 07FFFF0B  *.....E....,.....E.E.............*
1720  0A070502 070A0A04 0B130704 0404040A 02040704 04040710 04070705 04FFD4D6  *.............................MO*
1740  C4E4D3C5 40C9E240 F1F5F0F0 4B404040 40404040 40404040 40404040 40404040  *DULE IS 1500.                 *
1760  40404040 40404040 40404040 40404040 40404060 D6D3F0F2 F740E640 40D7D9D6  *                    -OL027 W  PRO*
1780  C7D9C1D4 40E6C9D3 D340D5D6 E340C6C9 E340C9D5 40E3C8C5 40C3D6D9 C540E2C9  *GRAM WILL NOT FIT IN THE CORE SI*
17A0  E9C540E2 07C5C3C9 C6C9C5C4 40404040 40404040 40404040 40404040 40404040  *ZE SPECIFIED                  *
17C0  40404040 40404040 40404040 40404040 40404040 60D6D3F0 F2F640E3 4040D7D9  *                    -OL026 T  PR*
17E0  D6C7D9C1 D440E6C9 D3D340D5 D6E340C6 C9E340C9 D540E3C8 C540D4C1 E7C9D4E4  *OGRAM WILL NOT FIT IN THE MAXIMU*
1800  F2970602 5BD1C503 01350215 6DB50212 B50204AC 0107016C 0777076E 0175076E  *2...$JE.............%........*
1820  01710374 02647C00 6A6C013B 05B50203 5D013B75 F2022A4C 004A0035 7402795D  *......@..%.........2..........*
1840  017971F2 02118C00 00199EBD FF00F201 067C016A BC3F00E2 02015F01 3B7BD087  *...2.........2..@.....S.....#..*
1860  30C20244 979C0101 77BC0008 C08747C0 00350000 00350035 0034FFFF FE000000  *.B.............................*
1880  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000  *.............................*

18E0  00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000FF3F  *.............................*
```

```
7F20   80548058 F2011238 808059F2 900B3501   80583601 80357A20 01350180 5D360180   *....2.....2.....$.............*
7F40   357A2000 35017CF1 C2018036 4C010680   58C0870A E6C0870E 7135017C F13D4080   *.....a1B..........W.....a1. .*
7F60   38F28106 7A0413F2 87067808 13F29003   7B1813C2 0180361C 00803307 0E008033   *.2.....2.....2..#..B..........*
7F80   80330E00 80338033 0E018058 80330E01   80588031 38608058 C0107F8E 0D018058   *.....................-.........*
7FA0   8056F282 0A3A8080 590C0180 5880540C   01805F80 5BB8403A F29007BB 403A3B80   *..2.................$. .2... ...*
7FC0   80598C01 08805F2C 0F860F49 3D007FDD   F2810C35 017CF1C2 02805BC0 878C6035   *.................2....a1B..$...-.*
7FE0   017CF14C 0121802D 7B4068C2 011B13C2   021500C0 971CE4C0 971CE900 01C3C800   *.a1.....# ,B...B..........I..CH.*
8000   00023BAB 1AA01500 80000100 E000E000   3B003B00 1E00E280 0000E080 A240A240   *.....................S..... . *
8020   22202230 22282224 22260010 3BAB0001   00040000 FFF00000 00B20000 000B8060   *.....................0.........-*
8040   00000048 00000000 00000000 00000000   008036C9 00CB00C9 0000806F 8C60860F   *.....................I...I.....-..*
8060   02D7B100 31284773 47F61C4A 324C0100   03D7B100 31284777 47F61C4A 0ED58000   *.P.......6.......P.......6...N..*
8080   04D7B100 31281E51 47F61B89 0ED58000   07D74800 1B891E5A 47F61B89 40000000   *.P.......6...N...P.......6.. ...*
80A0   02D7B100 43B84773 47F64809 E2E40100   03D7B100 43B84777 47F64809 0ED58000   *.P.......6..SU...P.......6...N..*
80C0   04D7B100 43B81E51 47F61B89 0ED58000   07D74800 1B891E5A 47F61B89 40000000   *.P.......6...N...P.......6.. ...*
80E0   0DD73613 03801BC6 167733BC 168A8007   07D74800 1B132B0C 1BC61677 80000000   *.P.....F.........P.......F.......*
8100   01D7C800 00002B2A 1BC61677 80000100   02C3B100 2DC84773 480F1C35 30AC0000   *.PH......F.......C...H...........*
8120   03C3B100 2DC84777 480F1C35 0ED58000   04C3B100 2DC81E51 480F1B13 0ED58000   *.C...H.......N...C...H.......N..*
8140   07C34800 1B131E5A 480F1B13 40000000   08C391A3 035D184E 154A1808 91800023   *.C.......... ....C......+.........*
8160   07C34800 1B89188F 91949180 80000000   02C3B100 2DCC4773 480F1C38 2DC80000   *.C...............C...........H..*
8180   03C3B100 2DCC4777 480F1C38 0ED5B000   04C3B100 2DCC1E51 480F1B13 0ED5B000   *.C...........N...C...........N..*
81A0   07C34800 1B131E5A 480F1B13 40000000   01C3C800 00003BAB 1AA01500 80000100   *.C.......... ....CH..............*
81C0   0ED70000 00001B89 16770000 00008007   0DD70079 035D9034 16679027 00008005   *.P...............P...............*
81E0   07D7C800 1B132B0C 90341667 80000000   01D7C800 00002B2A 90341667 80000100   *.PH..............PH..............*
8200   08C391DC 03243DD1 154A44B5 91A30039   05C3C800 02545396 61CE625A 85350000   *.C.....J.........CH.....%.....*
8220   02C3B100 33D04773 480F1C62 33B00100   03C3B100 33D04777 480F1C62 0ED58000   *.C...............C...........N..*
8240   04C3B100 33D01E51 480F1B13 0ED58000   07C34800 1B131E5A 480F1B13 40000000   *.C...........N...C.......... ...*
8260   05C34800 C802539F 61CE8CF0 804A0000   05C34800 BC020303 61CE8CF0 804A0080   *.C..H.../..0.....C....../..0....*
8280   07C34800 00032324 61CE8CF0 00080000   1BC38000 03249034 90348CF0 05208005   *.C....../..0.....C.........0....*
82A0   07C34800 1B893E27 90349034 80000000   01C3C800 00003BAB 1AA01500 80000100   *.C...............CH..............*
```

6-Byte Table

First Entry in Core

```
82C0   0ED70010 02909034 16670000 00008005   07D7C800 00FF47D3 47F69034 08000000   *.P.............PH....L.6.....*
-------------------------------------------------------------------------------------------------------------------------------------
82E0   02D7B100 314C4773 47F61C4D 31280100   03D7B100 314C4777 47F61C4D 0ED58000   *.P.......6.......P.......6...N..*
8300   04D7B100 314C1E51 47F61B89 0ED58000   07D74800 1B891E5A 47F61B89 40000000   *.P.......6...N...P.......6.. ...*
8320   07D74800 00FF47D3 47F60000 08000000   01D7C800 0000451F 15001B89 20000100   *.P.....L.6.......PH.............*
8340   01C3C801 00003BAB 1AA01500 80000120   02C3B100 2DD04773 480F1C3B 2DCC0120   *.CH.............C.............*
8360   03C3B100 2DD04777 480F1C3B 0ED58020   04C3B100 2DD01E51 480F1B13 0ED58020   *.C...........N...C...........N..*
8380   07C34800 1B131E5A 480F1B13 40000020   09C391DC 03243DF8 154A9190 91800023   *.C.......... ....C.....8........*
83A0   01C3C801 00003BAB 1AA01500 80000100   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
83C0   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
83E0   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8400   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
8420   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8440   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
8460   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8480   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
84A0   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
84C0   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
84E0   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8500   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
8520   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8540   07C3C800 00032324 61CE8CF0 00080020   0AC38056 03249034 90348CF0 00000010   *.CH...../..0.....C.........0....*
8560   01C3C800 00013BAB 1AA01500 80000120   05C3C800 BC0257C3 61CE8CF0 804A0020   *.CH.............CH....C/..0....*
8580   05C3C800 100202DD 61CE8CF0 804B00A0   07C3C800 00032324 61CE8CF0 00080020   *.CH...../..0.....CH...../..0....*
85A0   0AC38040 03249034 90348CF0 00000110   02C3B100 2F584773 480F1CCE 2F4C0120   *.C. .......0.....C.............*
85C0   03C3B100 2F584777 480F1CCE 0ED58020   04C3B100 2F581E51 480F1B13 0ED58020   *.C...........N...C...........N..*
85E0   07C34800 1B131E5A 480F1B13 40000020   07C34800 1B893506 90341B89 20000000   *.C.......... ....C.............*
                Most Recent Entry in Core
8600   01C3C800 00023BAB 1AA01500 80000100   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   *.CH............................*
                                                Last Entry in Core
8620   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF   *................................*

8C60   34088CDF 30008CE1 0D018CE1 8CE8F281   6B0D018C E18CE5F2 8139F297 5F784068   *.............Y2.,.....V2..2... ,*
8C80   F210593D 008CE3F2 81198502 0436028C   E62C008C E2003BE0 8CE20D00 8CE28CE3   *2.....T2........W...S....S...S.T*
8CA0   F20139F0 733E3000 8CE10D01 8CE18CE5   F201293C 808C78F0 70313000 8CE10D01   *2..0...........V2.....#0.......*
```

READER'S COMMENT FORM

IBM System/3
Model 10 Disk System
Communications Control Program
System Reference

GC21-7588-0
S3-36

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.

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
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM m/3 CCP System Reference Printed in U.S.A. GC21-7588-0

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)